Honeywell

Experion

Control Builder Components

Theory

Volume 1 of 2

EP-DCXX84
R300.1
5/06

Notices and Trademarks

Copyright 2006 by Honeywell International Inc.
Release 300.1 **May 5, 2006**

While this information is presented in good faith and believed to be accurate, Honeywell disclaims the implied warranties of merchantability and fitness for a particular purpose and makes no express warranties except as may be stated in its written agreement with and for its customers.

In no event is Honeywell liable to anyone for any indirect, special or consequential damages. The information and specifications in this document are subject to change without notice.

Honeywell, PlantScape, Experion, and **TotalPlant** are registered trademarks of Honeywell International Inc.

Other brand or product names are trademarks of their respective owners.

Honeywell International

Process Solutions

2500 West Union Hills

Phoenix, AZ 85027

1-800 343-0228

About This Document

Provides general and detailed theoretical or how it works information for selected Control Builder related functions and the control library function blocks. It does not cover the hardware associated blocks like the Control Processor Module (CPM) and Input/Output Module blocks.

Release Information

Document Name	Document ID	Release Number	Publication Date
Control Building User's Guide - cb_g Volume 1 of 2	EP-DCXX64	300.1	5/06

References

The following list identifies all documents that may be sources of reference for material discussed in this publication.

Document Title

Contacts

World Wide Web

The following Honeywell web sites may be of interest to Process Solutions customers.

Honeywell Organization	WWW Address (URL)
Corporate	*http://www.honeywell.com*
Honeywell Process Solutions	*http://hpsweb.honeywell.com*

Telephone

Contact us by telephone at the numbers listed below.

Location	Organization	Phone
United States and Canada	Honeywell IAC Solution Support Center	1-800-822-7673
Europe	Honeywell TAC-EMEA	+32-2-728-2704
Pacific	Honeywell Global TAC - Pacific	1300-300-4822 (toll free within Australia) +61-8-9362-9559 (outside Australia)
India	Honeywell Global TAC - India	+91-20-2682-2458
Korea	Honeywell Global TAC - Korea	+82-2-799-6317
People's Republic of China	Honeywell Global TAC - China	+86-10-8458-3280 ext. 361
Singapore	Honeywell Global TAC - South East Asia	+65-6580-3500
Taiwan	Honeywell Global TAC - Taiwan	+886-7-323-5900
Japan	Honeywell Global TAC - Japan	+81-3-5440-1303
Elsewhere	Call your nearest Honeywell office.	

Symbol Definitions

The following table lists those symbols used in this document to denote certain conditions.

Symbol	Definition
	ATTENTION: Identifies information that requires special consideration.
	TIP: Identifies advice or hints for the user, often in terms of performing a task.
	REFERENCE -EXTERNAL: Identifies an additional source of information outside of the bookset.
	REFERENCE - INTERNAL: Identifies an additional source of information within the bookset.
CAUTION	Indicates a situation which, if not avoided, may result in equipment or work (data) on the system being damaged or lost, or may result in the inability to properly operate the process.
	CAUTION: Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury. It may also be used to alert against unsafe practices. **CAUTION** symbol on the equipment refers the user to the product manual for additional information. The symbol appears next to required information in the manual.
	WARNING: Indicates a potentially hazardous situation, which, if not avoided, could result in serious injury or death. **WARNING** symbol on the equipment refers the user to the product manual for additional information. The symbol appears next to required information in the manual.

Symbol	Definition
	WARNING, Risk of electrical shock: Potential shock hazard where HAZARDOUS LIVE voltages greater than 30 Vrms, 42.4 Vpeak, or 60 VDC may be accessible.
	ESD HAZARD: Danger of an electro-static discharge to which equipment may be sensitive. Observe precautions for handling electrostatic sensitive devices.
	Protective Earth (PE) terminal: Provided for connection of the protective earth (green or green/yellow) supply system conductor.
	Functional earth terminal: Used for non-safety purposes such as noise immunity improvement. NOTE: This connection shall be bonded to Protective Earth at the source of supply in accordance with national local electrical code requirements.
	Earth Ground: **Functional earth connection.** NOTE: This connection shall be bonded to Protective Earth at the source of supply in accordance with national and local electrical code requirements.
	Chassis Ground: Identifies a connection to the chassis or frame of the equipment shall be bonded to Protective Earth at the source of supply in accordance with national and local electrical code requirements.

Contents

CONTROL BUILDER COMPONENTS.....1

Some Underlying Concepts 1
Introduction 1
In the beginning or the Single-Loop Controller Reference.....1
Experion architecture - partitioned functionality.....3
Blocks for building process control operations 5
Naming convention– independent versus dependent.....7
Full or expanded tag name.....9
Parameter names.....9
Naming restrictions and conventions.....11
Parameter data types 12
Data flow -– active versus passive 12
Active and passive connectors.....13
Cascade loop connections 13
Data pull or push 15

Control Capacity and Performance..... 16
Control network capacity 16
C200 configuration options.....18
C200 Redundancy.....19
C200 communication performance.....20
C200 processing resources.....22
C200 Memory Resources.....22
C300 configuration options.....23
C300 Redundancy.....24
C300 communication performance.....25
C300 processing resources.....26
C300 Memory Resources.....27
ACE configuration options.....27
ACE communication performance.....28
ACE processing resources.....29
ACE Memory Resources.....30

I/O Capacity and Performance..... 30
C200 IO capacity.....31
C300 IO capacity.....33
IO Unit load factors 34

Function Block Execution Schedules..... 35
Schedule consideration differences.....35
Control Module and Sequential Control Module FBs schedule 35
CM Component Function Block scheduling.....39

IOM FB schedule 42
CPM, ACE, and CEE FBs schedule 42
Cycle overruns 43

Block Configuration Load Considerations 44
About load considerations 44
Data categories 44
Container and self-standing blocks load versus states 45
Load error messages 46
RAM Retention Start Up (RRSU) 47

Memory Usage for CEE on CPM or CEEACE 48
Typical Requirements 48

CPU Utilization for CEE on CPM 52
CPU load categories 52
CPU utilization limits and estimates 52

CEE Standard Control Blocks 52

RELATIVE REFERENCES 55

About Relative References 55

Relative References structure rules 56
How relative reference matching function works 57

Display options for relative references 58
How short name display option works 59

Control Builder functions supporting relative references 59

Control Builder blocks supporting expressions 60

Example of TRANSITION block using relative references 60
A container reference from a chart 61
A reference to a parameter on the same block from a form 61
A reference to a parameter on a different block in the same CM 61
Expression display interaction 62

Example of parameter connector using relative references 62
A reference to a parameter on a different block in the same CM 63
Parameter connector display interaction 63

Example of other parameters using relative references 64
A reference to a parameter on a basic block of the CM 65
Other parameters display interaction 65

Bulk Builder and Bulk Editor support 65

PEER-TO-PEER FUNCTIONALITY 67

Basic Peer-to-Peer Design Concepts 67
About Peer-to-Peer 67
Data flow models 68
Peer environments and subscription periods 69
Software Architecture for CPM/CEE 69
A word about ACE/CEE architecture 71

Implications for Control Builder Configuration 72
Multiple CPMs and ACEs 72
CEE execution and subscription rates 72
Function block support 73
Peer-to-peer connections and DEF and REF blocks 75
Peer-to-peer configuration example 75
Peer-to-peer configuration guidelines 79

TIME SUPPORT IN EXPERION SYSTEM 81

Date and Time Synchronization 81

C300 and Series C Fieldbus Interface Module Time Synchronization 82
BOOTP Mechanism and Multiple Clusters 82

Time Configuration Considerations..... 83
Time Data Types in Control Execution Environment (CEE) Blocks 83
Time Data Types in User Algorithms and Displays 83
C200, Application Control Environment (ACE) Time Parameters 85
Custom Algorithm Block (CAB) on ACE Time Functions 86

Examples of Time Usage in CAB and SCM 87
Example - Custom Algorithm Block (CAB) writes Time CDP 87
Example - Custom Algorithm Block (CAB) reads Time CDP 88
Example – Sequence Control Module (SCM) waits for a specified Time 88
Example – Sequence Control Module (SCM) records Time of process command 89

COLD AND WARM RESTART FUNCTIONALITY.....91

Overview 91
CEE Initial Startup 91
CEE Restarts 91
CEE Restart Behaviors 92

Planning 93
Invariant or Variant Restart Behaviors 93
Blocks with Invariant Restart Behavior 93
Blocks with Variant Restart Behavior 96

Configuration of Restart Behaviors 99
CEE Block 99
Control Module (CM) 100
Control Module Function Blocks 100
Warm and Cold Restart Behaviors of Cascades 103

Operations 111
CEE Function Block 111
ACE Function Block 112
CPM Function Block 112

EXTERNAL OPC SERVER SUPPORT 113

OPC Data Access 113
OPC Server function block 113
OPC client/server data flow 113

OPC Data References 115
OPC Data Name Syntax 115
Parameter connectors only 115
OPC references in expressions 116
OPC references in SCM Alias table 117

OPC Data Type Conversions 118
About data type 118
Gets conversions 118
General data conversion considerations 120
Stores conversions 122

ACE interface to TPS system as OPC server 123
ACE recognizes HCI 123

SYSTEM NAMESPACE WORKAROUND 125

Inter-Cluster Communication 125
Functional Overview 125

Using the Inter Cluster Gateway for Communication between Experion Clusters 126
Overview 126
Inter Cluster Gateway configuration guidelines 126
When to use Inter Cluster Gateway 126
Reviewing Inter Cluster Gateway characteristics 128
Inter Cluster Gateway connectivity 130

Remote EEOUT (REEOUT) Function Block 130
Functional description 130

CONTROLLER REDUNDANCY FUNCTIONALITY.....131

Basic Redundancy Design Concepts 131
About Controller redundancy.....131
Switchover and Secondary readiness 133
Failure conditions and switchover 134
Role of the Redundancy Module 136

Control Module Independence 138
CM Independence Background.....138
CPU Utilization 139
Communication Bandwidth Utilization 140
Data Reference 141

Implications for Control Builder Functions..... 142
Redundant C200 CPM configuration.....142
RM configuration 142
RM monitoring.....143
RM/RCP dialog box.....145
Main tab 146
Summary tab.....147
RM Profiles tab.....150
Configuration tab.....152
Synchronization tab.....154
Chassis Profiles tab.....158
Display tab 160
Sever History tab.....161
Sever Displays tab 163
Auto-Synchronization events.....165

CONTROL MODE SHED ON LOSS OF I/O FUNCTIONALITY.....167

Basic Control Mode Shed Design Concepts..... 167
About Control Mode shed on loss of I/O.....167
How it works.....167

Implications for Operation 169
Resetting mode after I/O communications are restored. 169
Allowing Redundancy synchronization with lost I/O communications.....169

Remote Cascade function between Controllers..... 170

CONTROL BUILDER EXPORT AND IMPORT FUNCTIONALITY 171

Basic Export/Import Design Concepts 171
About Control Builder Export and Import functions.....171
Export functionality.....171

Import functionality 173
Export/Import functional overview 174

Some Operation Considerations 175
Export/Import usage notes 175

SCM AND CM CHART VISUALIZATION FUNCTIONALITY 177

Basic Chart Visualization Design Concepts 177
About Chart Visualization 177
How SCM chart visualization works 177

Some SCM and CM Chart Operation Considerations 182
Detail display interaction notes 182

I/O LINK INTERFACE 183

I/O Link Interface 183
Seamless integration 183
I/O Functions 183
A word about Point form 183
IOP validation 184

I/O Link Cable Status and Statistics 184
Periodic swap functionality 184
Periodic swap transition diagram 185
To enable periodic channel swap 186
Viewing cable status and statistics 187
To view channel/cable status and statistics parameters 187

PROCESS MANAGER INPUT/OUTPUT FUNCTIONALITY 189

High and Low Level Analog Input Points 189
Function 189
PV Characterization 193
Linear Conversions 193
Square root conversion 194
Thermal conversion 195

Smart Transmitter Interface Point 197
Smart Transmitter support 197
Multivariable transmitter support 197
Transmitter parameters and database access 198
STI parameter comparisons 199
Transmitter communication mode 200
STI IOP commands 201

Point states 202
STI IOP functions 203
STI PV characterization....204
STI linear conversion....205
STI square root conversion 206
STI thermal conversion 206
STI PV range checking and filtering 206

Analog Output Point 208
AO functions....208
AO direct/reverse output 209
AO output characterization....209
AO calibration compensation 210

Digital Input Point 211
DI functions 211
DI status point 213
DI PV source selection 213
DI off-normal alarming....213
Alarm delay 214
Event reporting....214
DI latched input point....214
DI sequence of events point....215
SOE definitions 215
SOE resolution considerations 216
DI SOE configuration considerations....217

Digital Output Point 220
DO functions 220
Pulse Width Modulated (PWM) Output Type....222
Status Output Type 223
Initialization request flag....223

COMPONENT CATEGORIES AND TYPES 225

Overview 225
About categories 225
Function block types and Data Organization 225

REGULATORY CONTROL 227

Regulatory Control Blocks....227
Functional overview....227
Common regulatory control functions for C200 and ACE controllers without insertion points231
Common regulatory control functions for ACE controllers with insertion points....233
CAB insertion configuration considerations for regulatory control blocks 235

Insertion type functional characteristics for regulatory control blocks....236
Functional requirements for Custom CAB algorithm for requlatory control block....238
CAB insertions for Regulatory Control block parameters....240
Pin connections to inserted CAB instances....241
Insertion status and fail alarm for regulatory control blocks....242
Handling of insertion failure on regulatory control block....242
CAB insertion on regulatory control block configuration examples....243

AUTOMAN (Auto Manual) Block....251
Description....251
Function....260
Configuration example....261
Inputs....263
Output....263
Initializable inputs and outputs....264
Output ranges....264
Output bias....265
Mode Handling....267
Timeout Monitoring....267
Control Initialization....268
Secondary initialization option....269
Override feedback processing....269
Windup processing....270
Anti-Reset Windup Status....272
AUTOMAN parameters....273

ENHREGCALC (Enhanced Regulatory Control Calculator) Block....274
Description....274
Function....287
Configuration example....288
Operating modes and mode handling....289
Inputs....289
Initializable input....290
SP limit checking....290
SP target value processing....290
Initializable outputs....294
Output ranges and limits....295
Assignable outputs....296
Output assignment rules....297
Control initialization....298
Output bias....299
Timeout monitoring....302
Timeout processing....302
Mode shedding on timeout....303
Override feedback processing....303
Windup handling....305
Windup processing....306
Anti-Reset Windup Status....309

Expressions....310
Parameters in Expressions....313
Guidelines for Writing Expressions....313
Enable/Disable switch example expression....314
String data support in expressions....315
Time support in expressions....315
Restart or Point Activation....318
ENHREGCALC parameters....318

FANOUT Block....319
Description....319
Function....328
Configuration example....328
Inputs....329
Outputs....329
Initializable inputs and outputs....329
Output ranges....330
Output bias....331
Mode handling....333
Timeout monitoring....333
Control initialization....334
Secondary initialization option....336
Override feedback processing....336
BACKCALC processing....336
Windup processing in FANOUT block....337
Windup processing in Regulatory blocks....337
Anti-Reset Windup Status....340
FANOUT parameters....341

INCRSUMMER (Incremental Summer) Block....342
Description....342
Equation....352
Configuration example....353
Function....354
Inputs....354
Outputs....354
Initializable inputs and outputs....355
Output Ranges....355
Output bias....356
Mode handling....357
Timeout monitoring....358
Control initialization....358
Override feedback processing....359
Windup processing....359
INCRSUMMER parameters....362

OVRDSEL (Override Selector) Block....363
Description....363

Function 373
Configuration example 376
Configuration considerations 379
Inputs 380
Input ranges 380
Input descriptors 380
Initializable outputs 380
Output ranges and limits 381
Mode handling 382
Timeout monitoring 382
Timeout processing 382
Bypass processing 383
Bad input option 383
Equations 384
Input switching 384
Output bias 384
Bad CV processing 385
Control initialization 385
Restart or function block activation 386
Override feedback propagation 386
Recommendations on configuring override strategies 387
Windup processing 387
Anti-Reset Windup Status 390
OVRDSEL parameters 391

PID Block 392
Description 392
Function 407
Functional scenario 409
Configuration examples 411
Operating modes and mode handling 414
Required inputs 415
Input ranges and limits 415
Initializable outputs 416
Control initialization 417
Output bias 417
Output ranges and limits 420
Output limit behavior for PID block recovering from Bad Control 420
Direct or reverse control 421
Set Point Ramping 422
PV tracking 426
PID equations 427
Gain options 429
Tuning Constant Change Considerations 431
Timeout monitoring 431
Timeout processing 431
Windup handling 432
Windup processing 432

Anti-Reset Windup Status 435
Override feedback processing....436
PID parameters 437

PID-PL (Profit Loop PKS) Block 439
Description 439
Function 455
Configuration examples....464
Operating modes and mode handling 471
Required inputs 472
Input ranges and limits 472
Initializable outputs....472
Control initialization 473
Output bias....473
Output ranges and limits 473
Output limit behavior for PID-PL block recovering from Bad Control....473
Direct or reverse control 473
Set Point Ramping 474
Range Control Ramping....474
PV tracking....475
PV Bad Value Handling....475
PV Calibration 476
Asynchronous PV Inputs 476
Control Equations....477
Tuning Constant Change Considerations....478
Timeout monitoring 478
Windup handling....478
Windup processing....478
Anti-Reset Windup Status 481
Override feedback processing....482
PID-PL parameters 482

PIDER (PID with External Reset Feedback) Block.... 483
Description 483
Configuration example 485
Function 500
Operating modes and mode handling 501
Required inputs 502
Input ranges and limits 503
Initializable outputs....504
Control initialization 505
Output bias....505
Output ranges and limits 508
Direct or reverse control 508
Set Point Ramping 509
PV tracking....513
PID equations....514
PIDER equation variations 516

Gain options 517
Tuning Constant Change Considerations 519
Timeout monitoring 519
Timeout processing 520
Windup handling 521
Windup processing 521
Anti-Reset Windup Status 524
Override feedback processing 525
Error handling for RFB and TRFB inputs 525
Restart or point activation 525
PIDER parameters 525

PIDFF (PID with Feedforward) Block 526
Description 526
Function 543
Functional scenario 544
Operating modes and mode handling 547
Required inputs 547
Input ranges and limits 548
Initializable outputs 549
Control initialization 550
Output bias 550
Output ranges and limits 552
Output limit behavior for PIDFF block recovering from Bad Control 553
Direct or reverse control 553
Set Point Ramping 554
PV tracking 559
Feedforward add or multiply action and equations 560
Feedforward value status 562
PID equations 563
Gain options 565
Tuning Constant Change Considerations 567
Timeout monitoring 567
Timeout processing 567
Windup handling 568
Windup processing 568
Anti-Reset Windup Status 571
Bypassing feedforward control action 572
Override feedback processing 573
PIDFF parameters 574

Tables

Table 1 Expression operators, functions, and strings reference 310

Figures

Figure 1 Generic single-loop controller functions. 2
Figure 2 Simplified overview of Experion architecture. 4
Figure 3 Typical linking of Function Blocks through Control Builder. 6
Figure 4 Component block names are dependent on container block tag name for system wide recognition. 8
Figure 5 Sample Control Builder configuration with sample tag name assignments. ..10
Figure 6 Sample PID cascade loop configuration 14
Figure 7 Cycle time loading for sample container block configurations for a 50 ms CEE. 36
Figure 8 Peer-to-peer is layered upon CDA connection-oriented communications. 70
Figure 9 Example of DEF and REF block functions in CB configuration using REMCAS block. 77
Figure 10 Typical RCP setup in 10-slot chassis. 132
Figure 11 Typical redundancy communications path. 136
Figure 12 RM/RCP dialog box. 144
View of example_scm chart in Control Builder Monitoring Tab. 178
View of example_scm chart in SCM Detail display Chart Tab in Station. 179
View of CM102 chart in Control Builder Monitoring Tab. 180
View of CM102 chart in CM Detail display in Station. 181
Figure 13 CAB insertion locations in regulatory control block major functions 234
Figure 14 Configuration example using single CAB insertion for requlatory control block 243
Figure 15 Configuration example using multiple CAB insertions for regulatory control block. 246
Figure 16 Configuration example using two CAB insetions of the same type for regulatory control block. 249
Figure 17 Example of CB configuration using AUTOMAN block. 261
Figure 18 Example of CB configuration using ENHREGCALC block. 288
Table 1 Expression operators, functions, and strings reference 310
Figure 19 Example of CB configuration using OVRDSEL block. 376
Figure 20 Simple cascade control loop example. 393
Figure 21 Functional block diagram of typical PID cascade operation. 409
Figure 22 Example of CB configuration using a PID block for single loop control. 411
Figure 23 Example of CB configuration using two PID blocks for cascade loop control. 413
Figure 24 PID block with SP ramping parameters configured for monitoring. 425
Figure 25 PIDER block used in a cascade control strategy with PID, AUTOMAN, and NUMERIC blocks 485
Figure 26 PIDER block with SP ramping parameters configured for monitoring. 512
Figure 27 Simple PID with feedforward control loop example. 528

Figure 22 Functional block diagram of typical PID feedforward operation..... 545
Figure 28 PID block with SP ramping parameters configured for monitoring. 557

Control Builder Components

Some Underlying Concepts

Introduction

As with any emerging technology, the Experion system has spawned its share of new or recast terms to define its unique features. In most cases, these terms clearly relate form and content with function, so you can intuitively determine their meaning. This section reviews some basic concepts behind the following Control Builder terms.

- Function Block (FB)
- Control Execution Environment (CEE)
- Independent Name
- Dependent Name
- Container Block
- Self-Standing Block

In the beginning or the Single-Loop Controller Reference

Before reviewing Experion concepts, it may be helpful to quickly review single-loop controller operation as a point of reference. A single-loop controller typically includes the following process control related functions in one form or another as shown in the following figure.

- Human Interface
- Communications Driver
- Control Data Processor
- Input/Output (I/O) Interface

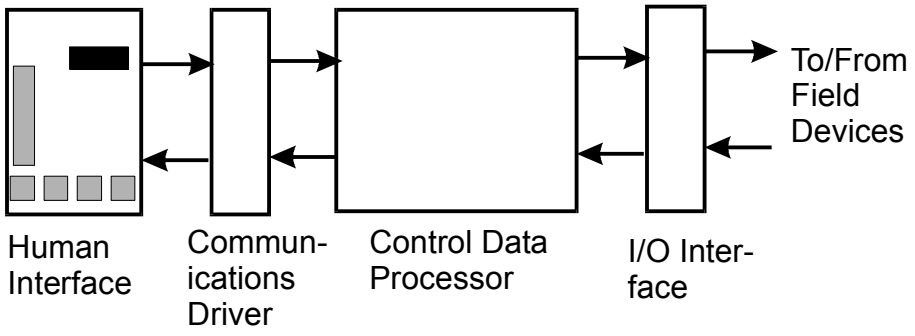


Figure 1 Generic single-loop controller functions.

A brief description of each of these functions is given in the following table.

Function	Description
Human Interface	The user's "window-on-the-process" as well as the configuration for the controller's functionality. It usually provides varying levels of access for viewing and changing process related parameters, such as the: • Process Variable (PV) • Set Point (SP) • Output (OP) • Tuning Constants, and • Alarm Conditions. In addition, operators can view entries made in the controller's configuration database, but changes are usually under keylock or password control.
Communications Driver	The communications driver serves as the translator for the data that flows between the human interface and the control data process or functions. It translates signals into appropriate display data or control action.

Function	Description
Control Data Processor	The control data processor defines the operating characteristics for the controller which are usually stored in memory as the controller's configuration database. It solves the configured or selected Proportional, Integral, and Derivative (PID) control equation and usually runs self-diagnostic tests.
I/O Interface	The I/O interface links all analog and digital I/O to the control data processor for communications with field devices. It provides any signal conversion needed to condition an input or output for use by the processor or field device.

Experion architecture - partitioned functionality

While we will not get into internal design details, it may be helpful to take a high-level look at how Experion works. In a very simple way, the following figure shows how Experion partitions control operations among its hardware components.

Experion features an object oriented system environment. This means it is based on the technology of Object Linking and Embedding (OLE) which promotes the partitioning of functions among multiple subsystems. The Control Builder is the heart of Experion functionality. It includes libraries of objects that are easily configured graphically to emulate the generic control operations of communications driver, control data processor, and I/O Interface as well as providing a run-time monitoring capability.

Experion includes several repositories for supporting build-time and real-time data exchange and storage. Data flow is based on client/server relationships where the client pulls data or the server pushes data.

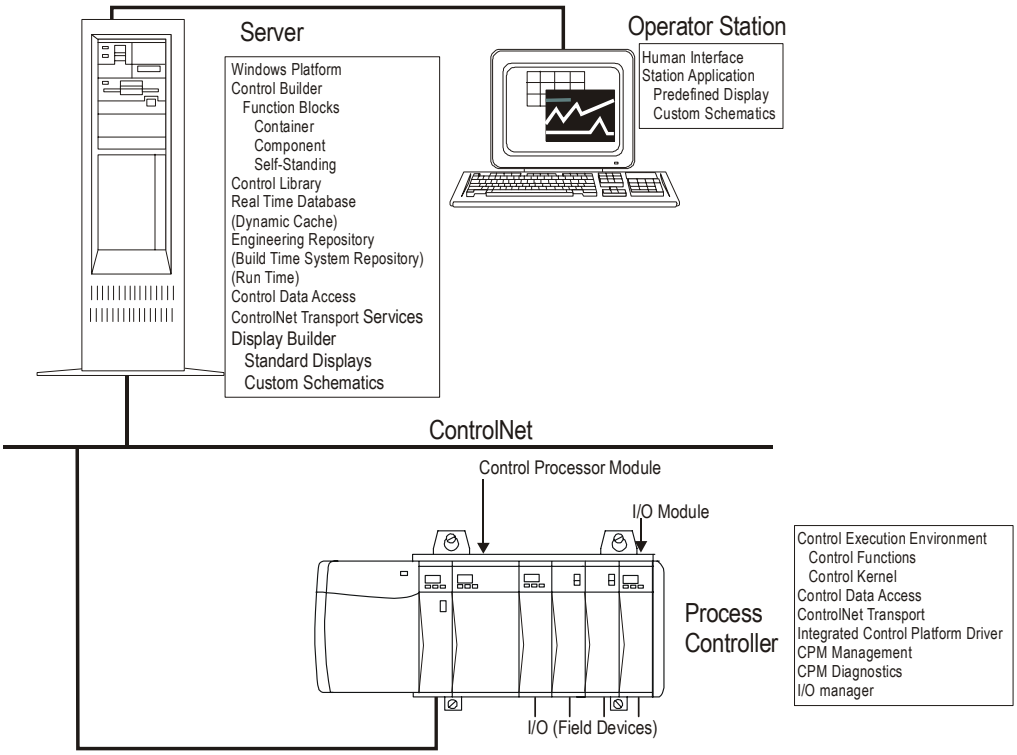


Figure 2 Simplified overview of Experion architecture.

Blocks for building process control operations

If we had to use one term to sum up Experion's Control Builder application, it would be Function Block. Essentially, a Function Block is an executable software object that performs a specific task. Control Builder has libraries of Function Blocks that let you graphically build the exact control operations you need for your process. There are three major types of blocks as listed below.

Block Type	Description	Name of example block in Control Builder
Container	A container block can "contain" other component blocks. It appears as a chart in Control Builder into which component blocks are placed.	Control Module (CM) Sequential Control Module (SCM)
Component	A block which exists only as a component of a container block. It appears as a generic named block with configurable pins and parameters within a container block in Control Builder. Please note that a component block may also be referred to as a Basic Function Block, or just a Basic Block.	PID (All blocks listed in CM and SCM Libraries in Control Builder, with the exception of CM and SCM, of course.)
Self-Standing (or Stand-Alone)	A block that stands on its own. It is neither a container for other blocks or a component of a container block. It appears as an icon in the menu area of Control Builder.	TC-IAH061 (All blocks listed in Input/Output Module (IOM) library as well as related blocks in Rail I/O and Pulse Input libraries in Control Builder.) CPM CEE RM

In this document, we use "Function Block" as a generic term, which applies to all three types of blocks, listed above. Once you begin using the Control Builder application, you will be able to readily associate block type with the graphic style used to represent a given Function Block on the display.

ATTENTION

The HANDLER blocks are component type blocks even though they do contain STEP and TRANSITION function blocks. Within the CEE, they are implemented as components of the SCM block and not as container type blocks.

The following figure gives a block diagram view of how FBs are typically linked through Control Builder configuration.

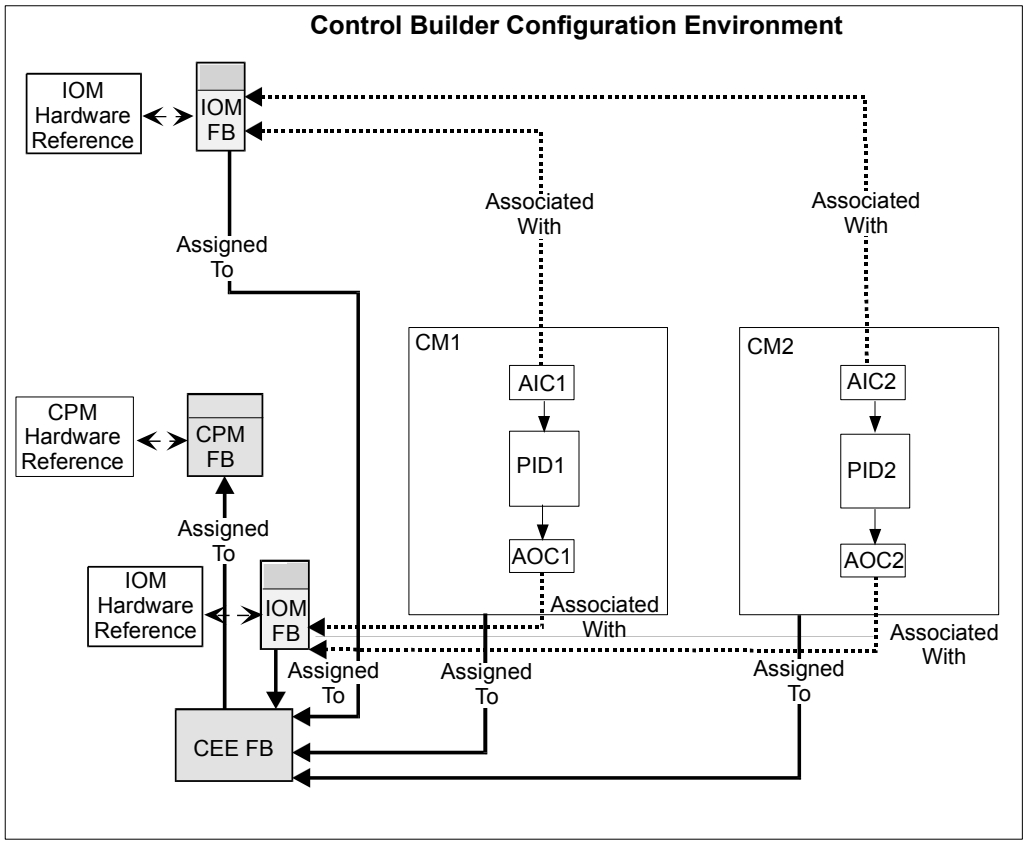


Figure 3 Typical linking of Function Blocks through Control Builder.

Naming convention– independent versus dependent

Every computer based application uses a more-or-less common naming convention for identifying data created within the application. If you were to use a word-processing program on you personal computer to draft a letter to Honeywell, you might assign a file name like HONLET1.DOC to save the letter as a file on your hard disk. In this case, the first part of the name represents a personal choice and the suffix or second part identifies the application used to create the file.

Like other computer applications, Control Builder employs a naming scheme to uniquely identify Experion system information. In most cases, Control Builder assigns default names for blocks, which you can change through configuration. We commonly refer to the name of a block as its Tag Name. This is a carry over from the Tag Number convention used to identify components on a process Piping and Instrumentation Diagram.

While the Experion system obviously recognizes all “Tag Names”, the tag names for component type blocks are considered “dependent” names and those for container and self-standing type blocks are “independent” names. The independent names are unique Tag Names within Experion by default. This means a block with an independent name receives system wide recognition without any other qualifications.

The dependent names are Tag Names that uniquely identify component blocks only within their container module. This means component blocks in different container modules can have the same Tag Name. For example, If you have two Control Modules named CM724 and CM725, you can have a PID block named PIDA in CM724 and in CM725 as shown in the following figure. In this case, the Tag Name PIDA is dependent on the Control Module’s independent name of CM724 or CM725 for system wide recognition.

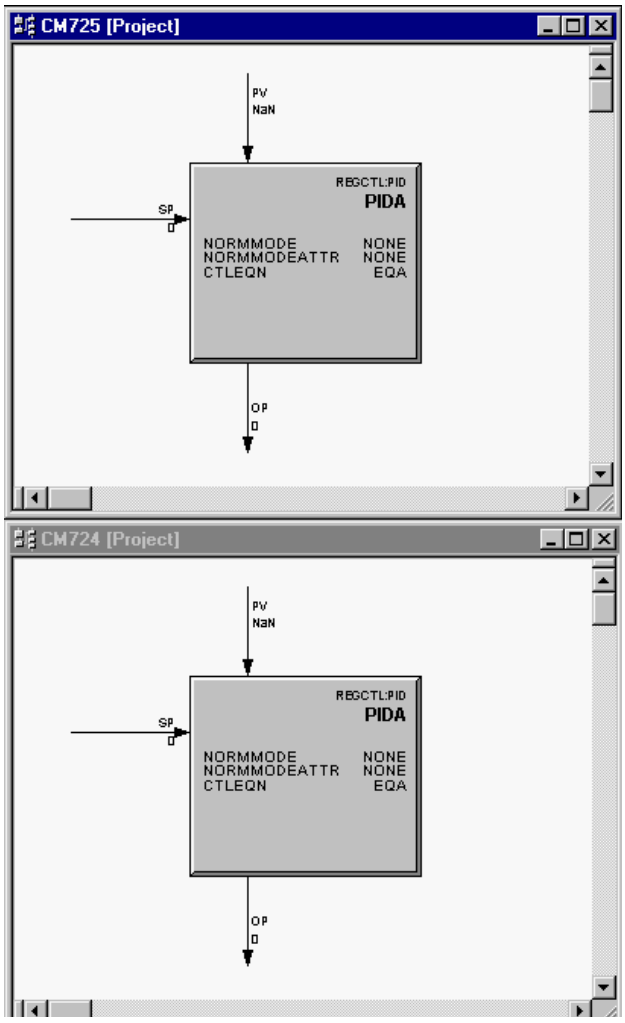


Figure 4 Component block names are dependent on container block tag name for system wide recognition.

Full or expanded tag name

You must prefix the component block Tag Name (or dependent name) with its associated container block Tag Name (or independent name) when you need to provide a reference to a specific component for system wide recognition. We call this independent and dependent name combination the Full or Expanded Tag Name. A Full Tag Name has this general format for a component type function block.

- <Independent Tag Name>.<Dependent Tag Name>

For example, the Full Tag Name for a PID block named PIDA in a Control Module named CM1 would be:

- CM1.PIDA

Parameter names

The parameters associated with a given Function Block have preassigned names. These parameter names are dependent type names. This means you must prefix a parameter name with its appropriate Tag Name or Full Tag Name when you need to provide reference to a specific parameter on a system wide basis. A parameter name has one of these general formats for system wide recognition depending upon whether it is associated with an independent or a dependent type Function Block.

- For an independent type block: <Independent Tag Name>.<Parameter Name>
- For a dependent type (component) block: <Independent Tag Name>.<Dependent Tag Name>.<Parameter Name>

For example, to reference the output (OP) parameter of a PID block named PIDA in a Control Module named CM1; you would identify the parameter as follows:

- CM1.PIDA.OP
- To reference the Execution State (EXECSTATE) parameter of a Control Module named PIDLOOP, you identify the parameter as follows:
- PIDLOOP.EXECSTATE

The main thing to remember about naming is that you must specify a unique name for the Function Block or parameter that you want recognized on a system wide basis.

The following figure illustrates some typical Tag Name assignments used in a sample Control Builder configuration.

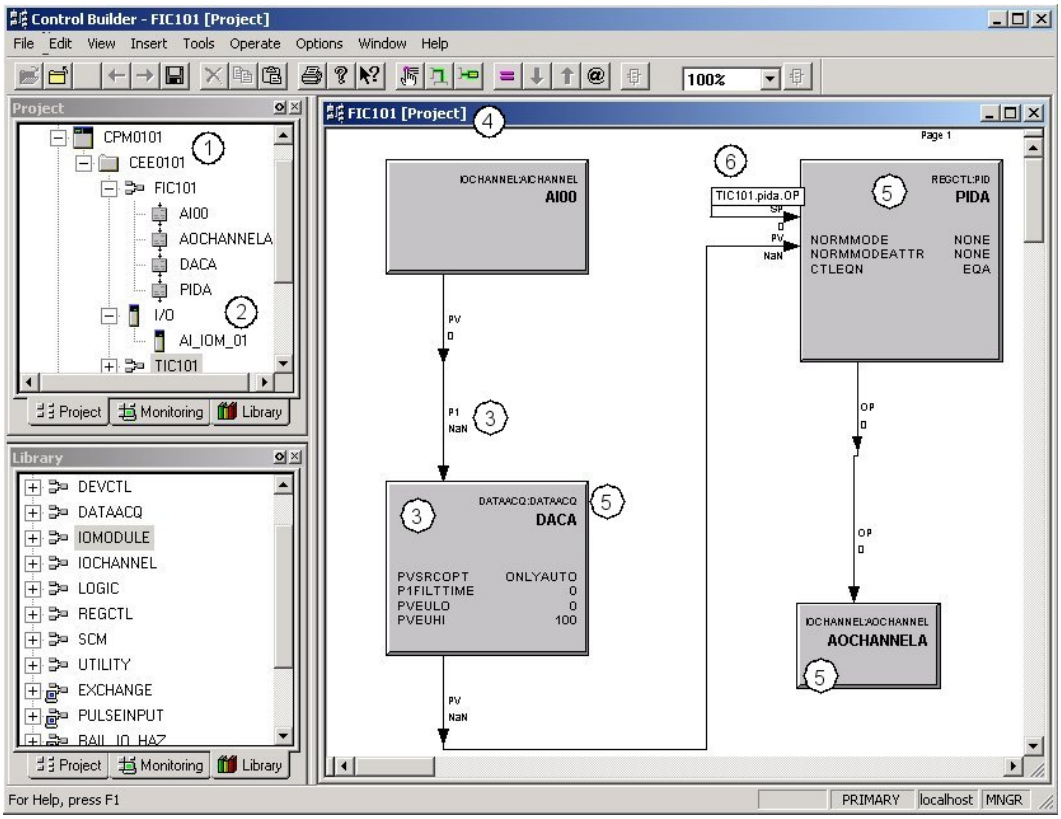


Figure 5 Sample Control Builder configuration with sample tag name assignments.

See the following table for a description of the callouts in the figure above.

Callout	Description
1	Tag names for CPM and CEE with MAC address of 01 and CPM installed in slot number 1. (CPM0101, CEE0101) Please note that the format for these tag names is used for example purposes only.
2	Configured tag name for analog input type IOM FB. (AI_IOM_01)

Callout	Description
3	Predefined names for FB parameters. (PVSRCOPT, PIFILTIME, PVEULO, PVEUHI, P1)
4	Configured tag name for Control Module. (FIC101)
5	Configured dependent block names for component FBs. (DACA, PIDA, AI00, AO00)
6	Configured full tag name for connection to a parameter in another Control Module. (TIC101.PIDA.OP)

Naming restrictions and conventions

The naming restrictions apply to point, area, and parameter naming.

- The following characters are restricted and may not be used:
  ` ~ ! @ # $ % ^ & * + - = ( ) { } [ ] | \ \ : ; ' < > , . ? / \ "
  - Dots are allowed in parameter naming only.
- An embedded space or leading space is not allowed.
  - Spaces are allowed in area naming only.
- Tabs may not be used.
- The name may not be null (no characters).
- One of these characters must be present in a valid name:
  ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
- The name must be less than the maximum size allowed for a tagname:
  - 16 for composites.
  - 15 for basic blocks.
  - 100 for parameter names.
- The name must be unique.

Parameter data types

Parameter values are expressed in one of the following major data types.

- BOOLEAN
- ENUM (Enumeration)
- INT16 (Signed 16-bit Integer)
- INT32 (Signed 32-bit Integer)
- UINT16 (Unsigned 16-bit Integer)
- UINT32 Unsigned 32-bit Integer)
- FLOAT32 (32-bit IEEE Floating Point)
- FLOAT64 (64-bit IEEE Floating Point)
- TIME
- STRING

In most cases, only output and input parameters with matching data types can be connected. One exception is with input and output connections for SCM blocks.

Data flow –– active versus passive

Before control data can flow in a Experion system, you must load the Control Execution Environment (CEE) FB and assign it to the Control Processor module that will contain the control strategy loaded from the Control Builder. You must also assign the Control Module to the CEE through Control Builder configuration. The procedures associated with these activities are covered in the *Control Building Guide*. They are only mentioned here to remind you that you must configure hardware as well as functional control operations through the Control Builder.

The CEE provides the control execution services for the Control Processor. It coordinates all data flow between parameters at execution time. Without getting into a lot of detail, the CEE parameters exist only within the Function Blocks for the loaded control strategy. These parameters are classified as “active” or “passive” based on whether or not the block itself handles the data flow connection processing. The active parameters require special connection processing for actions such as status handling or initialization propagation from inside the block. The passive parameters depend on activity outside the block to initiate data flow in or out. The Control Module Function Block is the agent for data flow between connected passive parameters.

Active and passive connectors

The CEE has active and passive connectors that support the functionality of active and passive parameters, respectively. These connections cause data flow to occur between connected parameters at block execution time. For active parameters, the data flow caused by connection processing is phased so that the block algorithm itself executes the transfer at the start of its execution. For passive parameters, the data flow is phased so that the Control Module FB executes the transfer just before it starts the execution of the block algorithm.

An active connector allows the block algorithm to read connection status as part of its processing and take action based on that status. A passive connector does not allow a block to determine its connection status and it returns a failsafe value in response to a connection break caused by a communication or configuration error. Depending on the data type, the failsafe value is OFF, 0 (zero), NaN (Not a Number), or blank.

For Control Module configuration, you can assume that the given failsafe value will appear for a passive input parameter when its connection is broken. Both active and passive connectors can reference parameters on blocks within the given container block or outside the container block. The active connector allocation is noted for each parameter in the *Control Builder Parameter Reference*.

Cascade loop connections

The following figure shows typical connections configured to provide a PID cascade loop with full initialization handling. The connections between the primary OP and secondary SP parameters must be made by the user during Control Module configuration. Control Builder automatically makes implicit/hidden connections between the primary BACKCALCIN and secondary BACKCALCOUT parameters. The cascade operation is also complemented by a general-purpose connection within the controller that is automatically configured when a user associates an IOCHANNEL FB with a particular channel on an IOM FB. This connection handles all of the data flow between the IOCHANNEL FB and the IOM FB. It allows one-shot initialization processing up the cascade chain in a single execution cycle.

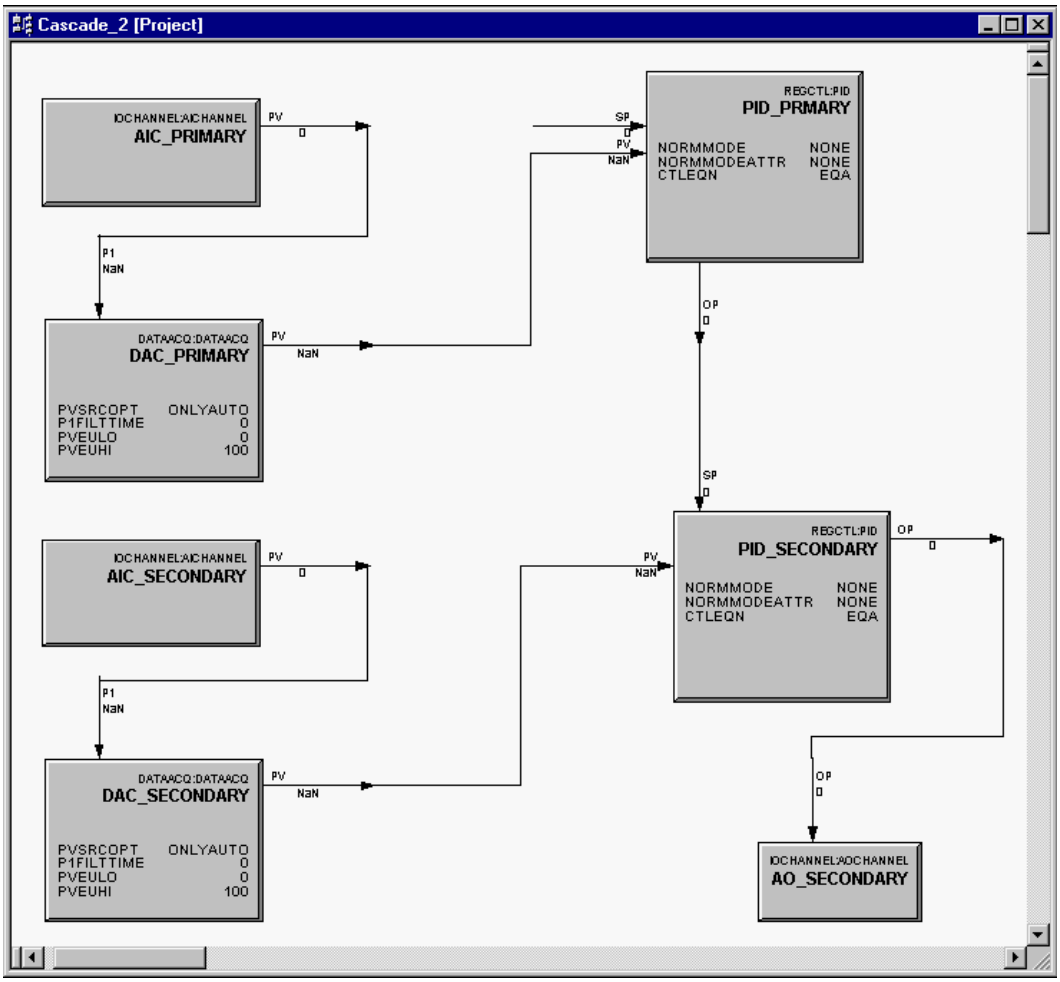


Figure 6 Sample PID cascade loop configuration .

Data pull or push

We use Pull or Push to describe how data flows across connections. In general, a block's active input parameter pulls data while a block's active output parameter pushes data. This information is for reference only, since the CEE uses a connection scheme that readily recognizes whether a Pull or a Push is causing data flow.

For Control Module configurations, active parameters always pull data and passive parameters connected between Control Module FBs also pull data. Passive parameter connections between blocks in the same Control Module FB are neither pull nor push. In this case, the parent Control Module FB effects the data transfer.

For Sequential Control Module configurations, the input expressions always pull data while the output expressions always push data.

Control Capacity and Performance

This section provides performance related information for the various Experion controllers for convenient reference. The following information is *subject to change without notice* and is superseded by information in applicable Experion product Specification and Technical data documents.

Control network capacity

The network capacity limits for each supervisory control network media available are listed in the following table.

Topic	Network Media and Limits		
	FTE	***CIP Ethernet (See Notes 4 to 8)***	***ControlNet (See Notes 9 to 11)***
Supervisory Networks per Experion Server	1 redundant network serviced by 1 or more redundant or non-redundant Experion servers up to the maximum FTE Nodes allowed per FTE Community. (See Note 1)	1 non-redundant network serviced by 1 redundant or non-redundant Experion server	1 redundant or non-redundant network serviced by 1 redundant or non-redundant Experion Server
ControlNet Connection Limit per PCIC	N/A	N/A	127
Maximum Number of Nodes allowed	330 FTE nodes per FTE community (See Note 2)	12	32
Controllers per Server	Up to **16** Redundant or Non-Redundant in any combination (See Note 3)	Up to **10** Non-Redundant only	Up to **10** Redundant or Non-Redundant in any combination
ACEs per Server	Up to **5**	Not Supported	Up to **2**
Transmission Rate	100 Mbits/sec 10 Mbits/sec for FTEBs	10 Mbits/sec	5 Mbits/sec

	Network Media and Limits		
Media Redundancy	Redundant only	Non-redundant only	Single cable or redundant media operation supported.

Notes:

1. SCADA connected PLCs can still be connected through ControlNet/PCIC when C200s or C300s reside on the FTE supervisory control network.

2. When any FTEB is present in the FTE community the limit is 200 FTE nodes.

3. SCADA PLCs do not count as “Controllers” against this limit on FTE.

4. Non-redundant 10Mbps Ethernet using the TC-CEN011 (obsolete) or TC-CEN021.

5. L3 FTE based servers will NOT support CIP Ethernet supervisory control network.

6. Only qualified for Redundant Servers + 10 non-Redundant Controllers.

7. FIMs are NOT supported on a supervisory CIP Ethernet system in any configuration.

8. ACEs are NOT supported on a supervisory CIP Ethernet system in any configuration.

9. Only 1 PCIC Card per computer supported.

10. Each of these nodes use 1 PCIC connection from the server regardless of where they reside in the system. For example, Supervisory or I/O ControlNet: CPM, IOLIM, FIM. Each ControlNet resident PLC should be limited to 2 connections, if configured as SCADA controllers (this is configured in RSLinx).

11. Default system is configured for 24 nodes (UMAX) but can be updated to 32 only if the actual number of physical nodes required exceeds 24. (Any unconnected but configured MAC addresses consume bandwidth time on the ControlNet.)

C200 configuration options

The following table lists the options related to configuring a C200 Controller.

Module Slot Width	2 slots	
Number of CPM Modules allowed per Chassis	1	
Base Execution Periods Supported	50 ms, 5 ms* (*Requires different firmware personality.)	
Loaded Controller Personality->	***5 ms CEE***	***50 ms CEE***
Redundancy Supported	No	Yes
Remote I/O Supported	No	Yes
I/O Module Execution Period	5 ms	50 ms
Configurable Values for CM/SCM Execution Periods	5, 10, 20, 50, 100, 200 ms.	50, 100, 200, 500, 1000, 2000 ms.
Configurable Values of Peer Update Rates (period). (Defines the period at which data is updated for all 'pull/get' requests for peer data required by all blocks within a CEE.)	10, 20, 50, 100, 200, 500, 1000 ms.	100, 200, 500, 1000 ms. (See Note)

Note:

Supervisory CIP Ethernet LAN clusters only guarantee a Pull/Get Request Rate of 500 milliseconds or greater. Faster Pull/Get rates can be configured, however, data arrival at the configured rate cannot be guaranteed for rates faster than 500 milliseconds.

C200 Redundancy

The following table lists data pertinent to the configuration of C200 Controllers with Series A chassis-based redundant hardware. The C200 Controller must have a 50 ms CEE personality to support redundant operation.

Redundancy compliant devices	CNI, C200 CPM, FTEB, FIM, IOLIM, RM, BEM, LIOM
Non-redundancy compliant devices (cannot be introduced in a redundant chassis pair)	Chassis IOM, SIM, PIM, SST-PFB-CLX, C100 CPM, TC-CEN021 Ethernet Module, DHRIO, DeviceNet Module
Control Processing Switchover Interruption Time	500 ms
Redundancy Module Cable Medium	Fiber Optic Cable
Redundancy Module Cable Lengths	1, 3, 10 meters
Redundancy Module Slot Width	2 slots
Initial Synchronization Time (from Sync Start to Completion)	90 sec
Maximum Elapsed Time Between Commanded Switchover and Completion of Initial Synchronization	150 sec
Maximum Elapsed Time Between Switchover Due to Power Cycle of the Primary and Completion of Initial Synchronization	200 sec

C200 communication performance

The following table lists performance related values for C200 Controller communications. The abbreviation PPS stands for *Average Parameters Per Second*.

Overall Data Access Performance	5 ms CEE	50 ms CEE
Maximum Total Parameter Access Response Rate (Includes all Server Data Requests, Console Station Data Requests, and peer communications including other ACEs, C200s, C300s, SIM-C200s, SIM-ACEs, and FIMs.)	2000 PPS	2000 PPS
Peer-to-Peer Performance	**5 ms CEE**	**50 ms CEE**
Maximum Initiator Pull/Get Request Rate to all target nodes (incoming data)	500 PPS	500 PPS
Maximum Target Response Rate to Pull/Get Requests from all initiator nodes (outgoing data)	500 PPS	500 PPS
Peer-to-Peer Subscription Rates	**ControlNet or FTE**	**ControlNet or FTE**
Maximum C200 peer capacity per update rate choices:	5 @ 10 ms 10 @ 20 ms 25 @ 50 ms 50 @ 100 ms 100 @ 200 ms 250 @ 500 ms 500 @ 1 sec	50 @ 100 ms 100 @ 200 ms 250 @ 500 ms 500 @ 1 sec
	CIP Ethernet	**CIP Ethernet**
	250 @ 500 ms 500 @ 1 sec	250 @ 500 ms 500 @ 1 sec
Push/Store Request Performance	**5 ms CEE**	**50 ms CEE**
Maximum Push/Store Request Rate to all target nodes	50 PPS	50 PPS
Maximum Response Rate to Push/Store Requests from all initiator nodes	50 PPS	50 PPS
Peer-to-Peer Capacity	**5 ms CEE**	**50 ms CEE**

Overall Data Access Performance	5 ms CEE	50 ms CEE
Peer Connection Units* (PCUs) *Remote CEEs that this C200 can initiate a peer connection with Notes: 1. Does not include Exchange Peer-to-Peer. 2. May be limited by other Connection capacities.	5 (Includes total of other ACEs, C200s, C300s, and/or SIM-C200s)	30 (Includes total of other ACEs, SIM-ACEs, C200s, C300s, SIM-C200s, IOLIMs and/or Primary FIMs)
Peer-to-Peer Capacity using Exchange FB	**5 ms CEE**	**50 ms CEE**
Maximum Number of REQUEST blocks per CEE/CPM	32	32
Maximum Number of RESPONSE blocks per CEE/CPM	32	32
Maximum Number of Target Devices for REQUEST blocks per CEE/CPM **Notes:** 1. 1 DHRIO Module only counts as 1 Target Device even when communicating with multiple PLCs on either of the two DH+ networks per DHRIO. 2. See **Error! Reference source not found.** for DHRIO capacities supported. 3. May be limited by Connection capacities.	8	8
Maximum Number of Remote Initiating Devices for RESPONSE block data per CEE/CPM	8	8

C200 processing resources

The following table lists the processing resources for a C200 Controller in Processing Units (PU). The PU represents a platform independent amount of processing resources (time) required to complete a predefined amount of computational (control) work.

For Maximum Cycle Loading, over a cycle (0-39), the "Average CPU Used" (CPUCYCLEAVG) statistic is not to exceed the stated maximums.

Minimum Reserved CPU to be Maintained During Runtime (CPUFREEAVG)	20%	
CEE/CPM Configuration	**PU Maximum**	**Maximum Cycle Loading**
50 ms CEE – Non-Redundant Configuration	3600 PU/sec	60%
50 ms CEE – Redundant Configuration	1600 PU/sec	60%
5 ms CEE – (Non-Redundant only)	2400 PU/sec	40%

C200 Memory Resources

The following table lists the memory units and block capacities for a C200 Controller. The memory unit (MU) represents 1 kilobyte or 1024 bytes.

Maximum Available C200 CEE Memory Resources	4000 MU
Maximum Total Number of CMs, SCMs and IOMs Configurable per C200 CEE	1000
Maximum Number of Component Blocks per CM	40
Maximum Total Number of Steps and Transitions (in all handlers) per SCM	160 (80 Step/Transition pairs)
Maximum number of parallel paths executing in SCM simultaneously	10

C300 configuration options

The following table lists the options related to configuring a C300 Controller.

Control Networks Supported	FTE only (Currently, every C300 must be connected to a Control Firewall)
Device Index Configuration	Between 1 and 510
IOTA Size	6 inches
Number of I/O Links Supported	2
I/O Link Speeds Available	375 Kbaud (PM I/O only) 750 Kbaud (Series C I/O only)
Base Execution Periods Supported	50 ms
Redundancy Supported	Yes
Remote I/O Supported	Yes – PM I/O or Series A I/O
I/O Module Execution Period	50 ms
Configurable Values for CM/SCM Execution Periods	50, 100, 200, 500, 1000, 2000 ms.
Configurable Values of Peer Update Rates (period). (Defines the period at which data is updated for all 'pull/get' requests for peer data required by all blocks within a CEE.)	100, 200, 500, 1000 ms.
C300 Time Source	SNTP
SNTP Update Period	1 minute
GPS Time Support (A GPS connector is available on the IOTA, but it is not supported until a future release.)	No

C300 Redundancy

The following table lists data pertinent to the configuration of C300 Controllers for redundant operation.

C300 Function Block Redundancy Configuration Selection:	"Module is Redundant"
Number of IOTAs used for Redundant C300	2
Redundant Device Index Configuration	Redundant Controller pairs must have Device Index values of **n** and **n + 1**, where **n** is an odd value
Redundancy Cable Medium	Ethernet STP
Redundancy Cable Lengths	36, 48, 60, 84 inches
Control Processing Switchover Interruption Time	500 ms
Initial Synchronization Time (from Sync Start to Completion)	200 seconds
Maximum Elapsed Time Between Commanded Switchover and Completion of Initial Synchronization	300 seconds
Maximum Elapsed Time Between Switchover Due to Power Cycle of the Primary and Completion of Initial Synchronization	300 seconds

C300 communication performance

The following table lists performance related values for C300 Controller communications. The abbreviation PPS stands for *Average Parameters Per Second*.

Overall Data Access Performance	50 ms CEE
Maximum Total Parameter Access Response Rate (Includes all Server Data Requests, Console Station Data Requests, and peer communications including other ACEs, C200s, C300s, SIM-C200s, SIM-ACEs, and FIMs)	2000 PPS
Peer-to-Peer Performance	**50 ms CEE**
Maximum Initiator Pull/Get Request Rate to all target nodes (incoming data)	500 PPS
Maximum Target Response Rate to Pull/Get Requests from all initiator nodes (outgoing data)	500 PPS
Peer-to-Peer Subscription Rates	**FTE**
Maximum C300 peer capacity per update rate choices:	50 @ 100 ms 100 @ 200 ms 250 @ 500 ms 500 @ 1 sec
Push/Store Request Performance	**50 ms CEE**
Maximum Push/Store Request Rate to all target nodes	50 PPS
Maximum Response Rate to Push/Store Requests from all initiator nodes	50 PPS
Peer-to-Peer Capacity	**50 ms CEE**
Peer Connection Units* (PCUs) (*Number of remote CEEs that this C300 can initiate a peer connection with) **Note**: Exchange Peer-Peer does not count against this limit.	30 (Includes total of other ACEs, SIM-ACEs, C200s, C300s, SIM-C200s, and/or Primary FIMs)
Peer-to-Peer Capacity using Exchange FB	**50 ms CEE**
Maximum Number of REQUEST blocks per C300	32
Maximum Number of RESPONSE blocks per C300	32

Overall Data Access Performance	50 ms CEE
Maximum Number of Target Devices for REQUEST blocks per C300 **Note** 1 DHRIO Module only counts as 1 Target Device even when communicating with multiple PLCs on either of the two DH+ networks per DHRIO.	8
Maximum Number of Remote Initiating Devices for RESPONSE block data per C300	8

C300 processing resources

The following table lists the processing resources for a C300 Controller in Processing Units (PU). The PU represents a platform independent amount of processing resources (time) required to complete a predefined amount of computational (control) work.

For Maximum Cycle Loading, over a cycle (0-39), the "Average CPU Used" (CPUCYCLEAVG) statistic is not to exceed the stated maximums.

Minimum Reserved CPU to be Maintained During Runtime (CPUFREEAVG)	20%	
CEE Configuration	**PU Maximum**	**Maximum Cycle Loading**
50 ms CEE – Redundant or Non-Redundant Configuration	2400 PU/sec	60%

C300 Memory Resources

The following table lists the memory units and block capacities for a C300 Controller. The memory unit (MU) represents 1 kilobyte or 1024 bytes.

Maximum Available C300 CEE Memory Resources	16000 MU
Maximum Total Number of CMs, SCMs and IOMs Configurable per C300 CEE	4095
Maximum Number of Component Blocks per CM	40
Maximum Total Number of Steps and Transitions (in all handlers) per SCM	160 (80 Step/Transition pairs)
Maximum number of parallel paths executing in SCM simultaneously	10

ACE configuration options

The following table lists the options related to configuring an ACE Controller.

Base Execution Periods Supported	500 ms
Redundancy Supported	No
Remote I/O Supported	N/A
I/O Module Execution Period	N/A
Configurable Values for CM/SCM Execution Periods	500 ms and 1, 2, 5, 10, 20 sec **Additional options for CMs only**: 30, 40 sec 1, 2, 5,10, 20, 30 min 1,2,4,8,12, 24 hr NOPERIOD
Configurable Values of Peer Update Rates (period) (Defines the period at which data is updated for all 'pull/get' requests for peer data required by all blocks within a CEE.)	500 ms and 1, 2, 5, 10 sec

ACE communication performance

The following table lists performance related values for ACE Controller communications. The abbreviation PPS stands for *Average Parameters Per Second*.

Overall Data Access Performance	**500 ms CEE**
Maximum Total Parameter Access Response Rate (Includes all Server Data Requests, Console Station Data Requests, and peer communications including other ACEs, C200s, C300s, SIM-C200s, SIM-ACEs, and FIMs)	2000 PPS
Peer-to-Peer Performance	**500 ms CEE**
Maximum Initiator Pull/Get Request Rate to all target nodes (incoming data)	500 PPS
Maximum Target Response Rate to Pull/Get Requests from all initiator nodes (outgoing data)	500 PPS
Peer-to-Peer Subscription Rates	**ControlNet or FTE**
Maximum ACE peer capacity per update rate choices	250 @ 500 ms 500 @ 1 sec 1000 @ 2 sec 2500 @ 5 sec 5000 @ 10 sec
Push/Store Request Performance	**500 ms CEE**
Maximum Push/Store Request Rate to all target nodes (See Notes 2, 3)	50 PPS
Maximum Response Rate to Push/Store Requests from all initiator nodes (See Note 3)	50 PPS
Peer-to-Peer Capacity	**500 ms CEE**
Peer Connection Units* (PCUs) (*Number of remote CEEs that this ACE can initiate a peer connection with) **Note**: Does not include Exchange Peer-to-Peer.	30 (Includes total of other ACEs, SIM-ACEs, C200s, C300s, SIM-C200s, and/or Primary FIMs)
OPC Gateway Peer Performance	**500 ms CEE**

Overall Data Access Performance	500 ms CEE
Maximum Initiator OPC Pull/Get Request Rate (to all target OPC Gateways)	5000 PPS
Maximum Initiator OPC Push/Store Request Rate (to all target OPC Gateways)	500 PPS
Notes: 1. CIP Ethernet is a non-redundant network that should not be used for a mission critical control strategy. 2. Currently the SCM Step and the PUSH Block are the only block types that can initiate peer push/store requests for CEE-to-CEE peer communications. 3. For CIP Ethernet Supervisory clusters; the maximum store rate is 500 msec.	

ACE processing resources

The following table lists the processing resources for a ACE Controller in Processing Units (PU). The PU represents a platform independent amount of processing resources (time) required to complete a predefined amount of computational (control) work.

For Maximum Cycle Loading, over a cycle (0-39), the "Average CPU Used" (CPUCYCLEAVG) statistic is not to exceed the stated maximums.

Minimum Reserved CPU to be Maintained During Runtime (CPUFREEAVG) (Note: CPUFREEAVG not supported on ACE; CPU Usage from Windows Task Manager provides the %CPU used.)	40%	
CEE Configuration	**PU Maximum**	**Maximum Cycle Loading**
500 ms CEE – Non-Redundant Configuration	100,000 PU/sec	60%

ACE Memory Resources

The following table lists the memory units and block capacities for a ACE Controller. The memory unit (MU) represents 1 kilobyte or 1024 bytes.

Maximum Available ACE CEE Memory Resources	32000 MU
Maximum Total Number of CMs, SCMs and IOMs Configurable per ACE CEE	4095
Maximum Number of Component Blocks per CM	40
Maximum Total Number of Steps and Transitions (in all handlers) per SCM	160 (80 Step/Transition pairs)
Maximum number of parallel paths executing in SCM simultaneously	10

I/O Capacity and Performance

Every I/O Device presents a “Load” on the C200 or C300 Controller performance by using I/O Network bandwidth, Controller memory, ControlNet or FTE Connections, and CPU resources. In order to not overload the ability for the C200 or C300 to operate correctly, configuration constraints are defined in the following Tables for C200 and C300 I/O. Many of these constraints are measured in “IO UNITS”, which is a generic loading definition used to “normalize” the load among the many different I/O devices and families supported. See the IO Unit load factors section below for the IO UNIT loads assigned to various devices and modules.

C200 IO capacity

The following table lists I/O configuration constraints you should observe to avoid overloading the C200 Controller and impacting its ability to operate correctly.

Function	5 ms CEE	50 ms CEE
Maximum Number of IO UNITS per CEE/CPM **Note 1**: See the IO Unit load factors below for details. **Note 2**: Specific IO Devices may be further limited as described below.	12 (Rail I/O is not supported)	64
Maximum Number of RemoteSeries A I/O Chassis + Remote Mixed Chassis + Rail Adapters (combined) per CEE/CPM **Note 1**: FIM-only chassis or redundant chassis pairs do not count against this limit. **Note 2**: PM I/O chassis do not count against this limit.	0 (Local I/O Only)	8
Maximum Number of IO UNITS that can be assigned per Downlink CNI **Note 1**: See the IO Unit load factors below for details.	0 (Local I/O Only)	24
Maximum Number of Downlink CNIs per C200 Controller Chassis	0 (Local I/O Only)	4
Maximum number of Allen Bradley variable speed drives supported per Controller. This function requires the optional control library TCABDL01 Each drive counts as one IO UNIT in the maximum number of IO per CEE Each drive has to be equipped with the 1203-CN1 communication module (available directly from Allen Bradley) **Note**: The drives have to reside on an isolated I/O network separate from standard Series A chassis or Rail I/O	0 (Local I/O Only)	64 (Limited to 20 per **isolated** downlink CNI)

Function	5 ms CEE	50 ms CEE
Maximum Number of DHRIO Modules per CEE/CPM	2 (Local I/O Only)	2
Serial Interface Related Only		
Maximum Number of Serial Interface Modules per CEE/CPM	1	3
Maximum Number of FTA assemblies per Serial Interface Module **Note**: Each SIM FTA is the equivalent of 4 IO UNITS in the 64 IO per CPM calculation and the 24 IO per CNI calculation. See the IO Unit load factors below for details.	2	2
Fieldbus Interface Related Only		
Maximum number FF non-redundant FIMs per CEE/CPM **Note**: Each FIM counts as 2 IO UNITS in the 64 IO/CPM calculations above. See the IO Unit load factors below for details.	Not Supported	21
Maximum number of Redundant FIMs per CEE/CPM **Note**: Each Primary FIM counts as 2 IO UNITS in the 64 IO/CPM calculations above. Each FIM pair counts as 4 IO Units in the 24 IO Units per downlink CNI calculation. See the IO Unit load factors below for details.	Not Supported	12
Maximum numberof Series C FIMs per C200	Not Supported	Not Supported
PROFIBUS Related Only		
Maximum number of PROFIBUS function block modules per CPM	2	10
Maximum number of PROFIBUS function block modules per downlink CNI (based on update rate, 25ms default)	Not Supported	4 @ 25 ms 2 @ 12.5 ms
DeviceNet Bridge Related Only		
Maximum number ofDeviceNet Bridge modules per CPM	2	32

Function	5 ms CEE	50 ms CEE
Maximum number of DeviceNet Bridge modules per downlink CNI	Not Supported	12

C300 IO capacity

The following table lists I/O configuration constraints you should observe to avoid overloading the C300 Controller and impacting its ability to operate correctly.

Function	50 ms CEE
Maximum Number of IO UNITS per C300 **Note 1**: See the IO Unit load factors below for details. **Note 2**: Specific IO Devices may be further limited as described below.	64
Series C FIM Relate Only	
Maximum number of Redundant or Non-Redundant Series C FIMs per C300 **Note**: Each Primary Series C FIM counts as 4 IO UNITS in the 64 IO/C300 calculations above.	15

IO Unit load factors

The following Table defines the **IO UNIT** load factor for the various I/O Devices the C200 and C300 Controllers support.

I/O Device or Module	IO Units
1 PM IOP Module **Note**: For redundant IOPs, each pair of IOPs counts as 1 IO Unit.	1
1 Series C I/O Module **Note**: For redundant IOMs, each pair of IOMs counts as 1 IO Unit.	1
1 Series A chassis-mounted I/O Module **Note**: except where noted elsewhere.	1
1 Series A or H Rail-mounted I/O Module	1
1 Serial Interface Module (SIM) FTA **Note**: For SIM using both FTAs, this count is doubled to 8 IO Units.	4
1 Pulse Input Module (PIM)	1 (for 64 Unit/CPM limit) 1.5 (for 24 Unit/CNI limit)
Series C Fieldbus Interface Module (FIM4) **Note**: Same IO Unit count whether redundant or non-redundant.	4
1 non-redundant Series A Fieldbus Interface Module (FIM)	2 (for 64 Unit/CPM limit) 3 (for 24 Unit/CNI limit)
1 Redundant Series A Fieldbus Interface Module Pair (Red-FIM)	2 (for 64 Unit/CPM limit) 4 (for 24 Unit/CNI limit)
1 DHRIO Module	1
1 1203-CN1 AB Drive Controller	1
1 SST-PFB-CLX Profibus Module	2 (for 64 Unit/CPM Limit) 6 (for 24 Unit/CNI Limit)
1 DeviceNet Bridge module (DNB)	2 (for 64 Unit/CPM Limit)

I/O Device or Module	IO Units
	2 (for 24 Unit/CNI Limit)
1 Source or Destination Device on Auxiliary Exchange peer-to-peer Ethernet network	1 (for 24 Unit/CNI Limit)

Function Block Execution Schedules

Schedule consideration differences

The CEE execution schedule considerations differ for each of the following Function Block types.

- Control Module and Sequential Control Module FBs(Container FB)
- Component FB
- IOM FB (Self-Standing FB)
- CPM and CEE FBs (Self-Standing FB)

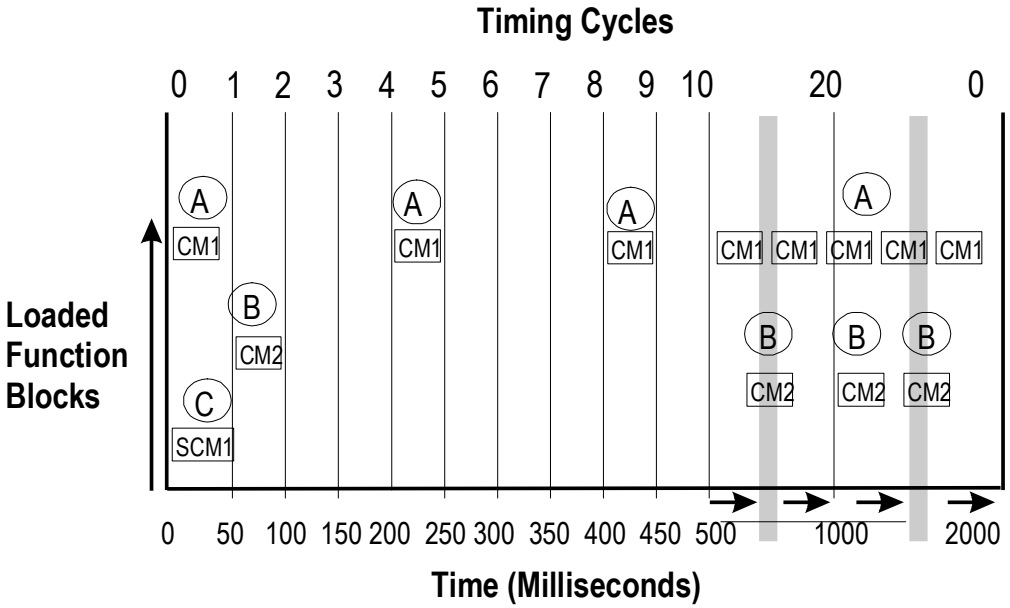
The scheduling considerations for a given FB type are discussed in the following sections.

Control Module and Sequential Control Module FBs schedule

You configure the Execution Period and the Execution Phase values for the Control Module FB and the Sequential Control Module FB through the respective Parameters Configuration form in the Control Builder. These values determine the scan period and the cycles allocated for the block's execution.

You can picture the execution cycle as a set of 40 timing cycles of 50 milliseconds each as shown in the following figure. The configured Execution Period value sets the block's scan period at DEFAULT, 5, 10, 20, 50, 100, 200, 500, 1000, or 2000 milliseconds. (Note that the configuration load will be rejected if the configured scan period is not supported by the 5 ms CEE or 50 ms CEE image loaded in the Controller.) This Period value defines the number of cycles needed for block execution. The configured Execution Phase value identifies the timing cycle in which the execution of the block starts. This lets you stagger the block execution starting times to balance the load processing across the cycles. The DEFAULT value is converted to 1000 milliseconds for a 50 ms CEE or to 200 milliseconds for a 5 ms CEE, when the CEE is loaded to a Controller. The following figure provides a graphical interpretation of how the

processing of some sample Control Module configurations would be scheduled across the timing cycles for a 50 ms CEE.



(A) Control Module CM1 configured for Execution Phase of 0, Execution Period of 200ms, and order in CEE of 10. It executes in timing cycles 0, 4, 8, 12, 16, 20, 24, 28, 32, 36.

(B) Control Module CM2 configured for Execution Phase of 1, Execution Period of 500ms, and order in CEE of 10. It executes in timing cycles 1, 11, 21, and 31.

(C) Sequential Control Module SCM1 configured for Execution Phase of 0, Execution Period of 2000ms, and order in CEE of 20. It executes in timing cycles 0.

Figure 7 Cycle time loading for sample container block configurations for a 50 ms CEE.

For example, a Control Module block with an Execution Period of 200 milliseconds and a Phase of 1 will run in cycles 1, 5, 9, ...,37. Another Control Module block with an Execution Period of 200 milliseconds and a Phase of 2 will run in cycles 2, 6, 10, ...,38.

The entry value range for the Execution Phase is -1 and 0 to 39. However, the system will accept and clamp values outside the appropriate range for a given Period as long as the value is within the overall entry range. Obviously, a block with an Execution Period of 50 milliseconds will always be evenly distributed, since it will run in every cycle. The following table identifies the timing cycles in which a container FB executes for the given combination of Execution Period and Phase values. For now, a Phase value of -1 is changed to 0. In the future, a Phase value of -1 will instruct the CEE to assign Phase values that will distribute the overall-processing load.

ATTENTION

This same scheduling concept applies for the ACE supervisory controller but for periods from 500 milliseconds to 20 seconds.

If Period in milliseconds is. . .		And Phase is . . .	Then, cycles of execution are. . .
5 ms CEE	**50 ms CEE**		
5	50	0	0, 1, 2, 3, . . ., 39
10	100	0	0, 2, 4, . . ., 38
10	100	1	1, 3, 5, . . ., 39
20	200	0	0, 4, 8, . . ., 36
20	200	1	1, 5, 9, . . ., 37
20	200	2	2, 6, 10, . . ., 38
20	200	3	3, 7, 11, . . ., 39
50	500	0	0, 10, 20, 30,
50	500	1	1, 11, 21, 31
50	500	2	2, 12, 22, 32
50	500	3	3, 13, 23, 33
50	500	4	4, 14, 24, 34
50	500	5	5, 15, 25, 35
50	500	6	6, 16, 26, 36

If Period in milliseconds is. . .		And Phase is . . .	Then, cycles of execution are. . .
5 ms CEE	**50 ms CEE**		
50	500	7	7, 17, 27, 37
50	500	8	8, 18, 28, 38
50	500	9	9, 19, 29, 39
100	1000	0	0, 20
100	1000	1	1, 21
100	1000	2	2, 22
100	1000	3	3, 23
100	1000	4	4, 24
100	1000	5	5, 25
100	1000	6	6,26
100	1000	7	7, 27
100	1000	8	8, 28
100	1000	9	9,29
100	1000	10	10, 30
100	1000	11	11, 31
100	1000	12	12, 32
100	1000	13	13, 33
100	1000	14	14, 34
100	1000	15	15, 35
100	1000	16	16, 36
100	1000	17	17, 37
100	1000	18	18, 38
100	1000	19	19, 39

If Period in milliseconds is. . .		And Phase is . . .	Then, cycles of execution are. . .
5 ms CEE	**50 ms CEE**		
200	2000	1	1
200	2000	2	2
200	2000	:	:
200	2000	39	39

For blocks scheduled to start execution in the same cycle, you can configure the Order in CEE parameter value (0 to 32767) on the Parameters Configuration form to stagger the execution order of the container blocks within the cycle. This means the block with the lowest Order in CEE value configured executes first. If both blocks have the same Order in CEE value, the CEE determines the order of execution and maintains it.

CM Component Function Block scheduling

Execution of a Control Module's component function blocks is always subordinate to the containing or parent CM. The component function blocks are executed during the same Period as their containing CM given that the conditions for execution are satisfied. You can configure the execution order of function blocks through the block parameter Execution Order in CM (ORDERINCM).

Function Block Execution Order in CM

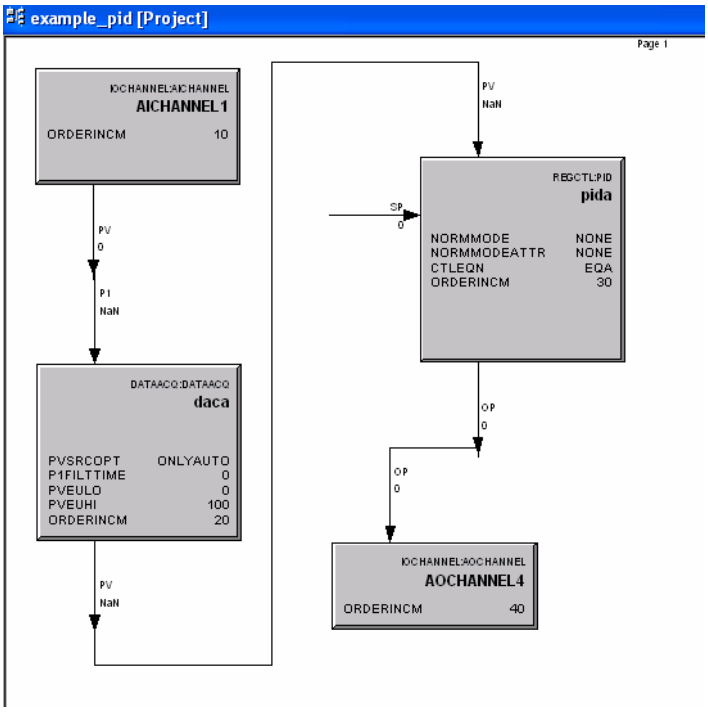
You can configure ORDERINCM parameter value of each function block. The value of ORDERINCM determines the execution order of function blocks within a Control Module (CM). In a given CM, a block with a lower value for ORDERINCM executes before any block with a higher value. The value of ORDERINCM can range from 1 to 32,767 where default values are assigned in multiples of 10, although keeping values as multiples of 10 is not a requirement.

If you do not individually assign ORDERINCM, the default value 10 is set into all blocks within the CM. When this occurs or when the same value is assigned to multiple blocks, Control Builder assigns a random execution order based on the order in which the function blocks were added to the CM, but Control Builder does not change the value of ORDERINCM. If the order of execution is critical, then assign ORDERINCM values to your CM.

You can choose to display the block's Execution Order in CM value through the block's ORDERINCM parameter.

Input type IOCHANNEL FBs do a sample and hold operation when they execute. IOCHANNEL FBs read data from the associated IOM FB and hold that data. If the parent CM has an Execution Period greater than 50 milliseconds, the input IOCHANNEL FBs will hold values static even if the corresponding data is changing at the IOM FB.

A typical PID Loop CM should have component Function Blocks with ORDERINCM as follows:



Connection Execution Order in CM

Connections are processed in relation to a function block's execution order. Inputs are fetched before the block executes and newly calculated values are available only after the block executes.

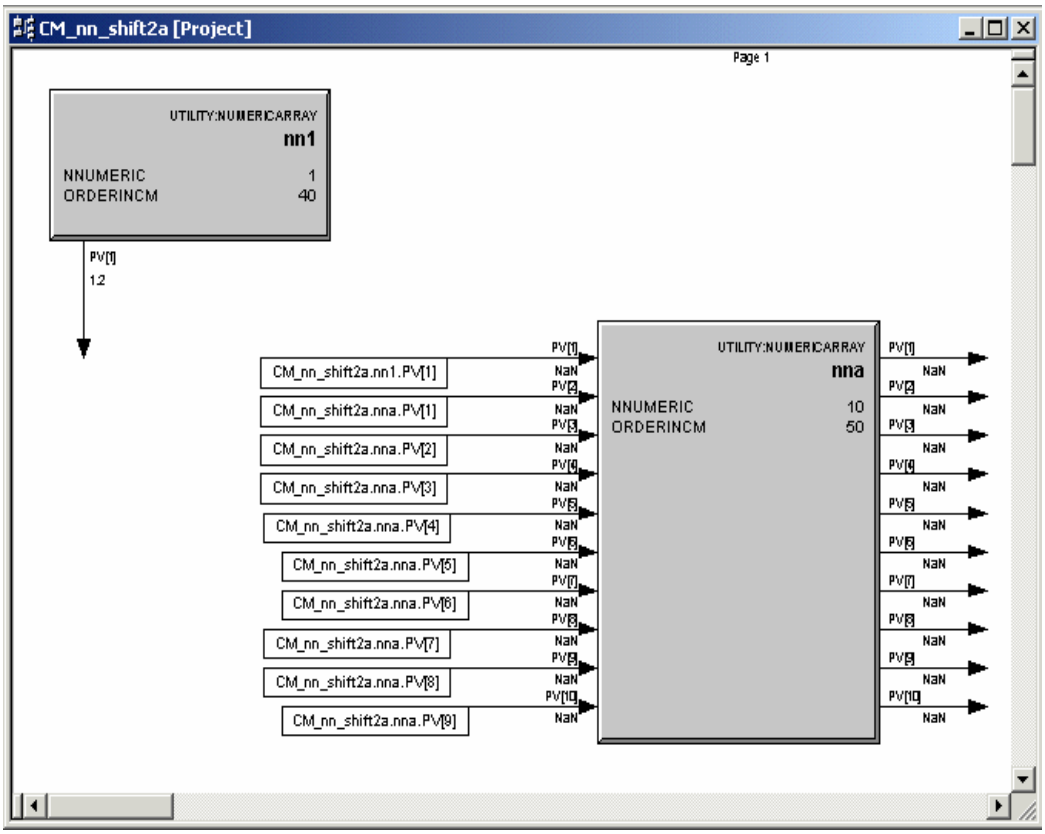
Example of connection cxecution order – shift register

The order in which connections are made is critical in rare cases when blocks without any execution method/behavior are used in a strategy. A Numeric Array used as a Shift

Register demonstrates this rare case. In this example, the numeric contained in Function Block nn1 feeds PV[1] of nna. This connection could have been wired, but for clarity the example shows a named connection. The purpose of the CM is to act as a register and move the value of Function Block nn1 through the ten inputs of function block nna by making each output of Function Block nna a different input to the same block.

If the connections were made in order from top to bottom, the execution order would be from top to bottom and in one execution cycle of the CM, the value of nn1 would be passed from PV[1] through all the PVs to PV[10]. At the end of the cycle, all the values would be equal and the intended register would be ineffective.

To create the register correctly and pass the value from one PV to the next on each cycle, the connections must be built from the bottom of Function Block nna to the top. The first connection created must be to PV[10] in order to ensure it is executed first. The next connection must be made to PV[9], then PV[8], and so on until the final connection is made to PV[1]. Making the connections in this order will ensure that the Control Module acts as a register moving the newest input value through the registers from PV[1] to PV[10].



IOM FB schedule

IOM FBs have a fixed Execution Period of 5 or 50 milliseconds, for 5 ms CEE or 50 ms CEE, and Execution Phase of 0. They execute ten times or once every 50 millisecond cycle at the beginning of each cycle. This means the Order in CEE value does not apply either, since the IOM FBs execute before all other blocks in the cycle.

IOM FBs collect and distribute I/O data as it passes between the Control Processor module and I/O Module devices. They pack and unpack the data in preparation for input and output operations.

CPM, ACE, and CEE FBs schedule

CPM and CEE FBs have a fixed Execution Period of 2000 milliseconds. The Execution Phase is fixed at 19 for the CEE FB and at 39 for the CPM FB. The CEE FB executes

after all other blocks in the cycle. The same is true for the ACE and CEEACE Function Blocks.

The CPM and CEE FBs handle housekeeping functions, which are not directly related to the configuration of control strategies. These functions include maintaining instrumentation statistics, maintaining state data, and reporting diagnostic alarms.

Cycle overruns

Cycle overruns occur when the scheduled processing for a cycle does not finish by the start of the next cycle. Potential causes for overruns include the following.

- Unbalanced loading across the execution cycle.
- Loaded configuration is too large.
- Combination of block and communication processing is too large for a particular 50-millisecond cycle.

The CEE FB responds to cycle overruns as follows.

- Completes execution of all blocks on the current cycle.
- Delays execution of the waiting cycle until the start of the next 50 millisecond time interval.
- Allows communications and housekeeping operations within the CPM to catch up while execution of the waiting cycle is being delayed.

The CEE issues a diagnostic alarm for cycle overruns that occur on a regular basis. The conditions for reporting and clearing this alarm are summarized below based on the controller running the CEE.

If Controller is . . .	CEE reports alarm if . . .	CEE clears alarm if . . .
CPM (50 ms BASEPERIOD)	two consecutive intervals of 2000 milliseconds have at least one cycle overrun.	four consecutive intervals of 2000 milliseconds have no cycle overruns.
ACE (500 ms BASEPERIOD)	two consecutive intervals of 20 seconds have at least one cycle overrun.	four consecutive intervals of 20 seconds have no cycle overruns.

Obviously, you must change a CEE configuration that causes regular overruns by reducing the total load or improving the balance of the load across the timing cycles.

Block Configuration Load Considerations

About load considerations

The following considerations influence CEE operations relative to the loading of block configurations.

- Data Categories
- Container and Self-Standing Blocks Load versus States
- Load Error Messages
- RAM Retention Start Up

These considerations are briefly reviewed in the following sections to provide general background information only.

Data categories

The major data categories found in the CEE FB are summarized below.

Data Category	Description
Live	Data that is derived from the process, such as PV; or is updated frequently as part of the control computation, such as OP and SP. This data can also be considered as the non-structural type.
Tunable Constant	Data that does not change frequently enough to be considered “Live” data. It includes parameters that support states of operation derived from the process or stored by the operator. You can modify values for these parameters on process, but they usually hold fixed values for the life cycle of a control strategy. You can also configure this data at load time. The Execution Phase and Execution Period parameters are examples of this data type, which can be changed independent of block load. This data can also be considered as the non-structural type.

Data Category	Description
Structural	Data that is considered to make up the structure of the block configuration. It includes parameters, which express the containment of a component FB, connection references, or the order of block execution. You can only modify these parameters as part of the block load. The Order in CEE and Order in CM parameters are examples of this data type.

Container and self-standing blocks load versus states

The CEE supports the independent loading of container and self-standing FBs. Component FBs can only be loaded as part of their parent container FB. This means you must load control strategies to the CEE as Control Module or Sequential Control Module configurations.

You can load IOM, CEE, and CPM FBs individually. However, you must load the CPM FB before you load the CEE FB and you must load the CEE FB before you can load other FBs. The status of the following state parameters determines whether or not you can make a FB load at a given time.

- CEEFB. CEESTATE
- CMFB.EXECSTATE
- SCMFB.STATE
- IOMFB.EXECSTATE

The following table summarizes the state parameter status versus FB load permissions.

If status is . . .	Then, you can load . . .
CEESTATE = Idle. Changing the CEESTATE to Idle causes container FBs and their component FBs to cease operation and set their live data to failsafe values. The self-standing FBs (IOM, CEE and CPM) continue to execute unless EXECSTATE is set to Inactive. .	• container blocks regardless of their execution state. – CMFB.EXECSTATE = Inactive or Active – SCMFB.STATE = Inactive or Active • self-standing blocks if their execution state is Inactive. – IOMFB.EXECSTATE = Inactive

Load error messages

Block configurations are loaded to the CEE as a series of parameter value stores. Checks are made on every store to determine if it is "valid" or not. The store of any parameter found to be "invalid" is stopped and a corresponding error message is displayed in Control Builder. The error message is a brief descriptive text string, which includes the name of the "invalid" parameter, so you can quickly determine the cause of the error.

The following three categories of severity apply to CEE error messages.

Severity Category	Description
Warning	The store of an invalid parameter is being accepted with applied qualification. For example, error message "OPHILM – Value Clamped Warning" tells you that the parameter OPHILM is being accepted with clamping applied.
Error	The store of an invalid parameter is being rejected. (Note that the load of other associated parameters continues.) For example, error message "TMOUTMODE – Parameter Invalid" which tells you parameter TMOUTMODE is not operative for this block configuration.
Severe Error	The load of a block is being stopped. (Note that the load of other container blocks will continue, if applicable). For example, error message "CM17 – Maximum Available User Memory Exceeded" tells you that the user memory allocation was exhausted during block load. When a severe error occurs, the block is left in an unusable state within the CEE. If desired, you can reload the "old" version of the block through the Monitor tab in Control Builder.

This error message generation applies for on-line parameter stores as will as block configuration load related stores. For example, an error message can be generated for an on-line parameter store if its value or other conditions are incorrect.

RAM Retention Start Up (RRSU)

The CPM includes a battery that provides backup power for Random Access Memory (RAM) retention during a power interruption. If power is restored while the RAM is still retained through the battery backup, the CEE will startup with the database retained prior to the power interruption. We refer to this as the RAM Retention Start Up (RRSU).

Upon the restoration of power, the CPM runs its startup diagnostics to verify that its RAM was retained during the power interruption. If the CPM detects RAM errors, it starts up with a "null" database. If the CPM detects no RAM errors, it starts up with the database it had prior to the power interruption. In this case, the CEE FB transitions to its Idle state at startup so you can determine if other FB data needs to be changed before you resume control by manually invoking the CEE FB Run state. (Note that the Idle to Run transition triggers output path initialization.)

If RAM errors were detected or you did not load the CEE FB before the power interruption, the CEE FB state will be NotLoaded at startup.

Upon any CPM startup, the CEE reissues all active notifications as part of the Experion notification recovery routine. The CEE also issues "state" transition notifications from the CEE FB and CPM FB that are logged in the event journal to show whether or not an RRSU occurred.

REFERENCE - INTERNAL

Refer to *Notifications Theory* for more information

Memory Usage for CEE on CPM or CEEACE

Typical Requirements

The following table identifies typical processing and memory resource requirements for IOMs, CMs, and SCMs based on Processing Resources per module and Memory Resource usage. The Processing Resources are computed as "Processing Resource Consumption (PU/module execution) divided by Execution Period (sec/module execution)". Please refer to the previous C200, C300, and ACE processing and memory resource sections for more information about the terms PU and MU. The following typical requirements also apply for the ACE supervisory controller.

ATTENTION

The following Control Builder Block Libraries will consume the extra Memory Units listed, when the first block of a given type is loaded to the C200 Controller:

- RAIL_IO_HAZ = 125 MU
- RAIL IO = 125 MU
- EXCHANGE = 70 MU
- PLUSEINPUT = 100 MU
- FBUSIF = 90 MU

Typical Module Types (FB Content in Parenthesis)	**Processing Resource Consumption (Per Module)**		**Memory Resource Usage** (See the ATTENTION note above.)
	50/5 ms CEE Non-Redundant (PU/Module Execution)	***50 ms CEE Redundant (PU/Module Execution)***	***50/5 ms CEE (MU/Mod)***
Typical I/O Module (Average consumption of available IOMs)	0.3	0.19	0.6

Analog Digital Acquisition Module (10 AI, 10 DataAcq FBs)	2.9	3.8	7.4
Small Analog Data Acquisition Module (1 AI, 1 DataAcq FB)	0.47	0.43	1.0
Regulatory Control Module (1 AI, 1 DataAcq, 1 PID, 1 AO, 6 Logic FBs)	2.8	2.8	3.9
Auxiliary Function Module (10 Aux. FBs, such as AuxCalc, Totalizer)	4.2	5.1	13.1
Digital Data Acquisition Module (10 DI, 10 Flag FBs)	1.2	1.2	3.1
Small Digital Data Acquisition Module (1 DI, 1 Flag FBs)	0.22	0.14	0.6
Device Control Module (2 DI, 2 DO, 1 DevCtl, 5 Logic FBs)	1.3	1.3	3.1
Logic Control Module (20 Logic FBs)	1.0	1.0	3.5
	500 ms ACE ***50/5 ms CPM CEE Non-Redundant(PU/Module Execution)***	***50 ms CPM CEE Redundant(PU/Module Execution)***	***500 ms ACE*** ***50/5 ms CPM CEE (MU/Mod)***
Sequence Control Module A (1 each of Main, Hold, Stop and Abort Handlers, 10 Steps with 8 Outputs each, 10 Transitions with 5 Conditions each, 10 Recipe items, 5 History items) SCM has <u>total</u> of 10 Steps and 10 Transitions among the 4 Handlers	2.0	3.0	28.9
Sequence Control Module B (1 Main Handler, no other Handlers, 20 Steps with 4 Outputs each, 20 Transitions with 3 Conditions each, 10 Recipe items, 5 History items) SCM has <u>total</u> of 20 Steps and 20 Transitions	2.0	3.0	35.7

Sequence Control Module with an alias table size of 45 rows by 100 columns (1 each of Main, Hold, Stop and Abort Handlers, 10 Steps with 8 Outputs each, 10 Transitions with 5 Conditions each interspersed in all the handlers, 10 Recipe items, 5 History items) SCM has total of 10 Steps and 10 Transitions among the 4 Handlers	2.0	3.0	128.5
Sequence Control Module with an alias table size of 500 rows by 9 columns (1 each of Main, Hold, Stop and Abort Handlers, 10 Steps with 8 Outputs each, 10 Transitions with 5 Conditions each interspersed in all the handlers, 10 Recipe items, 5 History items) SCM has total of 10 Steps and 10 Transitions among the 4 Handlers	2.0	3.0	124.5
Medium sized CDB Module (CM with 1 medium sized CDB instance) (CDB instance has a total of 15 Custom Data Parameters of which 2 are arrays: 5 32 character STRING; 4 scalar FLOAT64; 4 scalar BOOLEAN; 1 20 element FLOAT64 array; 1 20 element BOOLEAN array.) **Note 1**: Memory usage for first module instance. **Note 2**: Memory usage for every subsequent module instance.	0	0	1.5 (Note 1) 1.0 (Note 2)

NOTES:

These Block Libraries will consume the following extra memory when the first block is loaded to the C200 Controller:

AB_DRIVE_IF	90 MU
AGA	172 MU

DNETIF	106 MU
EXCHANGE	70 MU
FBUSIF (NI Fieldbus Linking Device)	90 MU
HARTIO	105 MU
PBUSIF	244 MU
PULSEINPUT	100 MU
RAIL_IO	125 MU
RAIL_IO_HAZ	125 MU

CPU Utilization for CEE on CPM

CPU load categories

We divide the total CPU load to a CPM into the following three categories.

Category	Description
Core CPU Load	CPU load from core services independent of any load imposed by user configuration.
Dynamic CPU Load	CPU load imposed circumstances, which are unaccounted for by block configuration. This includes such factors as CPU loading From: • notification report and distribution, and • parameter response Server Cache.
Configured CPU Load	CPU load imposed by the user's control strategy configuration. You can estimate this load by determining the number of "typical module types" as described in the previous *Typical Requirements* section.

CPU utilization limits and estimates

This is based on the previous maximum loading cycle ratings listed in the table in the C200 processing resources section. In summary, 60 percent of the CPU can be used for the configuration load with a 50 ms CEE or 40 percent with a 5 ms CEE.

CEE Standard Control Blocks

The following table lists the size of the standard function blocks for the CEE controller (C200, C300, ACE).

Block Name	Size (Bytes)	Block Name	Size (Bytes)
2OO3	68	NUMERIC	56
ACEFB	196	NUMERICARRAY	40
AGA3OM_92	460	OFFDELAY	56
AGA7TM_96	164	ONDELAY	56

Block Name	Size (Bytes)	Block Name	Size (Bytes)
AGA8DL_94	720	OR	60
AGA8GS_94	348	OVRDSEL	1852
AGA9UM_98	164	PID	1896
AND	60	PIDFF	2012
AUTOMAN	1272	PID-PL	2912
AUXCALC	2452	POSPROP	2020
CEEACEFB	15388	PULSE	56
CHECKBAD	52	PULSECOUNT	328
CONTROLMODULE	292	PULSELENGTH	308
DATAACQ	688	PUSH	204
DEADTIME	776	QOR	68
DELAY	56	RAMPSOAK	1596
DEVCTL	1092	RATIOBIAS	1408
EQ	84	REGCALC	6112
FANOUT	2696	REMCAS	1492
FLAG	108	ROL	48
FLAGARRAY	36	ROR	48
FTRIG	44	RS	44
GE	84	RTRIG	44
GENLIN	296	SCM	2196
GT	84	SEL	48
HANDLER	92	SELREAL	68
LE	84	SHL	48
LEADLAG	232	SHR	48

Block Name	Size (Bytes)	Block Name	Size (Bytes)
LIMIT	144	SR	44
LT	84	STEP	392
MAX	112	SWITCH	2520
MAXPULSE	56	TEXTARRAY	40
MESSAGE	348	TIMER	64
MIN	112	TOTALIZER	240
MINPULSE	56	TRANSITION	236
MUX	64	TRIG	44
MUXREAL	116	TYPECONVERT	148
MVOTE	72	UCNOUT	332
NAND	60	WATCHDOG	56
NE	84	XOR	60
NOON	108		
NOR	60		
NOT	44		

Relative References

This section describes what relative references are, how they are displayed, and what Control Builder functions support them as well as some examples of how they may be used.

About Relative References

Beginning in Experion R300.1, you can use a relative reference as a shorthand method for specifying a parameter reference in the following scenarios, since all relative parameter references are within the scope of a tagname.

- A parameter reference on any block could refer to another parameter on the same block.
- A parameter reference on a container block could refer to a parameter of a contained basic block.
- A parameter reference on a basic block could refer to a parameter on its container.
- A parameter reference on a basic block could refer to a parameter on another basic block within the same container as the basic block.

Relative References structure rules

The following table lists the syntax for relative references. As noted in the table, a special character ($) is used when the parameter name desired for the reference resides on the container block, but the reference owner also defines the same parameter name. Without special handling, the entry would resolve the reference to cm.myblock.param, where the user desired a reference to cm.param. In this case, a special character specifies that the container block is desired for the reference.

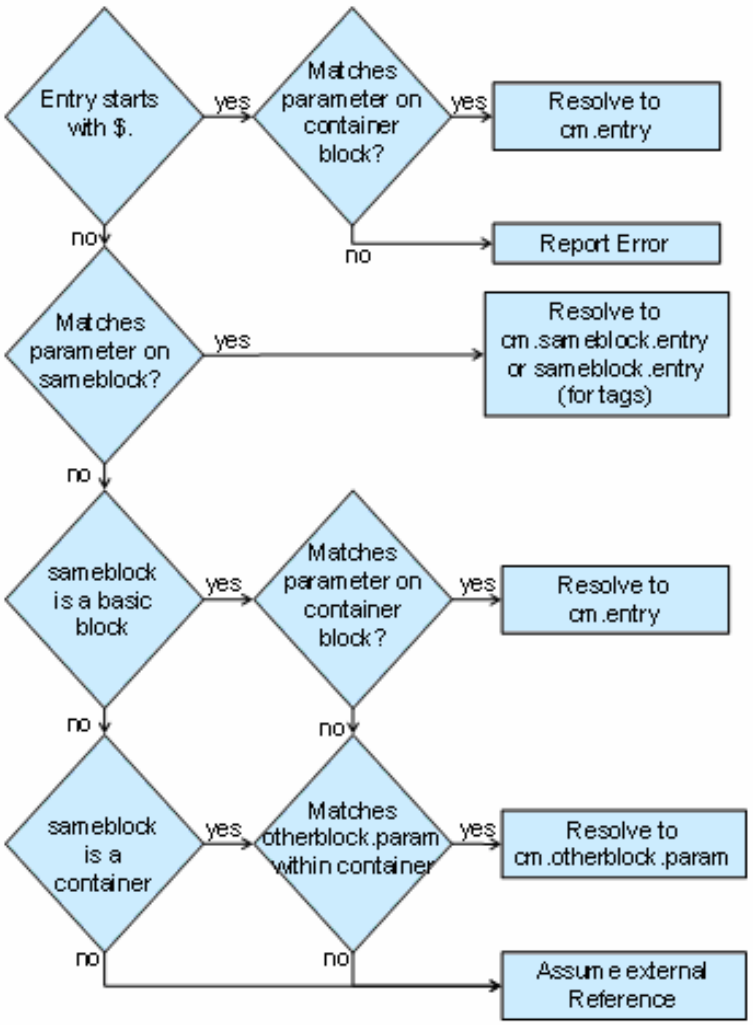
Reference Is Owned By	**Parameter to be Referenced Is Owned By**				
	Same Block	***Tagged Basic Block***	***Container with same param name as a basic block***	***Same Container***	***Other basic block within container***
Basic Block	param	$.tag.param or tag.param	$. param	param or $.param	block. param or $.block.param
Tagged Basic Block (Hart enabled channel blocks)[1]	param	FULLNAME	$.param	Param or $.param	block. param or $.block.param
Tagged Block	param or $.param	FULLNAME	FULLNAME	FULLNAME	FULLNAME
Container Block	param or $.param	$.tag.param or tag.param	FULLNAME	param or $.param	block. param or $.block.param

Notes

1. Tagged basic blocks are not in the same name space as the container that they reside in, but short name entry is supported. A relative parameter reference could **not** be made from a parameter reference on a tagged block to another tagged block within the same container.

How relative reference matching function works

The following flowchart diagram provides an overview of how the relative reference matching function searches for a match on the user entry.



Display options for relative references

You can select whether relative references are to be displayed in **Full Name** or **Short Name** format through the *Relative Reference Display Option* on the **General** tab of the **System Preferences** dialog in Control Builder. Refer to Setting system preferences in the Control Building Guide for more information. This option only applies to how relative references are displayed and the default is **Full Name**. Relative references may be entered in shorthand format, regardless of the option selected

If you select the **Full Name** option and enter a reference as block.param, the reference will be displayed as tagname.block.param.

If you select the **Short Name** option and enter a reference as tagname.block.param that is within the scope of the tagname, the reference will be displayed in a short name format. When a basic block references a parameter on its container block, a special character ($) precedes the parameter name. The following table outlines how references are displayed when the **Short Name** option is selected.

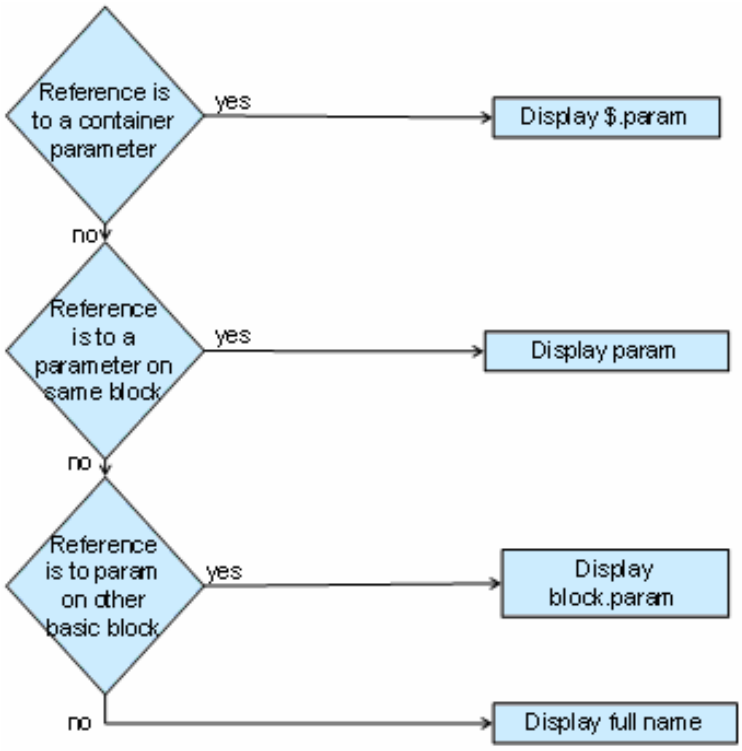
Reference Is Owned By	**Parameter to be Referenced Is Owned By**				
	Same Block	***Tagged Basic Block***	***Container with same param name that a basic block has***	***Same Container***	***Other basic block within container***
Basic Block	param	FULLNAME	$. param	$. param	block. param
Tagged Basic Block (Hart enabled channel blocks)[1]	param	FULLNAME	$. param	$. param	block. param
Tagged Block	param	FULLNAME	FULLNAME	FULLNAME	FULLNAME
Container Block	param	FULLNAME	FULLNAME	param	block. param

Notes

1. Tagged basic blocks are not in the same name space as the container that they reside in, so relative references are only displayed in short name format if the reference is on the same block as the tagged basic block.

How short name display option works

The following flowchart diagram provides an overview of how parameter references are displayed when you select the **Short Name** as the *Relative Reference Display Option*.



Control Builder functions supporting relative references

The following Control Builder functions support the entry of parameters as relative references.

- Expressions
- Parameter Connectors
- Parameters of type TC_PARAMID. Examples of these parameters are the history, trend and group configuration parameters associated with tags and key parameters used in the SCM/RCM table view.

Control Builder blocks supporting expressions

The following blocks support the use of expressions and the entry of parameters as relative references.

- AUXCALC
- ENHAUXCALC
- ENHREGCALC
- REGCALC
- STEP
- TRANSITION

Example of TRANSITION block using relative references

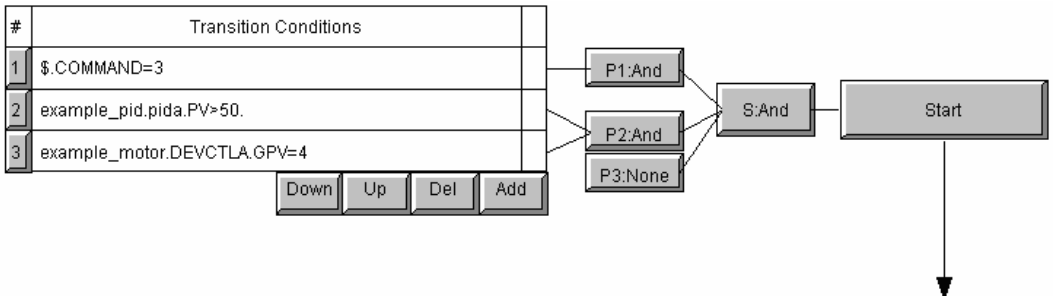
You can enter an expression using relative references for parameter references within the scope of the containing Control Module (CM), Sequential Control Module (SCM), or Recipe Control Module (RCM). The length of the expression will be validated against the string as it will be displayed based on the *Relative Reference Display Option* selection and not the length of the entry.

The following are valid entries for the same equation in a TRANSITION block within the Example_SCM.

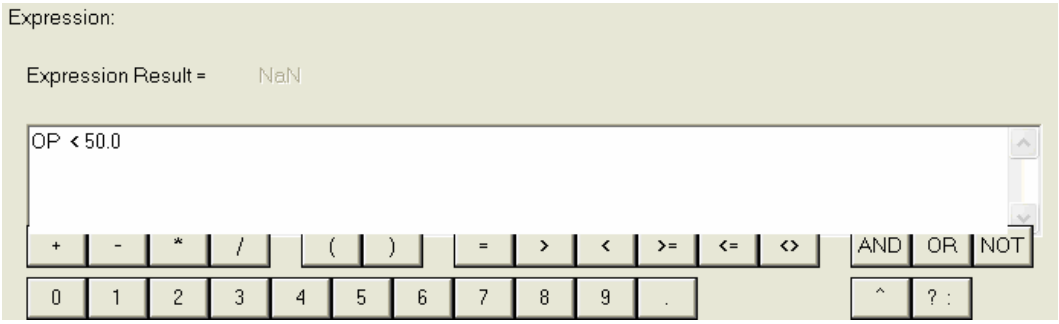
- example_scm.COMMAND=3
- COMMAND=3

The following example illustrations show how expressions would appear with **Short Name** as the selected *Relative Reference Display Option*.

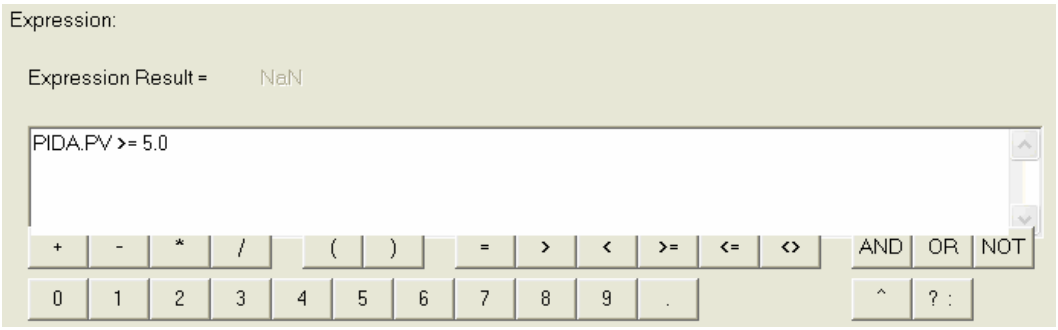
A container reference from a chart



A reference to a parameter on the same block from a form



A reference to a parameter on a different block in the same CM



Expression display interaction

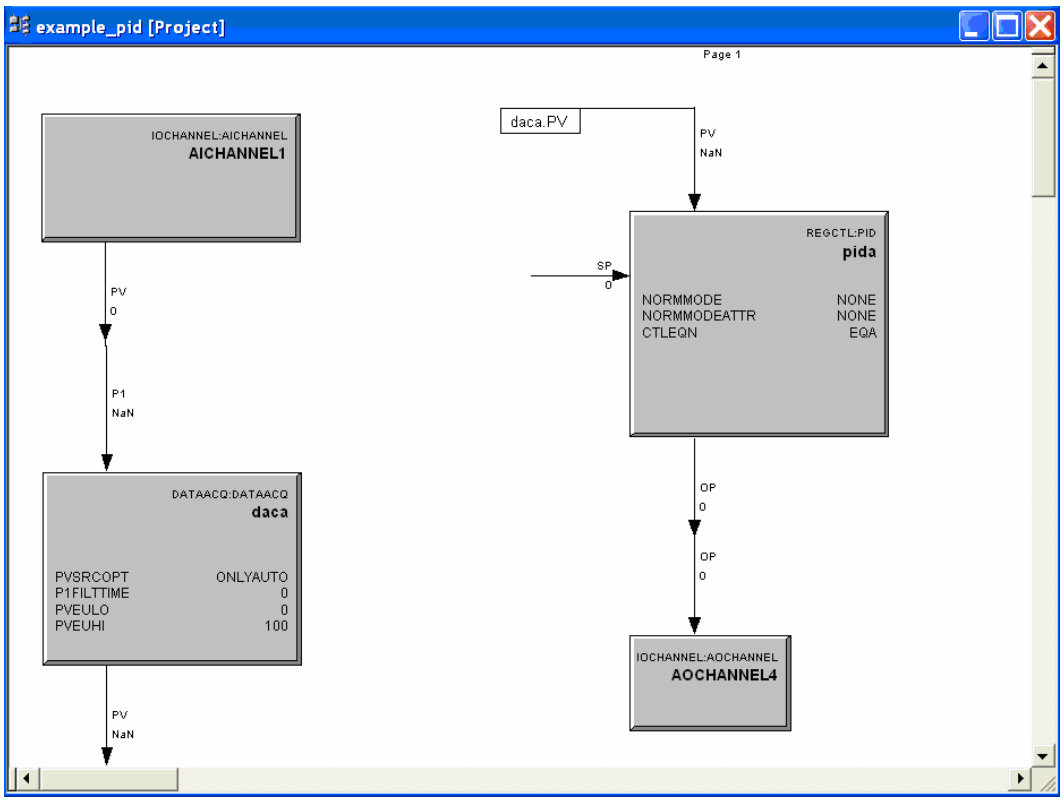
The format of an expression parameter that is read from Server (Station displays), displayed in a Control Builder Report, or included in an Import/Export (IXP) file depends on whether the *Relative Reference Display Option* selection on the Control Builder **System Preferences** dialog is **Full Name** or **Short Name**.

Example of parameter connector using relative references

You can enter a parameter connector as a relative reference. The container owns the parameter connection. The display format for a parameter connector in the Control Builder chart depends on whether the *Relative Reference Display Option* selection on the Control Builder **System Preferences** dialog is **Full Name** or **Short Name**. All cross references will be displayed as full name regardless of the *Relative Reference Display Option* selection.

The following example illustration shows how parameter connectors would appear with **Short Name** as the *Relative Reference Display Option* selection.

A reference to a parameter on a different block in the same CM



Parameter connector display interaction

The format of a parameter connector displayed in a Control Builder Report or included in an Import/Export (IXP) file will always be full name.

Example of other parameters using relative references

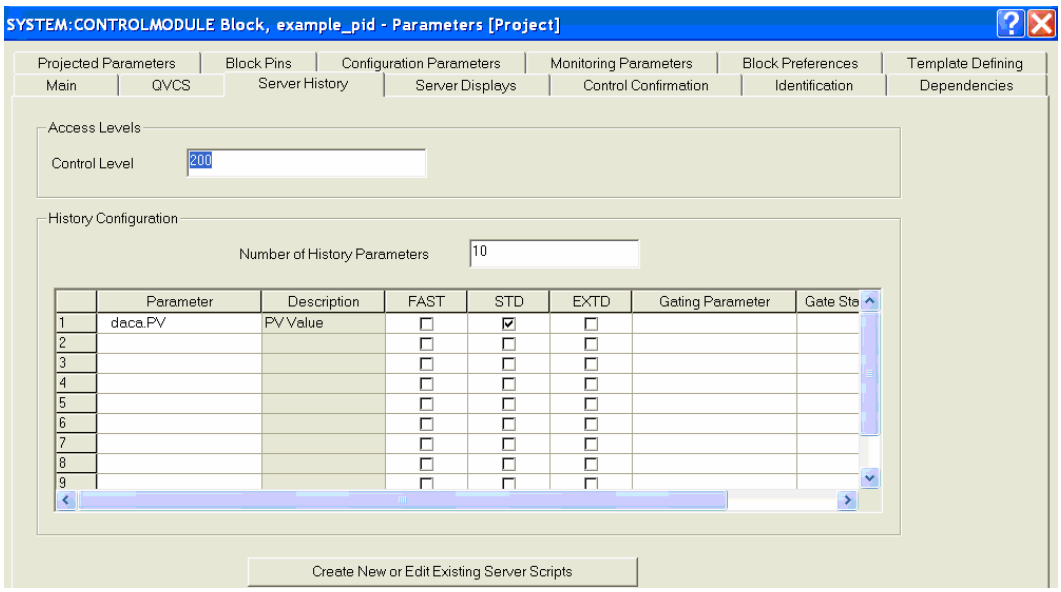
You can enter the following parameters as a relative reference. The display format for a parameter in the Control Builder chart depends on whether the *Relative Reference Display Option* selection on the Control Builder **System Preferences** dialog is **Full Name** or **Short Name**.

- History, trend and group parameters that are defined on TAGGED blocks:
  - GROUP.PARAM
  - HIST.GATEPARAM
  - HIST.PARAM
  - TREND.PARAM
- Parameters on the STEP block related to procedural operations:
  - CURRVALREF
  - MONTASKREF
  - OP[].CURRVALREF
  - OP[].ENTRYVALREF
  - OP[].MONTASKREF
  - OP[].TARGETVALREF'
- Parameters on the SCM and RCM associated with the alias table:
  - ALIASREF

In addition to these parameter references, Control Builder allows the origin parameter of projected parameters to be entered as a relative reference

The following example illustration shows how other parameters would appear if **Short Name** is the *Relative Reference Display Option* selection.

A reference to a parameter on a basic block of the CM



Other parameters display interaction

The format of other parameters read from Server (Station displays) will always be full name.

The format of other parameters displayed in a Control Builder Report, or included in an Import/Export (IXP) file depends on whether the *Relative Reference Display Option* selection on the Control Builder **System Preferences** dialog is **Full Name** or **Short Name**.

Bulk Builder and Bulk Editor support

The Bulk Builder and Bulk Editor do **not** support entry or display of shorthand relative references.

Peer-to-Peer Functionality

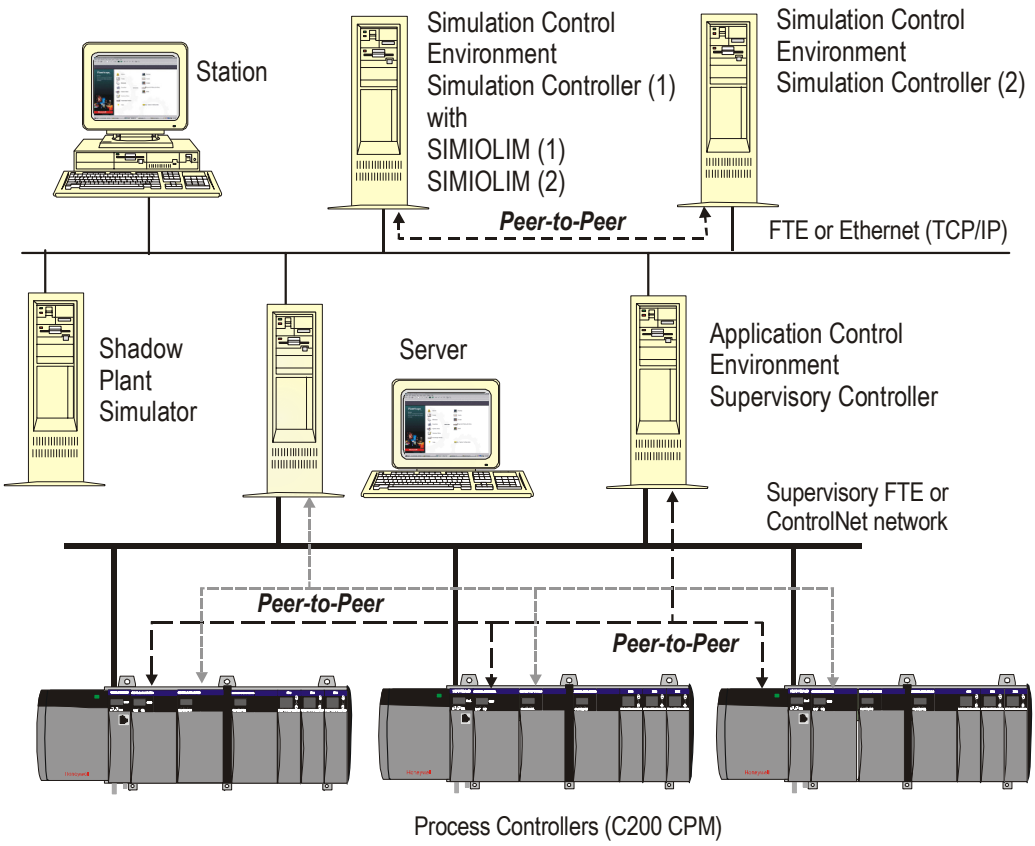
Basic Peer-to-Peer Design Concepts

About Peer-to-Peer

You can implement peer-to-peer communications among CEEs in multiple C200 Control Processor Modules (CPMs) networked to the same Server.

You can visualize the peer-to-peer function as a set of connections between two CPMs. This allows function blocks in different Control Execution Environments (CEEs) to share data through user configured parameter connections. The Control Data Access (CDA) services assume the responsibility for maintaining all connections.

You can implement peer-to-peer communications among CEEs in multiple Application Control Environment (ACE) supervisory controllers that belong to the same system Server over the Fault Tolerant Ehternet (FTE) or Ethernet network. The ACE supervisory controller can peer-to-peer with CEEs in multiple C200 CPMs through an optional direct connection to the ControlNet supervisory network. The following figure illustrates the possible peer-to-peer communication topologies. It shows peer-to-peer between ACE supervisory controllers over Ehternet, between ACE supervisory controller and C200 Process Controllers over ControlNet supervisory network, and between C200 Process Controllers over the supervisory network.



Data flow models

The peer-to-peer design uses both Publish/Subscribe and Request/Response data flow models.

The Publish/Subscribe model establishes a publication contract between two CEEs. The one CEE (Subscriber) asks the other CEE (Publisher) to periodically publish the data value. You can configure the Subscriber's subscription rate as 10, 20, 50, 100, 200, 500 or 1000 milliseconds through the associated CPM block configuration form. (Note that the applicable subscription rates vary for the 5 ms CEE or 50 ms CEE image loaded in the Controller.) The Publisher periodically publishes data on a report by exception basis. If a Publisher supports two Subscribers with different publication rates, it maintains independent publication rates for all of its clients. The Publisher continues to publish data until the Subscriber cancels the contract.

The Request/Response model involves a one-time request for get/store data. A get request reads the value of a data item and a store request writes the value of a data item. Once the responder fulfills the request and the requester receives the response, the operation is complete.

Peer environments and subscription periods

You can identify different peer environments and configure individual subscription periods and store reponse time values through the applicable CEE configuration form in Control Builder. This includes other ACE supervisory controllers, C200 Process Controllers, and external servers, such as OPC servers. The configuration form also specifies default supscription period and store response time values to be used for all peer environments without individually specified settings.

The peer subscription period parameter defines the update period used for cylic "get" requests for peer references. The peer store response time expritation time parameter defines the expriation time used in waiting for "store" responses. In addition to system wide default values, the values for specific CEE peers can be adjusted by users with an Engineer access level or higher in the Monitor mode of Control Builder.

Software Architecture for CPM/CEE

The graphic in the following figure gives a high level view of Experion's communication architecture. It shows how CDA Publish/Subscribe and Request/Response data access methods provide network communications to/from blocks associated with CEEs loaded in CPMs.

This graphic is only intended to give you a general idea of how CDA manages the flow of data values for CEEs, CB Monitoring , and Station displays within the Experion system. The dynamic cache acts like a communications filter to reduce multiple requests for the same data value down to one, which in turn, results in the sending of only one publication request for the item.

The important thing to remember is that peer-to-peer communications are layered upon CDA, which uses connection-oriented communications.

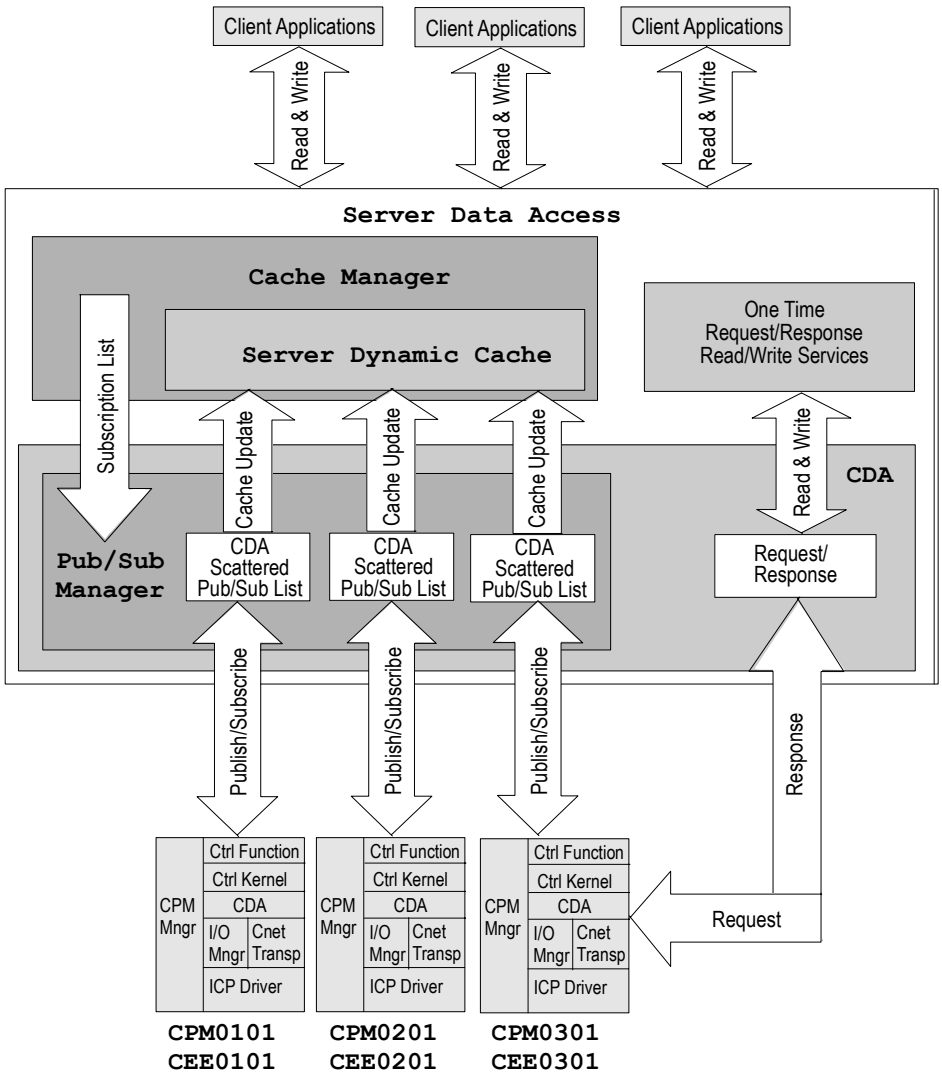


Figure 8 Peer-to-peer is layered upon CDA connection-oriented communications.

A word about ACE/CEE architecture

The overall architecture for the ACE supervisory controller is similar to the C200 Process Controller. It featues separate CEE and Control Data Access-superviory platform (CDA-sp) processes that communicate through shared memory and Windows 2000 events.

The ACE/CEE executes on a personal computer running a Windows 2000 Server operating system. The CEE consists of two subsystems: Control Kernel (CK) and Control Functions (CF). The CK provides services for function block execution and data transfer for controlling a process. The CF is a collection of function (algorithm) blocks for controlling a process.

Since the ACE/CEE runs on a time-sharing operating system, the timed event for function block execution may be delayed or interrupted by other tasks. The ACE/CEE function block includes statistical parameters for calculating timed event for function block execution.

Cyclic "get" requests are forwarded to the CDA during function block connection configuration load. This results in the allocation and addition of a request in the shared memory between CDA-sp and ACE/CEE where peer data is imaged. CDA-sp collects peer data and copies it from communication buffers into shared memory. At run time, a function block converts memory data into a reference to intimately access peer data.

Implications for Control Builder Configuration

ATTENTION

Please note that the format used for the CPM and CEE tag names in this section is for example purposes only and does not reflect the current default naming conventions used in Control Builder.

Multiple CPMs and ACEs

You can now configure multiple CPMs for C200 Process Controllers networked to the same system Server as well as up to two ACE supervisory controllers. This also means that there are multiple CEEs - one associated with each CPM and one associated with each ACE supervisory controller.

As before, each CPM and CEE will have unique tag names based on the Media Access Control (MAC) address for the controller and the numbered slot position of the CPM in the controller. This means you can now have multiple CEEs for I/O Modules, Control Modules, and Sequential Control Modules assignment.

Like the CPM/CEE, the ACE and CEE function blocks will have unique tag names assigned as default by the system or configured by the user. The ACE supervisory controller node is identified on the Ethernet network by its host name and Internet Protocol (IP) address that must be configured by the user. If the ACE supervisory controller includes a network interface card for direct connection to a ControlNet supervisory network, users must also specify the MAC address assigned to the interface card.

CEE execution and subscription rates

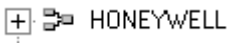
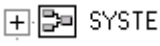
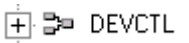
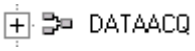
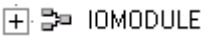
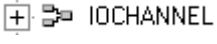
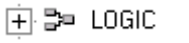
The configurable Base Execution Period (BASEPERIOD) defines the marco cycle based on the 40 phases used for CEE scheduling. See the previous Function Block Execution Schedules section for more information. The BASEPERIOD is user configurable as 5ms or 50ms for a CPM/CEE and fixed at 500ms for ACE/CEE.

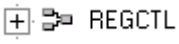
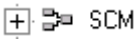
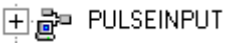
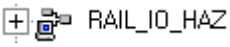
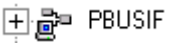
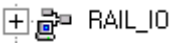
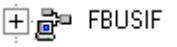
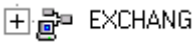
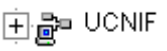
You can configure default Peer Subscription Period (SUBSCPERIOD) and Peer Store Response Expriation Time (STRRESP) values to be used for all peer environments without a specific configured value.

You can configure specific Peer Subscription Period (PEERSUBSCPER[ ]) and Peer Store Response Expriation Time (PEERSTRRESP[ ]) values for specific environments (PEERENV[ ]) identified through configuration.

Function block support

Since ACE supervisory controller operation is based upon the Control Processor Module (CPM) design, ACE supports many of the same function blocks as the CPM. The ACE supervisory controller does not support any of the existing function blocks associated with I/O communictation interface. The CPM does not support the new UCNIF block. The following table lists the Function Block Libraries in Control Builder and identifies which control environment supports them.

If Function Block Is From This Library in Control Builder . . .	Then, It Can Be Used With This Control Environment . . .
 (Typical Fieldbus Device vendor library.)	CPM/CEE
	ACE/CEE CPM/CEE (Restrictions apply - loaded strategies must contain supported blocks only.)
	ACE/CEE CPM/CEE
	ACE/CEE CPM/CEE
	ACE/CEE CPM/CEE
	CPM/CEE
	CPM/CEE
	CPM/CEE
	ACE/CEE CPM/CEE

If Function Block Is From This Library in Control Builder . . .	Then, It Can Be Used With This Control Environment . . .
	ACE/CEE CPM/CEE
	ACE/CEE CPM/CEE
	ACE/CEE CPM/CEE
	CPM/CEE
	CPM/CEE
	CPM/CEE
	CPM/CEE
	CPM/CEE
	CPM/CEE
	ACE/CEE

Peer-to-peer connections and DEF and REF blocks

You make peer-to-peer connections by specifying the full tag names for parameters in parameter connectors, calculation expressions, and input and output condition expressions for TRANSITIONs and STEPs in SCMs. This identifies the source of the parameter value from a compatible block parameter in another CM or SCM, which is assigned to a different CEE.

In peer-to-peer applications, the function block that defines a parameter value that can be referenced by another function block is referred to as the DEF (definition) block. The function block that reads a parameter value from a DEF block is referred to as the REF (reference) block. The DEF versus REF block concept is useful in quickly determining whether CEEs contain publishers (DEF blocks) or subscribers (REF block) of data. However, it is possible for a function block to be both a DEF and a REF block. Refer to the Configuration example section next for details.

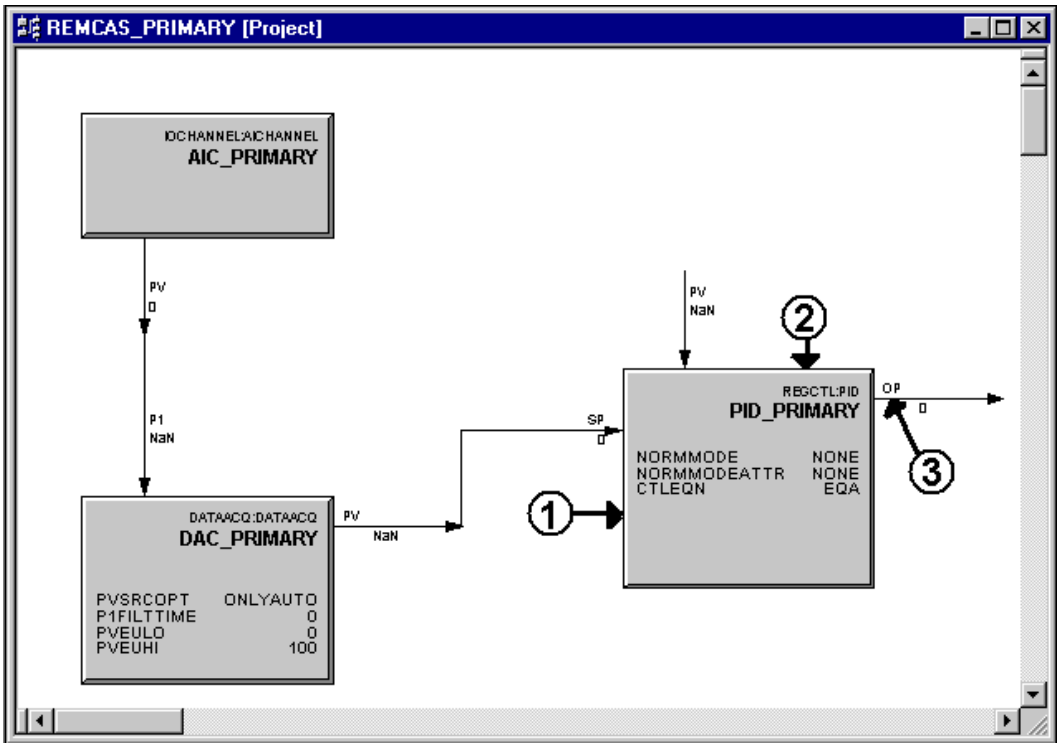
Also, the storing of an output from a STEP block in an SCM to another function block is handled as a one-time request/response operation. The STEP block is the REF block and the function block receiving the output store is the DEF block. A one-time request is issued each time the STEP is executed.

Peer-to-peer configuration example

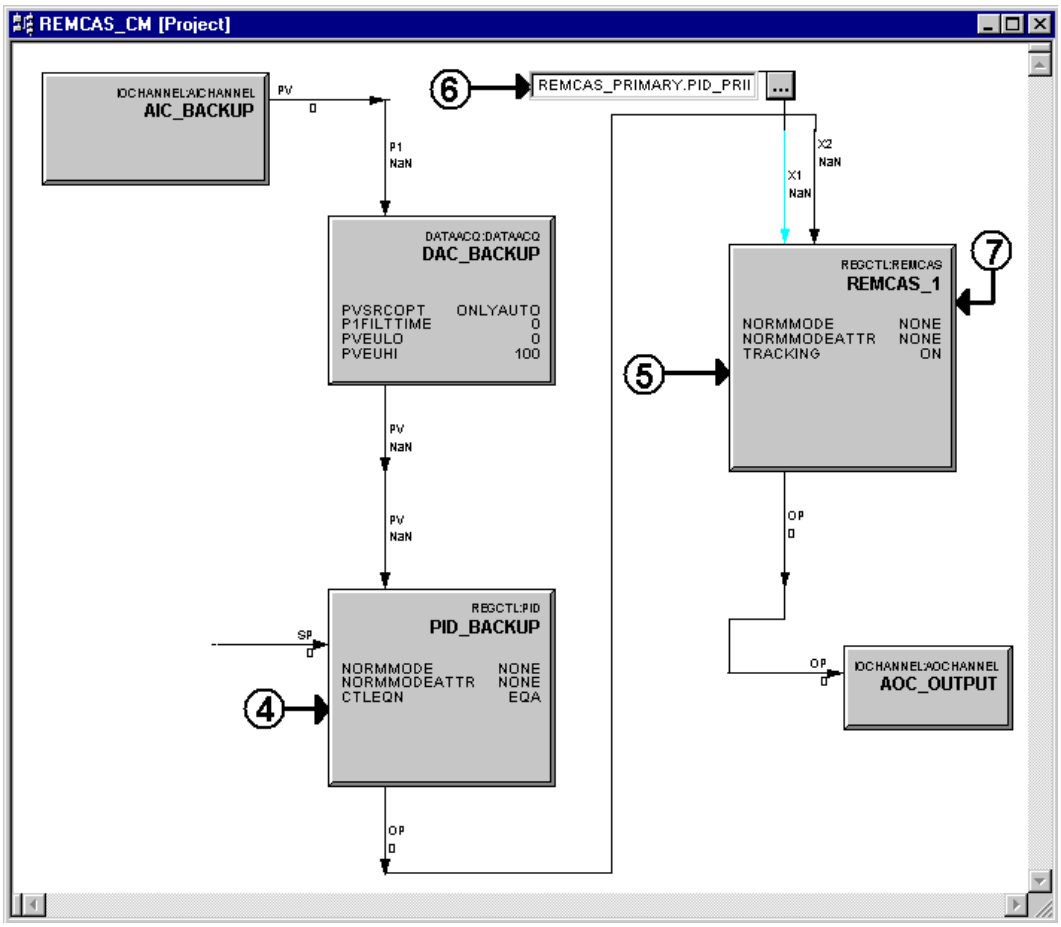
Views A and B in the following figure and the companion callout description table show a sample configuration that uses a REMCAS block to form a cascade control loop with remote and local primary loops.

For this example, assume that this application includes two C200 Process Controllers with CPM blocks configured as CPM0101 and CPM0301, and their associated CEE blocks CEE0101 and CEE0301, respectively. The remote primary loop is configured as a CM named REMCAS_PRIMARY and is assigned to CEE0101. The local primary loop is configured as a CM named REMCAS_CM and is assigned to CEE0301.

In this example, the PID_PRIMARY and REMCAS_1 blocks have both DEF and REF relationships for peer-to-peer communications.



View A – Remote Primary Control Loop



View B – Local (Backup) Primary Control Loop

Figure 9 Example of DEF and REF block functions in CB configuration using REMCAS block.

The following table includes descriptions of the callouts in the figure above.

Callout	Description
1	The PID_PRIMARY block represents the remote primary control loop for a cascade loop using the REMCAS block. It is contained in a CM named REMCAS_PRIMARY, which is assigned to CEE0101.
2	CB builds implicit/hidden connections for BACKCAL data. In this case, CB makes a peer-to-peer connection to the X1BACKCALOUT parameter from the REMCAS_1 block contained in another CM named REMCAS_CM, which is assigned to CEE0301. This means the PID_PRIMARY block is considered a REF type block and the REMCAS_1 block is considered a DEF type block for peer-to-peer communications of the BACKCAL data.
3	The output parameter OP is to be used as the input value for the X1 parameter for the REMCAS_1 block contained in another CM named REMCAS_CM. This means the PID_PRIMARY block is considered a DEF type block and the REMCAS_1 block is considered a REF type block for peer-to-peer communications of the control variable data.
4	The PID_BACKUP block represents the local or backup primary control loop for a cascade loop using the REMCAS block. It is contained in the REMCAS_CM CM with the REMCAS_1 block. If there is a problem with the remote primary loop, the REMCAS_1 block switches its input to the backup primary loop.
5	The REMCAS block operates like any cascaded secondary loop except that it can switch between two different primaries.
6	A parameter connector is used to form a peer-to-peer connection to the OP parameter from the PID_PRIMARY block contained in another CM named REMCAS_PRIMARY, which is assigned to CEE0101. In this case, the full tag name for the parameter is REMCAS_PRIMARY.PID_PRIMARY.OP. This means the PID_PRIMARY block is considered a DEF type block and the REMCAS_1 block is considered a REF type block for peer-to-peer communications of the control variable data.

Callout	Description
7	CB builds implicit/hidden connections for BACKCAL data. In this case, the BACKCAL parameter X1BACKCALOUT is to be used as the input value for the BACKCALCIN parameter for the PID_PRIMARY block contained in another CM named REMCAS_PRIMARY. This means the REMCAS_1 block is considered a DEF type block and the PID_PRIMARY block is considered a REF type block for peer-to-peer communications of the BACKCAL data.

Peer-to-peer configuration guidelines

Observe the following guidelines when configuring peer-to-peer functions in CB.

- Assign CMs with I/O Channel blocks and their associated IOM blocks to the same CEE.
- Assign CM containing ultimate secondary control block and CM containing its output channel and IOM blocks to the same CEE.
- Assign CM containing data acquisition or regulatory control block and CM containing its input channel and IOM blocks to the same CEE.
- Do not assign IOM blocks to multiple CEEs. Use peer-to-peer communications to share input channel values among CMs in multiple CEEs.
- If you delete and unassign a loaded CM containing a DEF block from one CEE and reassign and load it to another, you **must** reload all CMs containing REF blocks with peer-to-peer relationships to the DEF block in the reassigned CM. Otherwise, the REF blocks will continue to receive published failsafe data from the former CEE. You can use the loaded version of the REF block CM for the reloading operation.
- We recommend that you assign CMs containing DEF and REF blocks to their respective CEEs before you make the named parameter connections to the REF blocks.
- You must load CMs containing REF and DEF blocks to CEEs associated with CPMs in C200 Process Controllers or ACE supervisory controllers networked to the same Experion server. This means controllers must be in the same management domain.
- You must assign CMs with DEF blocks to their CEEs before you load CMs with REF block relationships. Otherwise, the peer-to-peer connections are invalid.
- You can add CMs with REF block relationships to CEEs online at any time.

- Control Modules can contain any number of parameter references as well as any mix of intra-CEE and inter-CEE (peer-to-peer) references.
- A function block may have both DEF and REF relationships as shown in the configuration example in the previous figure. This means a REF block can also be a DEF block to another DEF block.

Time Support in Experion System

Experion uses Coordinated Universal Time (UTC) which is raw time with no time zone and no daylight savings time adjustments. UTC is stored internally in Experion and is converted to local time when shown to the user using time zone and daylight savings time adjustments.

Whenever time is stored by Server for history and events, Server also stores bias (minutes offset from UTC including daylight adjustment) so that local time can be reconstructed at any time in the future regardless of any adjustments in daylight savings rules in future years, which can be expressed as

- Local Time = UTC + bias

Date and Time Synchronization

Experion Time Source varies depending on Windows Domain. Servers that are part of Windows Domain use the Primary Domain Controller (PDC) as the time source. Without Windows Domain, a primary Server is designated as the time source and the backup/DSA Servers identify the time source machine and are synchronized by the Windows Time service.

Time source time is set via Windows Date/Time applet and all Windows machines in a cluster (Servers, Stations) are synchronized via Windows mechanisms.

Execution Environment (EE) devices (C200, ACE, FIM, IOLIM) are synchronized by Experion Control Data Access (CDA) Server with the one exception of Custom Algorithm Block (CAB) on Application Control Environment (ACE). Every minute, CDA obtains current time from Server and point to point delivers it to primary Execution Environment (EE) devices on the network (ControlNet or Fault Tolerant Ethernet). Devices update their system time and use local free-running counters to compute the current system time between updates.

CAB on ACE functionality is not synchronized by CDA; it relies on the underlying .NET and Windows OS services of the ACE platform. Time sensed within CAB programs is based on Windows OS synchronization derived from the same Experion Server that supplies time synchronization to CDA.

C300 and Series C Fieldbus Interface Module Time Synchronization

The C300 Controller and Series C Fieldbus Interface Module require high-quality time from a Network Time Protocol (NTP) Server. The quality of time determines how tightly events from different devices can be correlated, especially with regard to digital Sequence-of-Events (SOE). You must specify the address for the NTP source through the System Preference dialog that is accessible through the Tools menu in Control Builder. The procedure is described in the Supplementary Installation Tasks Guide section <IBS><S1284>"Setting up time synchronization"</IBS>

Note that the C300 and Series C Fieldbus Interface Module will start with time sources noted below, but they require an NTP time source for optimal operation.

Time Source	Comments
CDA Time	Time is provided by the first Windows Server/Station that makes a data connection. C300 control resumes as configured by user for warm or cold initialization, based on time controller was powered down. A notification and soft failure indication will flag degraded time source.
No time	If C300 control is configured to resume, it will do so with cold initialization but it will not run until it receives time from the specified source.

Details of NTP and its subset SNTP (Simple Network Time Protocol) are available by referencing RFC 1361 on the Web.

BOOTP Mechanism and Multiple Clusters

The C300 and FIM4 receive their IP Addresses and the IP Addresses of the configured NTP Servers through the Experion R300 BOOTP Service. BOOTP is a broadcast protocol; all BOOTP servers on a given subnet respond to all requests. In a mixed release system on one subnet, even when using different FTE Multicast Addresses, only the newest (R300) BOOTP can provide all of the necessary information for R300 and prior-release devices.

ATTENTION

BOOTP services on prior-release Servers must be disabled because they do not provide enough information.

ATTENTION

All FTE System Preferences for all clusters must be configured with the same information.

Once a device has obtained its IP Address and NTP Server IP Address(es), it retains them until its Device Index is changed or firmware is reloaded.

Time Configuration Considerations

Experion supports three time data types in CEE Blocks which encode values of time:

- TIME encodes absolute time
- TIMEOFDAY encodes time of day
- DELTATIME encodes time differences

Time Data Types in Control Execution Environment (CEE) Blocks

CURTIME is of type TIME and always shows the current "wall time" for the system where the platform blocks reside. The CURTIME parameter is defined on platform entity blocks such as ACE, CPM, and FIM.

Parameters of time data types most frequently appear in custom block types. The two categories of custom block types supported in Experion are Custom Algorithm Blocks (CAB) and Custom Data Blocks (CDB).

Time Data Types in User Algorithms and Displays

Time data types are often used in conjunction with some form of user written algorithm. In the case of the Custom Algorithm Block (CAB), the user defines a block type which encapsulates both a data definition and an algorithm definition. If the data definition incorporates a Custom Defined Parameter (CDP) of time data type then the CDP is read or written by the CAB program as needed.

In the case of a CDB type incorporating time CDPs, the CDB itself never reads or writes the CDPs since the CDB is a data repository and contains no algorithm. However, a CDB's time CDP may be read or written by user defined expressions contained in the algorithm of other blocks. User defined expressions associated with instances of SCM, AUXCALC, or REGCALC blocks may access time CDPs as appropriate.

TIME

Use of TIME CDPs requires attention to issues of time zone. In particular, users who create algorithms to manipulate CDPs of type TIME must understand the Experion policy with respect to time zones and daylight savings time. This policy is stated below.

- When TIME parameters are transported from an EE to displays they are always assumed to be in UTC. User algorithms must publish TIME parameters in UTC and not in local time.
- When HMI receives TIME parameters from an EE and displays them, they are converted from UTC to local time. The conversion takes into account whether daylight savings time is currently in effect.
- When HMI receives user input of TIME parameters it is assumed that they have been supplied as values consistent with the local time zone and with any seasonal daylight savings adjustment currently in effect. They are converted to UTC before transport to any EE or user algorithm.

TIMEOFDAY

As with TIME CDPs, use of TIMEOFDAY CDPs requires attention to issues of time zone. In particular, users who create algorithms to manipulate CDPs of type TIMEOFDAY must understand the Experion policy with respect to time zones and daylight savings time. This policy is stated below.

- When TIMEOFDAY parameters are transported from an EE to HMI they are always assumed to be in UTC modulo 24 hours. User algorithms must publish TIMEOFDAY parameters in UTC modulo 24 hours and not in local time.
- When HMI receives TIMEOFDAY parameters from an EE and displays them, they are converted from UTC modulo 24 hours to local time. The conversion takes into account whether daylight savings time is currently in effect.
- When HMI receives user input of TIMEOFDAY parameters it is assumed that they have been supplied as values consistent with the local time zone and with any seasonal daylight savings adjustment currently in effect. They are converted to UTC modulo 24 hours before transport to any EE or user algorithm.

DELTATIME

Whenever a user algorithm manipulates CDPs of type DELTATIME, be that algorithm implemented as a CAB program or as an SCM or calculator expression, behavior of the datum is simple and intuitive with respect to HMI.

- Time zone is irrelevant to the display of DELTATIME parameters.
- DELTATIMEs are always displayed as time intervals in a format such as “DD HH:MM:SS”.

C200, Application Control Environment (ACE) Time Parameters

Parameter	Descripton
Current Time (CURTIME)	Format is MM/DD/YY HH:MM:SS Server local time communicated to the controller by Control Data Access (CDA) from Server. Not affected by TIMEZONE and DAYLIGHTTIME CURTIME is stored as UTC within the CEE and converted to local time when displayed at HMI.
TIMEZONE default = 0.0	Conversion value for controller’s “local time”; must be configured once for each individual platform Range of Values: -12.0 to 13.0 Consult a World Map of Timezones for the appropriate value for your time zone. Allows support for unusual case of controller physically residing in different time zone than the Server, and interpretation of YEAR, MONTH, etc. parameters based on controller’s time zone
DAYLIGHTTIME default = OFF	Conversion value for controller’s “local time”; must be configured once for each individual platform Possible Values: ON = Daylight savings time is active OFF = Daylight savings time is inactive Allows support for unusual case of controller physically residing in different time zone than the Server, and interpretation of YEAR, MONTH, etc. parameters based on controller’s time zone
YEAR	“local time” for use in control strategies, including expressions IF TIMEZONE and DAYLIGHTTIME are set to their default values, YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, WEEKDAY integer values will always be in raw UTC.
MONTH	Same as above
DAY	Same as above

Parameter	Descripton
HOUR	Same as above
MINUTE	Same as above
SECOND	Same as above
WEEKDAY	Same as above

Examples of TIMEZONE and DAYLIGHTTIME configuration:

In Windows Date/time applet, Server Time Zone is set to GMT -05:00 Eastern Time (US & Canada)	Settings	'local HOUR' for use
Current time (CURTIME) = 11/02/2003 08:54:00	TIMEZONE = 0 DAYLIGHTTIME = OFF	HOUR will be 13
Current time (CURTIME) = 06/12/2003 13:54:00	TIMEZONE = -5 DAYLIGHTTIME = ON	HOUR will be 17

Custom Algorithm Block (CAB) on ACE Time Functions

Custom Algorithm Blocks (CAB) access .NET functions on the local Application Control Environment (ACE) machine and use .NET Time functions supported on the DateTime structure as follows:

Function	Description
UtcNow	current UTC time For CAB VB.NET programs, DateTime.Now is derived from Experion synchronized time. Time zone and daylight selection are configured through the Windows date / time applet on the primary server which has been designated as the system time source. This configuration need only be done on one server within the system
Now	current local time DateTime Now is based on Windows OS time services and is independent of parameters TIMEZONE and DAYLIGHTTIME.
ToUniversalTime	converts local to UTC time

Function	Description
ToLocalTime	converts UTC to local time

It is appropriate for some applications to make use of the UTC time references, DateTime.UtcNow in the case of CAB VB.NET and UTCNOW in the case of CEE expressions. For other applications, it is appropriate to use the local time references, DateTime.Now in VB.NET and NOW in CEE expressions.

Application Control Environment (ACE) machine time synchronization

The ACE node time is synchronized two ways:

a) By Control Data Access (CDA) for non-CAB CEE functions, like controllers.

b) By Windows mechanism for Windows time used by the Custom Algorithm Block (CAB).

Application Control Environment (ACE) Implications

1. Within CAB or within non-CAB Control Execution Environments, time precision is consistent down to the execution period of the blocks.

2. Between CAB and non-CAB CEE, the potential skew is the same or slightly worse than the Windows mechanism. For example, Windows Simple Network Time Protocol (SNTP) should not be expected to provide better than +/- 2 seconds synchronization between machines.

Examples of Time Usage in CAB and SCM

Example - Custom Algorithm Block (CAB) writes Time CDP

The CAB VB.NET program uses variable CurrentTime to capture the current time in the local time zone, uses variable NextCalcHour to hold the next hour when shift calculations are due, and publishes the last time of calculation in a Custom Defined Parameter (CDP) called LASTRUNTIME. CurrentTime is used as a local time variable internally. LASTRUNTIME must be published as UTC.

To create a CAB to do shift calculations at the end of every 8 hour shift:

Step	Action
1	Create a CDP called LASTRUNTIME.

Step	Action

2	Code the CAB's VB.NET program as
	`Dim NextCalcHour As Integer`
	`Dim CurrentTime As DateTime`
	`Dim CurrentHour As Integer`
	`CurrentTime = Now`
	`CurrentHour = CurrentTime.TimeOfDay.Hours`
	`If CurrentHour = NextCalcHour Then`
	`NextCalcHour = (CurrentHour + 8) Mod 24`
	`LASTRUNTIME.Value = CurrentTime.ToUniversalTime`
	`DoShiftCalculations()`
	`End If`

Example - Custom Algorithm Block (CAB) reads Time CDP

The CAB VB.NET program uses a Custom Defined Parameter (CDP) called CAPTURETIME specified by the operator in local time.

To create a CAB to capture data at an operator specified time.

Step	Action

1	Code the CAB's VB.NET program with:
	`If DateTime.UtcNow = CAPTURETIME.Value`
	`DoCapture();`
	`End If`
	(The statement `If DateTime.Now = CAPTURETIME.Value` would **not** work because by the time CAB receives the value of CAPTURETIME it is in UTC, not local time.)

Example – Sequence Control Module (SCM) waits for a specified Time

To create an SCM transition that designed to wait until a time specified by an operator or supervisor:

Step	Action
1	Setup Custom Data Block (CDB) located within a Control Module. Name the

Step	Action
	Custom Data Block CDB2.
2	Locate CDB2 in a control module called CM2. The SCM will use CDB2 within CM2 as a data repository.
3	CM2.CDB2 has a parameter called STARTTIME of type TIME. The operator will specify STARTTIME.
4	The SCM's transition waits until the specified time before advancing. The transition condition expression reads: UTCNOW = CM2.CDB2.STARTTIME UTCNOW is the valid time parameter to use in this instance. UTC (Universal Coordinated Time) is stored raw time with no time zone and no daylight savings time adjustments. UTC is converted to local time when shown to the user and then includes time zone and daylight savings time adjustments. (The transition condition expression NOW = CM2.CDB2.STARTTIME would **not** work even though the operator enters values for STARTTIME according to the local time zone.)

Example – Sequence Control Module (SCM) records Time of process command

To record the time at which an important process command was issued:

Step	Action
1	Create a Custom Data Block (CDB) called CDB1 in a Control Module called CM1.
2	Create a parameter of type TIME in CM1.CDB1 called CMDISSUETIME.
3	To capture the time the command was issued and save it in a proper format, SCM1.STEP1 uses the output expression: CM1.CDB1.CMDISSUETIME := UTCNOW UTCNOW is the valid time parameter to use in this instance. UTC (Universal Coordinated Time) is stored raw time with no time zone and no daylight savings time adjustments. UTC is converted to local time when shown to the user and then includes time zone and daylight savings time adjustments. (The output expression CM1.CDB1.CMDISSUETIME := NOW is incorrect because the display would **not** use the value of CMDISSUETIME according to the local time zone.)

Cold and Warm Restart Functionality

Overview

CEE Initial Startup

The initial startup occurs when a CEE is started by initial load of its parent controller / device block. In this situation, the CEE starts out empty of algorithm blocks and is basically inert. Cold and Warm Restart functionality is irrelevant to the start of a CEE empty of algorithm blocks.

CEE Restarts

The two types of restarts are:

- Cold Restart initiated on transition of CEESTATE from idle to run occurs
  - After the CEE FB has just been created and the initial configuration is loaded.
  - After restore of a checkpoint so that the CEE database is neither null nor newly loaded from CB. In this case, a careful policy must be followed as to which data is preserved and which is reinitialized.

The policy followed by cold restart assumes that much of the state data is stale. Any live data which can be derived directly from the process is wiped out and much of the operational data which was captured in the saved checkpoint is reinitialized. All configuration data is saved across cold restart.

A typical example of operational data which is lost is the mode of a primary RegCtl block connected to a direct analog output. In cases of CEE-to-CEE regulatory cascades the mode is shed to manual upon cold restart.

- Warm Restart is initiated on transition of CEESTATE from idle to run and its most important application occurs after restore of a checkpoint. Warm Restart differs from Cold Restart in the policy it applies in choosing data to preserve and data to reinitialize. Warm restart preserves all data that cold restart preserves but preserves additional operational data as well.

A typical example of operational data which is preserved is the mode of a primary RegCtl block connected to a direct analog output. In cases of CEE-to-CEE regulatory cascades this mode is retained across warm restart.

CEE can be commanded to go to run from idle through a warm start transition even when there has been no checkpoint restore. For example, if a user loads an entire CEE through configuration load he would normally go to run with a cold start command but

nothing prevents him from commanding warm start. In regulatory cascades MODE values loaded as configuration data will be retained across warm restart even if the cascade terminates on a direct analog output.

CEE Restart Behaviors

All CEE blocks perform some kind of initialization in response to a restart condition. Most blocks do the same kind of initialization regardless of whether the restart is warm or cold. For blocks that vary behavior based on restart the categories of functionality involved are:

- Mode shed of regulatory cascades
  Depending on the restart type and on the nature of the cascade configuration, cascades will either shed to manual or hold last mode. The behavior that occurs depends on whether the type of cascade is CEE-to-Legacy or CEE-to-CEE, upon the type of block that is in the role of secondary, upon the type of node undergoing restart, and other factors.

- Resumption of sequences
  Depending on the particular sequence application it may be appropriate for an SCM to remain inert awaiting operator command before it resumes execution. On the other, in some special applications, it may be appropriate that an SCM restart automatically. SCM supports a range of restart behaviors subject to configuration options.

- Resumption of totalizers
  Within CEE totalizers always remain stopped after cold restart but can automatically resume totalizing after warm restart. The warm restart behavior is configurable and is coupled with the selection of bad input behavior.

- Restart signals
  In some cases it may be necessary for an application engineer to cause his control strategy to go through a special initialization in response to restart. To enable this, parameters are supported which allow SCM expressions or CM component blocks to detect that a restart has just occurred and what kind of restart it was.

Planning

Invariant or Variant Restart Behaviors

CEE blocks can be broadly divided into two categories with respect to how the behave on restart.

- Blocks which perform the same start up initialization regardless of the type of restart in progress
- Blocks which perform different start up initializations depending on whether a cold or warm restart occurs

The following broad principles generally apply to all blocks regardless of whether they change behavior in response to cold or warm restart.

- PV handling on warm and cold restart
  PV handling does not change in response to the selection of warm or cold restart unless otherwise noted (PITOTALIZER and TOTALIZER are the only exceptions). Upon restore of a checkpoint PVs are initialized to their fail safe values (NaN in the case of floating point PVs). For PV algorithms which involve historical computation state (filters, delays, etc.) all history is re-initialized.
- Alarm handling on warm and cold restart
  When a node other than that which issued the alarm is restarted, alarm timestamps are preserved. However, when a CEE or IOLIM is restarted, any alarms reported by the nodes themselves acquire the time stamp that goes with the time of restart. The timestamp of original report is not retained. This applies regardless of whether the restart is warm or cold.

Blocks with Invariant Restart Behavior

While behavior varies according to block type most CEE blocks execute the same kind of restart and initialization regardless of whether the start up is warm or cold. These behaviors conform to a few general principles.

- **Data storage parameters and data storage blocks: hold on all restarts**

  There is a category of parameters whose sole purpose is to serve as data storage. For these parameters there is no built-in algorithm implemented by the block underneath, neither in the sense that the parameter value is read and processed by the algorithm, nor in the sense that the value is written by the algorithm. Rather, the parameter serves to hold data which can be read or written by applications created by the end user. In some cases, there are blocks (simple blocks) whose sole purpose is to

establish this kind of parameter. In other cases, parameters like this are one kind of parameter implemented along with many others within a block.

Examples of data storage parameters and their blocks are as follows: PV and PVFL parameters on FLAG block; PV parameter on NUMERICARRAY block; RECTARGET[ ] parameter on the SCM module; all parameters of Custom Data Block (CDB) types created by end users.

For data of this kind the restart behavior is simply that the value is held. This means that if the value was restored by checkpoint restore then it is held across restart.

- **Logic gates: hold on all restarts**

  Logic gate inputs and outputs are held for any type of CEE restart. This is consistent with the Inactivation and Activation behavior of an individual Control Module. The inputs and outputs are held to allow certain configuration changes without affecting other strategies that may reference gate outputs. Note that this implies a logic gate contained in a Control Module that is not executing has its output held at the value calculated at the last time it did execute, or at an initialized value after load. Careful consideration is required when the outputs are referenced outside the containing Control Module. For restarts of an entire CEE, the normal method of proper ordering of a set of related Control Modules' execution through PERIOD, PHASE, and ORDERINCEE should be used. For agents in peer nodes referencing a gate output, or in critical applications where Inactivation of only one of several related Control Modules needs to be tolerated, consideration should be given to implementating the inter-Control Module references using CHECKBOOL pairs.

  Logic gate inputs and outputs are not treated as persistent state variables in CEE. Thus, they are not restored as part of a configuration load nor saved as part of checkpoint data. On any execution, including the first following a restart, connected gate inputs are fetched just before the output is calculated. This insures that logic gate outputs contained within an executing control module are a function of the most currently fetched inputs.

  In the case of a load or checkpoint restore, all outputs are set to False, and inputs are set to default values appropriate for the block type. This policy insures that output values will be at a defined, known state, after load even if the value is read before first block execution. Thus, if an output value happens to be read by an off node agent before first execution, it will always return False. Logic strategies must be designed so that they are tolerant of the appearance of False valued outputs before first execution of the source block. Again, for critical applications, it may be

necessary to utilize CHECKBOOL pairs for inter-Control Module references to fully specify behavior under all restart scenarios.

The policies governing logic gate behavior for restart and load allow strategies to be configured that allow detection of certain types of restarts. For example if a FLAG block's PVFL is TRUE at the time as checkpoint save, or is configured as TRUE before load, it will be restored as TRUE. If connected to an RTRIG's IN, the RTRIG would generate a pulse only on the first restart following a restore. In all cases this type of implementation is not recommended. If detection of various restart types is required, it should only be implemented using the STARTSIGNAL block.

- **Non-Logic Blocks that maintain state/history: initialize to current inputs from infinite past**

  A non-logic block which accumulates state or history in response to inputs initializes at first execution in such a manner that current output is consistent with current inputs as if those inputs had been applied infinitely in the past.

  Examples of blocks which follow this rule are DATAACQ (filter state is initialized to current input upon all restarts), LEADLAG (difference equation state is set to zero or to match input upon all restarts), DEADTIME (delay queue is initialized to current input upon all restarts), DELAYTIME (shift register is initialized to current input upon all restarts).

Not all block types with invariant restart behavior are covered by the principles listed above. For these, restart behavior is case specific and is prescribed by their functional definition. A few examples are noted in the table below.

Special Case Blocks with Invariant Restarts

#	Library	Block Type	Summary Description
1	DEVCTL	DEVCTL	Initialization behavior is the same regardless of restart type. Supports different initialization behaviors based on input mapping, output mapping and other configuration.
2	REGCTL	RAMPSOAK	Different from other RegCtl blocks in that behavior is the same for all restarts. Behavior is: set MODE to manual; set CV to NaN; reset internal timers; set current profile identifier to first; set current segment identifier to first; set

#	Library	Block Type	Summary Description
			RESETTIMER to On; apply STARTOP[ ] value at next Man to Auto transition. Behavior is always safe because it forces the ramping action to stop and await intervention by the operator.
3	UTILITY	TIMER	Continues timing in response to any restart. Output parameters which indicate state are held. The PV parameter, which gives time accumulated since start, continues to increment from its previously retained value.
4	<Custom Libraries Created By End Users>	Custom Algorithm Block Types (CAB Types)	CEE and CAB infrastructure leave values of all custom parameters unchanged across all restarts. User-implemented CAB algorithms may detect restarts and either modify or leave unchanged the values of custom parameters.

Blocks with Variant Restart Behavior

Within CEE there are several block types whose behavior is impacted by restart handling. The following is a list of these block types with a summary description of their behavior, categorized by Control Builder library.

#	Library	Block Type	Summary Description
1	SYSTEM	CEE FB	Reports notifications which record fact of warm or cold restart in event journal when idle to run transition occurs.
1	SYSTEM	CONTROLMODULE	Forwards selected restart commands (warm restart or cold restart) to component basic blocks. Supports option parameter which allows selection of whether the CONTROLMODULE should follow CEE, always restart cold or always restart warm upon idle to run transition.
1	SYSTEM	SCM	Supports option parameter which allows selection of whether the SCM should follow CEE, always restart cold or always restart warm upon idle to run transition. Supports a set of additional parameters which allow application engineers to control

#	Library	Block Type	Summary Description
			how an SCM resumes execution upon restart. Supports signal parameter which tells type of restart executed.
2	SCM	STEP	Holds instruction state information which is held or reset based on selection of cold or warm restart and on SCM configuration.
3	PULSEINPUT	PITOTALIZER	On cold restart, stops totalizing. On warm restart, either stops totalizing or continues totalizing based on configured equation.
4	AUXILIARY	TOTALIZER	Same as above.
4	AUXILIARY	SIGNALSEL	On a Warm restart, Cold restart or when this FB is inactivated the following parameters are initialized P(1) through P(6) values are set to NaN. PSTS(1) through PSTS(6) are set to BadValSts.
5	REGCTL	AUTOMAN	Either holds last mode or sheds to manual based upon whether warm or cold restart is selected and upon the nature of the cascade relationship.
5	REGCTL	ENHREGCALC	Same as above.
5	REGCTL	FANOUT	Same as above.
5	REGCTL	INCRSUMMER	Same as above
5	REGCTL	OVRDSEL	Same as above.
5	REGCTL	PID	Same as above.
5	REGCTL	PIDEXTRFB	Same as above.
5	REGCTL	PIDFF	Same as above.
5	REGCTL	PID-PL	Either holds last mode or sheds to manual based upon whether warm or cold restart is

#	Library	Block Type	Summary Description
			selected and upon the nature of the cascade relationship.
5	REGCTL	POSPROP	Either holds last mode or sheds to manual based upon whether warm or cold restart is selected and upon the nature of the cascade relationship.
5	REGCTL	PULSECOUNT	Same as above.
5	REGCTL	PULSELENGTH	Same as above.
5	REGCTL	RATIOBIAS	Either holds last mode or sheds to manual based upon whether warm or cold restart is selected and upon the nature of the cascade relationship.
5	REGCTL	RATIOCTL	Same as above.
5	REGCTL	REMCAS	Same as above.
5	REGCTL	REGCALC	Same as above.
5	REGCTL	REGSUMMER	Same as above
5	REGCTL	SWITCH	Same as above.
6	HIWAYIF	HIWAYOUT	Supports cascade relationship to enable prescribed restart behavior for RegCtl primary in ACE.
7	UCNIF	UCNOUT	Same as above.
8	LOGIC	STARTSIGNAL	Provides parameters which can optionally be used within CONTROLMODULE configurations to trigger initializations in response to restart events.

Configuration of Restart Behaviors

Restart behaviors vary depending on the type of block and the cascade control design. To view the specific configurations relating to restart behavior,

Topic	For more information
CEE Block	Click here
Control Module (CM)	Click here
Sequential Control Module (SCM)	Click here
Control Module Function Blocks	Click here
Regulatory Cascades	Click here
Regulatory Control Blocks	Click here
UCNIF:UCNOUT	Click here
Logic Block	Click here

CEE Block

The CEE function block does nothing that distinguishes warm restart from cold restart except for notification reporting. Its behavior with respect to notification reporting is as follows.

- On Warm restart, the CEE FB reports the idle to run state change event occurred with a warm restart.
- On Cold restart, the CEE FB reports the idle to run state change event occurred with a cold restart.

Control Module (CM)

Control modules do not have functionality modified by selection of warm or cold restart but the behavior of blocks contained within the control module can be impacted. Control Modules therefore have a configuration parameter CEESTARTOPT which optionally allows the application designer to override the warm or cold selection made by an operator at runtime.

CM Parameter	Configuration Options
CEESTARTOPT (Access Lock = ENGR)	• ALWAYSCOLD For this configuration parent control modules instruct their component blocks to do a cold restart regardless of which restart option has been selected for the CEE as a whole. • ALWAYSWARM For this configuration parent control modules instruct their component blocks to do a warm restart regardless of which restart option has been selected for the CEE as a whole. • FOLLOWCEE (default) For this configuration parent control modules instruct their component blocks to initialize according to whichever type of restart the operator has selected for the CEE as a whole: cold or warm.

Control Module Function Blocks

Block	For more information
Pulse Input Totalizer	Click here
Auxiliary Totalizer	Click here
StartSignal	Click here

Pulse Input Totalizer

On CEE startup the behavior of the PITOTALIZER block varies depending upon whether warm or cold restart has been commanded and upon how parameter PVEQN is configured. The value selected for PVEQN actually controls PITOTALIZER behavior on restart handling and on bad input handling. This is shown in the following table.

PVEQN	Bad Input Handling	Warm Restart Handling	Cold Restart Handling
EQA (default)	Stop accumulation while input is bad.	Continue after input turns valid.	Set PV to bad and stop accumulation.
EQB	Use last good value if input is bad.	Continue after input turns valid.	Set PV to bad and stop accumulation.
EQC	Stop if the input is bad and set PV to NaN.	Continue after input turns valid.	Set PV to bad and stop accumulation.
EQD	Stop accumulation while input is bad.	Set PV to bad and stop accumulation.	Set PV to bad and stop accumulation.
EQE	Use last good value if input is bad.	Set PV to bad and stop accumulation.	Set PV to bad and stop accumulation.
EQF	Stop if the input is bad and set PV to NaN.	Set PV to bad and stop accumulation.	Set PV to bad and stop accumulation.

Auxiliary Totalizer

All restart behavior of the AUXILIARY TOTALIZER block is equivalent to that of the PULSEINPUT PITOTALIZER block.

Startsignal

STARTSIGNAL is a function block supported to aid in the handling of restarts within Control Modules. It may be optionally used within any Control Module to give the designer better control over how the module initializes in response to events such as cold or warm restart. STARTSIGNAL is useful in logic gate configurations which hold state.

Block STARTSIGNAL supports read-only parameters which may be accessed to drive initialization actions. Each parameter has "pulse" semantics which means that each parameter normally holds a value which indicates that no initialization is required. When a transition occurs appropriate parameters acquire an informative value. This value lasts until the end of the first block execution which follows the transition. After first

execution the parameter is reset to a value which indicates that no restart has occurred since the last execution. A STARTSIGNAL instance must always be configured so that its ORDERINCM parameter places its execution after that of any blocks which read its parameters.

Parameter	Description
RESTART	An enumeration valued summary parameter. Following a transition, it shows a value different from NONE until the end of first block execution. For each of the values of RESTART, a corresponding flag is set. The flag holds true until the restart state changes. The possible values for RESTART and the corresponding flags are: • NONE - normally shown. • CMLOAD – Holds CMLOADFL true for the first execution cycle following a load of the parent control module. • CMACTIVE – Holds CMACTIVEFL true for the first execution cycle following inactivation and then activation of the parent control module. • CEECOLD –Holds CEECOLDFL true for the first execution cycle following cold start of the CEE. • CEEWARM –Holds CEEWARMFL true for the first execution cycle following warm start of the CEE. • CEESWITCH - Holds CEESWITCHFL true for the first execution cycle following redundant switchover of the CEE on a C200. • ANYRESTART – Holds ANYRESTARTFL true for the first execution cycle if any of the preceding flag parameters are true. It is possible for more than one transition to occur between the time that a control module stops executing and the time that it restarts. For this reason, there is a priority built into the values of parameter RESTART. This priority insures that the strongest initialization signal is the one which will be seen. The priority for RESTART values is CMLOAD > CMACTIVE > CEECOLD > CEEWARM > CEESWITCH > NONE. There is no implied priority or mutual exclusivity among the flag parameters. Consider a scenario in which the CEE starts out in idle, the control module is loaded, the control module is

Parameter	Description
	activated, then the CEE is warm started. On first execution of the control module parameters CMLOADFL, CMACTIVEFL, CEEWARMFL and ANYRESTARTFL will be True. Parameters CEECOLDFL and CEESWITCHFL will be False. Parameter RESTART will have the value CMLOAD.

Parameter RESTART may be displayed on the symbol faceplate of block RESTART for monitoring purposes but it will only show a value other than NONE until first execution.

Parameter RESTART may be used to drive initializations within the control module but in many cases it will be more convenient to use one of the associated Boolean-valued flag parameters.

Warm and Cold Restart Behaviors of Cascades

Definitions:

CEE-to-CEE Cascade - a cascade relationship in which the primary is located within the CEE-ACE or CEE-C200 and the secondary is located within a CEE-ACE, CEE-C200 or within an IOP presented by an IOLIM. The primary is always a CEE regulatory control block. The secondary may be a regulatory control block or a direct analog output block (or slot). CEE-to-CEE Cascades can be local (primary and secondary located within the same EE) or peer-peer (primary and secondary located in different EEs).

CEE-to-Legacy Cascade - a cascade relationship in which the primary is located within the CEE-ACE and the secondary is located within a legacy controller presented by an interface gateway or bridge. The primary is always a CEE regulatory control block assisted by a special purpose interface block. The secondary may be a regulatory control block or direct analog output located in an XPM accessed through a TPN Server and one or more additional gateways. It may be a regulatory control block or direct analog output located in a TDC2000 Hiway box accessed through a TPN Server and one or more additional gateways.

Restart Behaviors of	For more information
Regulatory Cascades	Click here
CEE to CEE Cascades	Click here
CEE to Legacy Cascade when Node hosting Primary is restarted	Click here

Restart Behaviors of	For more information
CEE to Legacy Cascade when Node hosting Secondary is restarted	Click here
RegCtl Cascades when Node hosting the Primary is restarted	Click here
Cascades through UCNIF:UCNOUT	Click here

Regulatory Cascades

CEE regulatory control (RegCtl) blocks can be organized in cascade relationships with other RegCtl blocks and with non-RegCtl blocks. The cascades can be confined to a single Execution Environment (EE) or they can extend beyond a single EE. In the context of restart, cascade behavior generally falls into two broad categories.

c) Resume automatic control without human intervention if automatic control was operative before shut down.

d) Do not resume automatic control until there has been some form of acknowledgement from a human operator.

Which of these two behaviors applies depends on:

- the nature of the cascade relationship
- the type of node hosting the primary
- the type of node hosting the secondary
- how the node coming back from shutdown is restarted

When the node undergoing the restart is an ACE or C200 and it is hosting the cascade primary then the CEE generally responds as follows according to restart type.

- Warm Restart
  All cascades resume processing under the mode that applied before shut down.
- Cold Restart
  Cascades in which the secondary is another regulatory block resume processing under the mode that applied before shut down. Cascades in which the secondary is a

direct analog output shed to manual and do not resume automatic control until the operator changes mode.

While the above description conveys a guiding principle, actual behavior can be different depending on the type of node which hosts the cascade secondary. Behavior is as described above when the secondary is hosted by an ACE, C200 or IOLIM. If the secondary is presented by a different type of node (for example a TPN Server) the behavior can be different.

In all cases for all restart types and all topologies, CEE restarts eventually result in full initialization of internal dynamics of RegCtl blocks. The general behavior is that the output (OP) is initialized to match the downstream output value and the internal states of difference equations are set to zero. Set point (SP) is either initialized to track PV or is held unchanged. Whether SP tracks PV or is preserved depends upon the setting of parameters PVTRAKOPT and PVTRAKOPTAI.

CEE to CEE Cascades

When the cascade primary block is hosted by an ACE or C200 and the cascade secondary block is presented by an ACE, C200 or IOLIM, cascades are of the CEE-to-CEE type.

When the node hosting the primary goes through a restart, the behaviors are the same regardless of whether primary and secondary are in the same node or in peer nodes. In the following two behaviors, assume that the database at the primary is reloaded through checkpoint restore and that the mode of the primary RegCtl is restored along with checkpoint.

- When configured with Recall Primary Mode
  The primary mode is retained across restart transition and the primary RegCtl goes through complete initialization and resumes control as consistent with its mode.
- When configured with Shed Primary Mode
  The primary mode is overwritten to manual upon restart transition and the primary RegCtl awaits mode change from operator before initializing and resuming automatic control.

When the node presenting the secondary rather than the node hosting the primary goes through a restart and the primary and secondary are in separate nodes, the CEE-to-CEE cascades have two additional behaviors. These behaviors are the same regardless of whether the fault detected at the primary results from shutdown of the node hosting the secondary or from interruption of communication.

- When configured with Hold Primary Mode across Secondary Fault
  The mode of the secondary RegCtl block is either recalled or shed to manual based on the type of start up and on role of the secondary as primary to a subsequent secondary. The primary RegCtl block holds its mode across fault and recovery of secondary. The primary initializes and resumes control action as appropriate to mode of secondary after its host node restarts.
- When configured with Shed Primary Mode on Secondary Fault
  The secondary AO block has no mode. Primary RegCtl sheds mode to manual on first detection of communication failure with secondary. Primary RegCtl awaits mode change by operator before initializing and resuming automatic control.

The following table enumerates the types of events which cause each of the 4 behaviors listed above.

Cascade Primary RegCtl location	Cascade Secondary type and location	Event	Functional Behavior of Cascade
RegCtl block in ACE	RegCtl block in ACE	Shutdown and cold restart at primary.	Recall Primary Mode
	RegCtl block in C200	Shutdown and cold restart at primary.	Recall Primary Mode
	AO block in C200	Shutdown and cold restart at primary.	Shed Primary Mode.
	AO point under C200 IOLIM	Shutdown and cold restart at primary.	Shed Primary Mode.
RegCtl block in C200	RegCtl block in ACE	Shutdown and cold restart at primary.	Recall Primary Mode
	RegCtl block in C200	Shutdown and cold restart at primary.	Recall Primary Mode
	AO block in C200	Shutdown and cold restart at primary.	Shed Primary Mode.
	AO point under C200 IOLIM	Shutdown and cold restart at primary.	Shed Primary Mode.
RegCtl block in ACE	RegCtl block in ACE	Shutdown and warm restart at primary	Recall Primary Mode
	RegCtl block in C200	Shutdown and warm restart at primary	Recall Primary Mode

Cascade Primary RegCtl location	Cascade Secondary type and location	Event	Functional Behavior of Cascade
	AO block in C200	Shutdown and warm restart at primary	Recall Primary Mode
	AO point under C200 IOLIM	Shutdown and warm restart at primary	Recall Primary Mode
RegCtl block in C200	RegCtl block in ACE	Shutdown and warm restart at primary	Recall Primary Mode
	RegCtl block in C200	Shutdown and warm restart at primary	Recall Primary Mode
	AO block in C200	Shutdown and warm restart at primary	Recall Primary Mode
	AO point under C200 IOLIM	Shutdown and warm restart at primary	Recall Primary Mode
RegCtl block in ACE	RegCtl block in ACE	Shutdown and any recovery at secondary	Hold Primary Mode Across Secondary Fault
	RegCtl block in C200	Shutdown and any recovery at secondary	Hold Primary Mode Across Secondary Fault
	AO block in C200	Shutdown and any recovery at secondary	Shed Primary Mode On Secondary Fault
	AO point under C200 IOLIM	Shutdown and any recovery at secondary	Shed Primary Mode On Secondary Fault
RegCtl block in C200	RegCtl block in ACE	Shutdown and any recovery at secondary	Hold Primary Mode Across Secondary Fault
	RegCtl block in C200	Shutdown and any recovery at secondary	Hold Primary Mode Across Secondary Fault
	AO block in C200	Shutdown and any recovery at secondary	Shed Primary Mode On Secondary Fault

Cascade Primary RegCtl location	Cascade Secondary type and location	Event	Functional Behavior of Cascade
	AO point under C200 IOLIM	Shutdown and any recovery at secondary	Shed Primary Mode On Secondary Fault

CEE to Legacy Cascade when Node hosting Primary is restarted

When the cascade primary is hosted by an ACE and the cascade secondary is presented by a TPN Server, cascades work according to a CEE-to-Legacy relationship. The database at primary is reloaded through checkpoint restore and the Mode of the primary RegCtl block is restored along with checkpoint. CEE-to-Legacy cascades have 2 behaviors when the node hosting the primary goes through a restart.

- When configured for Honor Secondary Cascade Request
  The primary mode is retained across restart transition. At start of execution the primary RegCtl honors any outstanding cascade requests at downstream secondaries. The primary RegCtl block goes through complete initialization and resumes control as consistent with its mode.
- When configured for Clear Secondary Cascade Request
  At start of execution the primary RegCtl turns off any outstanding cascade requests at downstream secondaries and leaves the primary Mode unchanged. The primary RegCtl awaits command from operator before initializing and resuming automatic control.

The following table enumerates the types of events which cause each behavior listed above.

Cascade Primary	Cascade Secondary	Event	Functional Behavior of Cascade
Regulatory control block in ACE	SPC cascade to regulatory control slot under TPN Server	Shutdown and warm restart at primary.	Honor Secondary Cascade Request
	DDC cascade to regulatory control slot under TPN Server	Shutdown and warm restart at primary.	Honor Secondary Cascade Request
	DDC cascade to AO slot under TPN Server	Shutdown and warm restart at primary.	Honor Secondary Cascade Request

CEE to Legacy Cascade when Node hosting Secondary is restarted

For the case where the primary of a regulatory cascade is hosted by an ACE, the secondary is presented by a node other than ACE, C200 or IOLIM and the secondary's node is the one that goes through restart, there are a variety of possible behaviors for cascades. Events described focus on restart of gateway nodes which present the secondary to the primary.

In this instance, control automatically resumes with the TPN Server restarting, the Cascade requests fulfilled and automatic control resuming as consistent with primary mode.

The following table enumerates the types of events which cause behavior listed above.

Cascade Primary	Cascade Secondary	Event	Functional Behavior of Cascade
Regulatory control block in ACE	SPC cascade to regulatory control slot under TPN Server	Shutdown and restart at TPN Server	Automatically Resume Control
	DDC cascade to regulatory control slot under TPN Server	Shutdown and restart at TPN Server	Automatically Resume Control
	SPC cascade to AO slot under TPN Server	Shutdown and restart at TPN Server	Automatically Resume Control

RegCtl Cascades when Node hosting the Primary is restarted

Regulatory Control blocks AUTOMAN, FANOUT, INCRSUMMER, OVRDSEL, PID, PIDEXTRFB, PIDFF, PID-PL, POSPROP, PULSECOUNT, PULSELENGTH, RATIOCTL, REGCALC, REGSUMMER, REMCAS, SUMMER, and SWITCH behave as follows:

Type of Cascade	Type of Restart	Behavior
CEE to CEE	Cold	When the secondary is a regulatory control block, mode is preserved. When the secondary is a direct analog output, mode is overwritten to manual.
	Warm	Mode is preserved regardless of the type of downstream secondary.

Type of Cascade	Type of Restart	Behavior
CEE-to-Legacy	Cold	Mode is preserved for all cascades. Outstanding cascade requests at downstream secondaries are reset.
	Warm	Mode is preserved for all cascades. Outstanding cascade requests at downstream secondaries are honored.

Cascades through UCNIF:UCNOUT

UCNOUT serves as an interface agent between ACE RegCtl blocks acting as primary and RegCtl blocks hosted by an XPM acting as secondary. Access to the XPM on UCN is via the TPN Server.

Though not a RegCtl block in itself, UCNOUT is designed to support appropriate restart behavior for the cascades it mediates.

Behavior implemented within UCNOUT supports the restart functionality described by the following table.

Cascade Primary	Cascade Secondary	Event	Functional Behavior of Cascade
Regulatory control block in ACE	Cascade to regulatory control or AO slot under TPN Server	Shutdown and cold restart at primary.	Clear Secondary Cascade Request
Regulatory control block in ACE	Cascade to regulatory control or AO slot under TPN Server	Shutdown and warm restart at primary.	Honor Secondary Cascade Request
Regulatory control block in ACE	Cascade to regulatory control or AO slot under TPN Server	Shutdown and restart at TPN Server	Automatically Resume Control

Operations

Across Experion there are several different kinds of subsystems which respond to "activation" or "start up" commands. Though similar, the manner in which these subsystems respond is not equivalent in all cases. The table below illustrates the differences.

Subsystem	Supports Activation	Supports Warm Start Idle to Run Transition	Supports Cold Start Idle to Run Transition	Supports Generic Idle to Run Transition
CM	Yes	No	No	No
IOM	Yes	No	No	No
CEE	No	Yes	Yes	No
IOLIM EE	No	No	No	Yes
FIM EE	No	No	No	Yes

Because of the different types of startup transitions supported across different Experion subsystems, the HMI which issues startup commands includes Restart functionality. The following principles apply:

- "Activation" applies only to CMs, SCMs and IOMs
- "Run", "Warm Start" and "Cold Start" are applied, as appropriate, to the idle to run transition of Execution Environments

CEE Function Block

CEE Function Block represents an Execution Environmemt (EE). To change the CEECOMMAND:

Step	Action
1	Open the CEE function block form either directly from Control Builder project or monitor.
2	Choose the command from the CEE Command pull down menu. Pull down menu choices are: None, Idle, WarmStart, and ColdStart.
3	To accomplish an Idle to Run transition, choose Run, WarmStart, or ColdStart. Commanding None has no effect.

ACE Function Block

ACE Function Block represents a controller platform and is unique in that it presents CEECOMMAND and CEESTATE for view and manipulation. Commands for CEECOMMAND in the ACE Function Block is parallel to the CEE Function Block.

CPM Function Block

Commands for CEECOMMAND in the CPM200 Function Block is parallel to the CEE Function Block.

External OPC Server Support

OPC Data Access

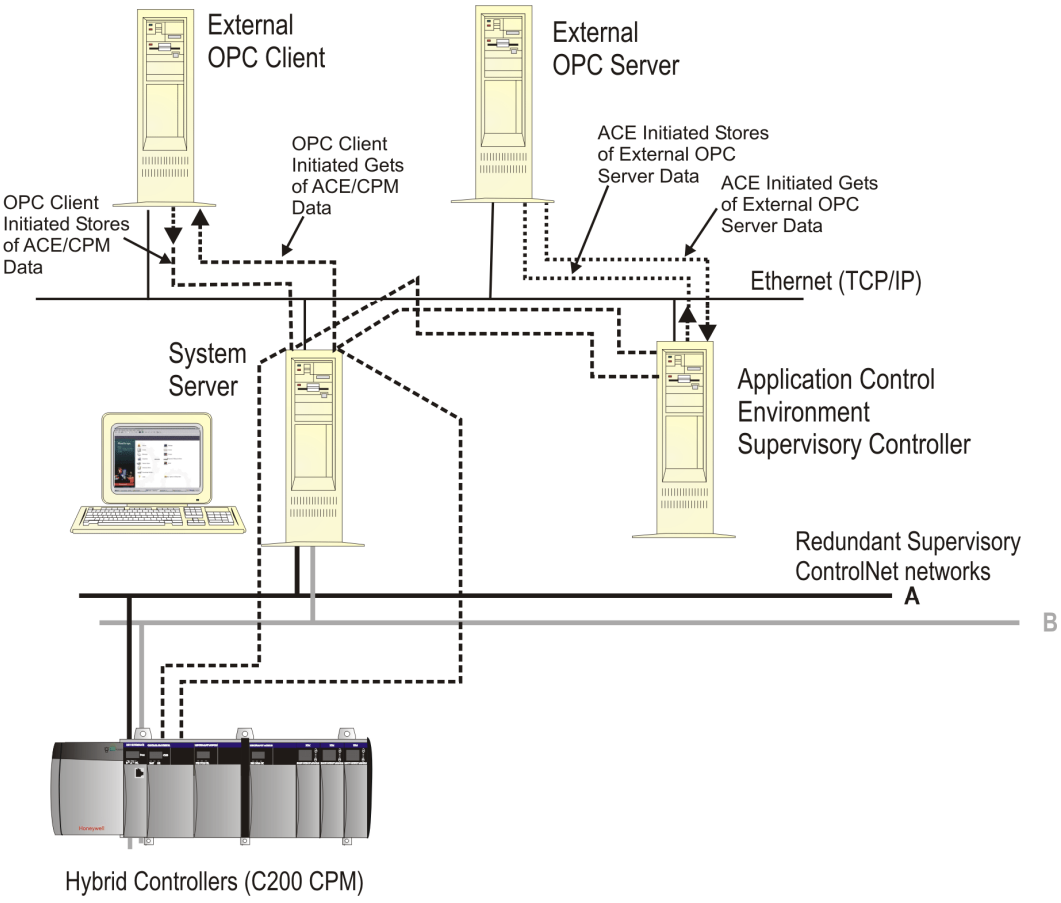
OPC Server function block

The Control Builder application includes a function block that represents an external OPC server. This is an independent, tagged block that is used to identify an external subsystem to exchange data with the system through the Application Control Environment (ACE) supervisory controller.

Users specify the communications path to the OPC server through the blocks configuration form. This data is stored in the System Repository when the block is loaded.

OPC client/server data flow

The following figure illustrates how the ACE supervisory controller initiates communications directly with external OPC servers and indirectly with external OPC clients through the OPC server of the system Server. The CPM in the C200 Process Controller does not support an OPC client, but OPC clients can access its data through the OPC server of the system Server.


External
OPC Client
External
OPC Server
OPC Client
Initiated Gets
of ACE/CPM
Data
ACE Initiated Stores
of External OPC
Server Data
ACE Initiated Gets
of External OPC
Server Data
OPC Client
Initiated Stores
of ACE/CPM
Data
Ethernet (TCP/IP)
System
Server
Application Control
Environment
Supervisory Controller
Redundant Supervisory
ControlNet networks
A
B
Hybrid Controllers (C200 CPM)

OPC Data References

OPC Data Name Syntax

Use the following naming syntax to identify the data of interest in the OPC server.

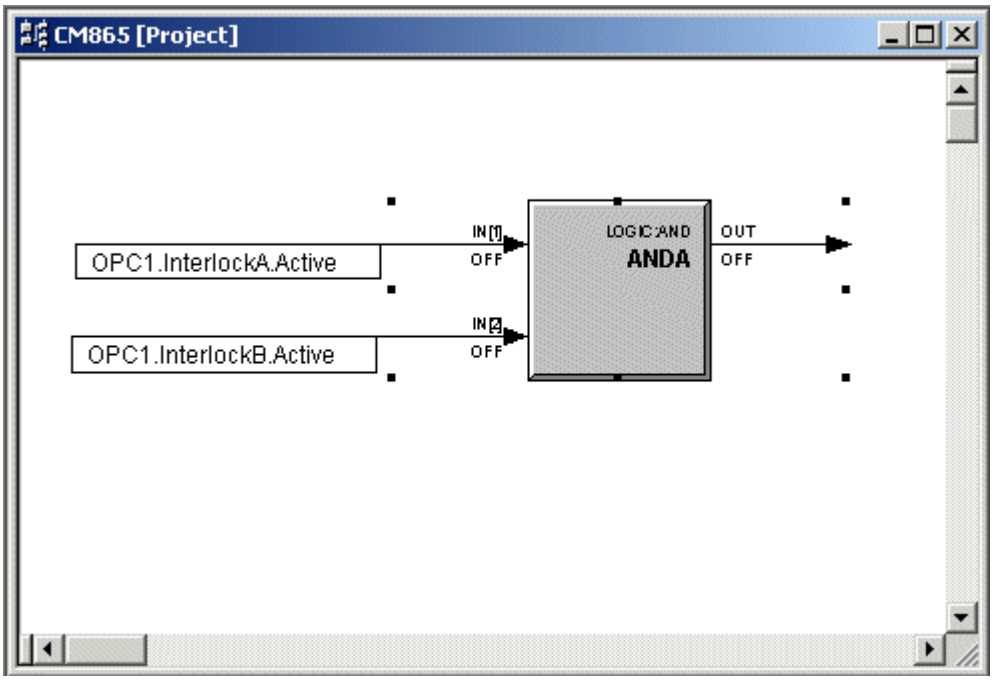
<OPC server function block name>.<OPC server data specific name>

For example, you configure an OPC server block in Control Builder with the name OPC1 and you want to access data named InterlockA.Active in the configured OPC server. The correct reference to this data would be formatted as follows:

OPC1.InterlockA.Active

Parameter connectors only

Since an OPC server block cannot be contained in a Control Module (CM), you must use parameter connectors to route OPC server data elements to applicable block connections in the control strategy. The following figure shows two parameter connectors being used to connect data elements named InterlockA.Active and InterlockB.Active from the configured OPC server named OPC1 to inputs 1 and 2 of an AND function block in a Control Module named CM865 for reference. This strategy is only valid if loaded to an ACE/CEE environment.



OPC references in expressions

You configure OPC data references in SCM Step Output, SCM Transition, AUXCALC block, or REGCALC block the same as other system parameters using the valid naming syntax for the OPC data elements. The following figure shows a SCM Step Output expression used to read the "Count.PV" data from an OPC server named OPC2, add 1.0 to the read value, and store the new value to the Count.PV data in OPC2.

#		Step Outputs
1	-	OPC2.Count.PV := OPC2.Count.PV + 1.0

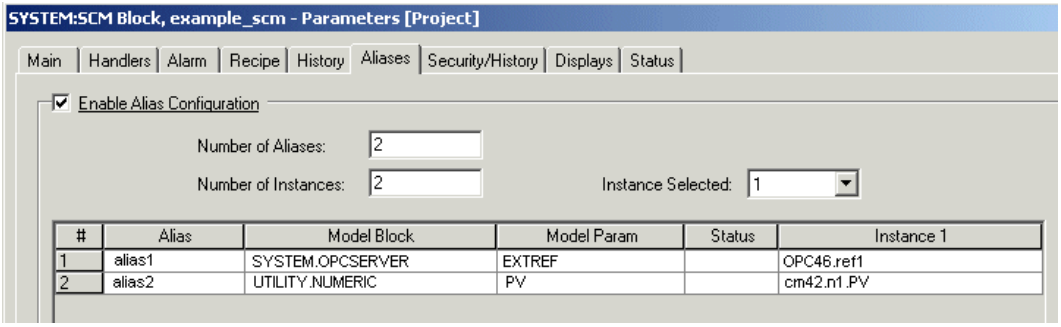
Down Up Del Add

OPC references in SCM Alias table

Use the following guidelines to configure OPC data references in the Aliases table for a SCM block.

- Choose the OPC server block as the Model Block for the given Alias.
- Choose either the External Reference (EXTREF) or the External Reference Structure (EXTREFSTRUCT) parameters of the OPC server block for the given Alias.
  - For an Alias with the EXTREF parameter, the instance references can be references to either OPC server data or FLOAT64 type system Server data.
  - For an Alias with the EXTRETSTRUCT parameter, the instance references can be references to either the OPC server data or EXTREF_STRUCT type system Server data.

The following figure shows a sample reference to OPC server data in a SCM Aliases table. The first alias is for an OPC server block with an EXTREF parameter for an Instance 1 data reference from an OPC server named OPC46. The second alias is for a Numeric block with a PV parameter for an Instance 1 data reference from a Numeric block named n1 in a Control Module named cm42.



OPC Data Type Conversions

About data type

Since the data type of the OPC server data reference is not known at configuration time, it is possible that the actual data type accessed at run time will not match the data type expected by the control strategy in the ACE supervisory controller. Please review the following sections for a summary of the guidelines covering data type conversions for OPC data Gets and Stores.

Gets conversions

The ACE supervisory controller receives the actual OPC data type for "gets" at run time and translates the value into a system data type based on the following translation table.

If OPC VARTYPE Is . . .	Then, System Data Type Is . . .
VT_BOOL	BOOL
VT_UI1	UINT8
VT_UI2	UINT16
VT_UI4	UINT32
VT_UI8	UINT64
VT_I1	INT8
VT_I2	INT16
VT_I4	INT32
VT_I8	INT64
VT_R4	FLOAT32
VT_R8	FLOAT64
VT_BSTR	STRING
VT_FILETIME	TIME
VT_ARRAY VT_BLOB	EXTREF_STRUCT

If OPC VARTYPE Is . . .	Then, System Data Type Is . . .
Other VT_ data types	Bad Value Status

The general translation scenario is as follows.

- The CDA-sp performs the interface specific translation of external reference type to system type.
- The system value status is set according to the quality bit field of the OPC item state quality.
- The substatus and limits bit fields of the OPC item are ignored.
- The OPC array and blob variant types are forced into a system External reference structure type to distinguish them from internal system structures. A specific purpose function block, such as the UCNOUT block, is required to use an External reference structure type.
- The OPC variant types that are not supported result in a data type mismatch error.
- The CDA-sp delivers a system value to the appropriate function block.
- When the data types do not match, the function block receiving the data makes an attempt to convert between the actual and expected data types.

General data conversion considerations

The following table shows the data type conversions that the system does or does not support. A Yes means that the conversion is supported or not required, when the data types match, and a No means that the data type conversion is not supported.

From / **To**	BOOL	UINT8 UINT16 UINT32	UINT 64	INT8 INT16 INT32	INT 64	ENUM SD_ENUM	FLOAT 32 FLOAT 64	STRING	TIME
BOOL	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No
UINT8	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No
UINT16	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No
UINT32	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No
UINT64	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No
INT8	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No
INT16	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No
INT32	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No
INT64	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No
FLOAT 32	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No
FLOAT 64	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No
STRING	No	No	No	No	No	No	No	Yes	No
TIME	No	No	No	No	Yes	No	No	No	Yes
EXTREF_STRUCT	No	No	No	No	No	No	No	No	No

Keep the following considerations in mind when dealing with data conversions.

- Be aware that downcasting occurs when conversions involve a type that supports a wider range of values to a type with a more narrow range of values.
  - For example, an INIT16 value of 333 is converted to UNIT8 value of 255.

  - For example, an INT32 value of 100,000 is converted to the maximum Enumeration value of 65,535. The function block that receives the Enumeration ordinal of 65,535 will likely reject this value, since it is out of range for the applicable Enumeration parameter. This is consistent with the current behavior where an out of range store to an Enumeration parameter does not cause the destination parameter to assume the failsafe value.
  - For example, an OPC server data type of Integer is connected to a PID block's MODE parameter. If the Integer value is 7 at run time, the MODE parameter rejects the value because it is outside the MODE enumeration range of 0 to 5.

- Float to integer conversions use truncation instead of rounding. For example, a Float of 3.75 is converted to the Integer 3.
- The underlying type for system Boolean values is unsigned character. Since the underlying type for Boolean external references cannot be assumed, all external Boolean references are converted in the context of the system, which uses 0 for False and 1 for True. For example:
  - Boolean True converts to Integer 1 or Float 1.0
  - Boolean False converts to Integer 0 or Float 0.0
  - Integer 7 converts to Boolean True
  - Integer 0 converts to Boolean False
  - Integer –336 converts to Boolean True
  - Float 33.33 converts to Boolean True
  - Float –0.567 converts to Boolean True
  - Float 0.0 converts to Boolean False
  - Float 0.0001 converts to Boolean True (Similar to expressions, no threshold value is used in the test for zero. The TypeConvert block does provide a threshold for Float compares to zero, so use this block when this conversion is a concern.)
  - Enumeration ordinal 0 converts to Boolean False
  - Enumeration ordinal 8 converts to Boolean True
  - Float 5.82 converts to Enumeration ordinal 5
  - Float –11.0 converts to Enumeration ordinal 0

- Conversions between Strings and Integers and between Strings and Floats are not supported

The following table summarizes the failsafe data value that is substituted when a given data type conversion is not supported.

If Data Type Is . . .	Then, Failsafe Value Is . . .
BOOL	Off
UINT8 UINT16 UINT32 UINT64	0
INT8 INT16 INT32 INT64	0
ENUM SD_ENUM	Ordinal value of 0
FLOAT32 FLOAT64	NaN
STRING	Blank
TIME	0
EXTREF_STRUCT	Bad Status

For example, if the OPC1.InterlockA.Active parameter used as the input to an AND block in the previous figure in section Parameter connectors only returns a Boolen, Integer, or Float data type at run time, the appropriate conversion is made and the data get is completed successfully. If the OPC1.InterlockA.Active parameter returns a String data type, the Boolean failsafe value of Off is applied to the AND block input.

Stores conversions

The CDA-sp receives the store value from the initiating function block and converts it as needed before storing the value to the OPC server. The CDA-sp executes a one time read of the value from the OPC server to learn its actual data type before initiating the Store.

The ACE supervisory controller supports the UCNOUT block, PUSH block, and SCM Step Output expressions for initiating stores to the OPC server.

ACE interface to TPS system as OPC server

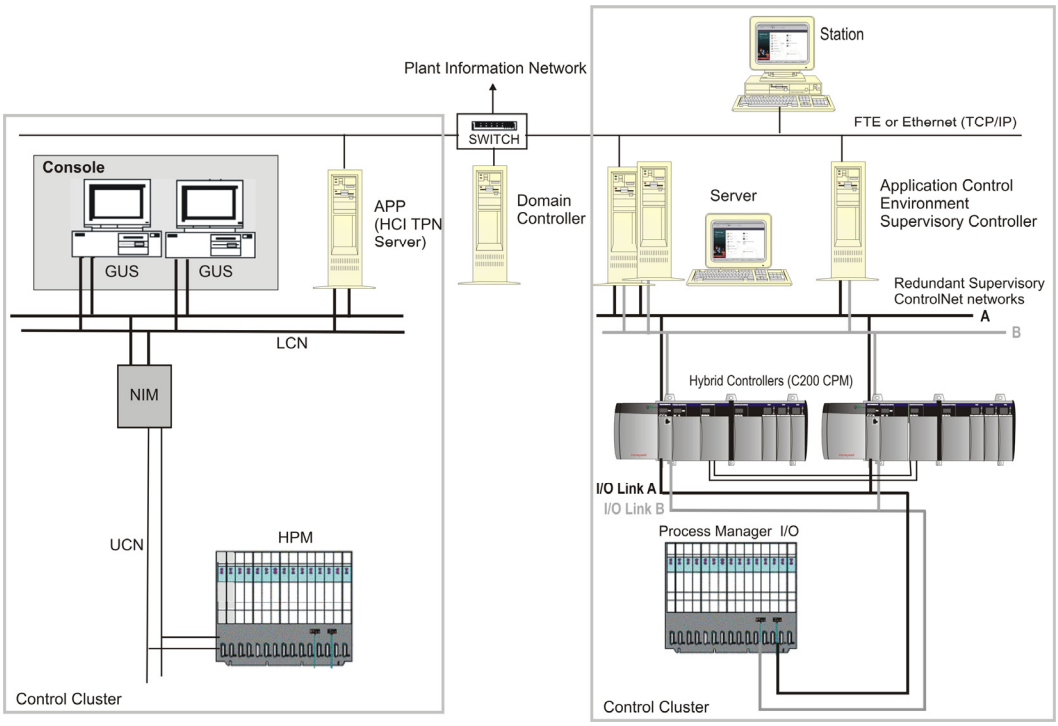
ACE recognizes HCI

The ACE supervisory controller interfaces to the TPS system through an Application (APP) node running Total Plant Network (TPN) server as another OPC server - See the figure below. The TPN server supports both OPC data access and Honeywell Communications Infrastructure (HCI) extensions to OPC. The ACE supervisory controller recognizes when it is communicating with an OPC server that includes HCI, such as the APP node, and provides the following additional functions.

Added ACE Function With HCI Server	What the Added Functions Provides
Experion Server to TPS system access level propagation	Access levels for stores from Control Strategies loaded in ACE supervisory controller to a TPN server in APP node are sent along with the data. The Sequential Control Modules (SCMs) in Control Strategies use Program access level, while other function blocks like the UCNOUT block use a Continuous Control access level. Since the access level definitions for an Experion system and a TPS system are consistent, no translation is necessary.
Redirection Server	When the Redirection Server is installed on the ACE node, the CDA-sp interfaces to it just like a remote HCI server. As long as the OPC server block configured in Control Builder identifies the HCI Redirection Server, the HCI Redirection capabilities are used.

TIP

The additional HCI related functions are transparent to the Control Modules and Sequential Control Modules in control strategies that are executing in the ACE supervisory controller. No changes are required in control strategy configurations through Control Builder whether the OPC server being accessed supports HCI or not.


Plant Information Network
SWITCH
Station
FTE or Ethernet (TCP/IP)
Console
GUS
GUS
APP
(HCI TPN
Server)
Domain
Controller
Server
Application Control
Environment
Supervisory Controller
Redundant Supervisory
ControlNet networks
A
B
LCN
NIM
Hybrid Controllers (C200 CPM)
I/O Link A
I/O Link B
Process Manager I/O
UCN
HPM
Control Cluster
Control Cluster

System Namespace Workaround

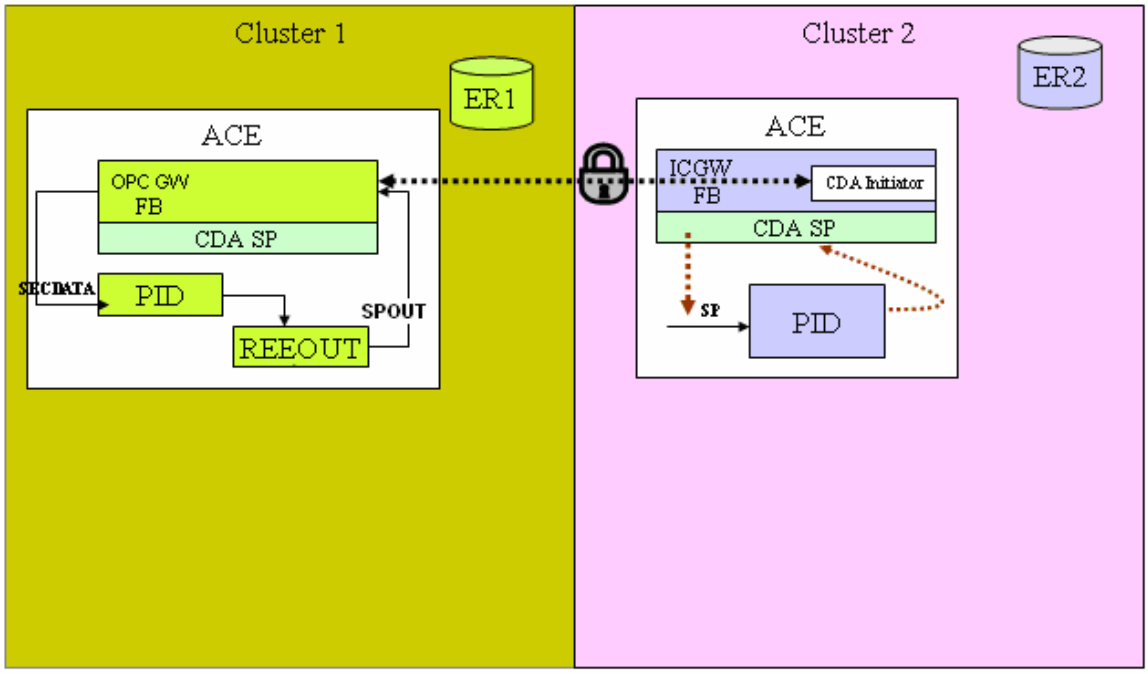
Inter-Cluster Communication

Functional Overview

The issue of communication between Experion clusters is handled in R300 by the introduction of the following components and configuration changes:

- The Inter Cluster Gateway (ICG) block
- The Remote EEOUT (REEOUT) function block
- The addition of SP "push" to the configuration of a subset of the existing function blocks

The relationship of these components is shown below:



Using the Inter Cluster Gateway for Communication between Experion Clusters

Overview

The Inter Cluster Gateway (ICG) serves as a communication bridge between the Application Control Environment (ACE) (or Simulation-Application Control Environment (SIM-ACE)) supervisory controllers in two separate Experion clusters.

The ICG supports dual roles, server and client. Every ICG block acts as an OPC server (I-C Server) that can be connected to by an OPC Gateway block. An ICG can also be used as a client to connect to other I-C Servers.

Inter Cluster Gateway configuration guidelines

- An ICG is required for each Experion cluster which contains blocks that are secondary to another Experion cluster.
- The primary Experion cluster requires an OPC Gateway configured to talk to the I-C Server ("Hci.OPCGateway"). The primary Experion cluster can contain an ICG if there is a need for bi-directional control cascade or data transfer.
- There can only be one ICG per ACE.
- The ACE node hosting the ICG may also host OPC Gateways. Up to four gateways of either type can be hosted by the ACE

When to use Inter Cluster Gateway

The Inter Cluster Gateway shares common infrastructure with the OPC Gateway. The following section describes when each block should be used.

The client-server characteristics of the ICG and the OPCG are summarized in the following table:

Characteristic	OPC Gateway	Inter Cluster Gateway
Connect to any OPC server	Yes	No, only to another ICG
Simple Data	Yes	Yes
Whole arrays	Yes	Yes
Control cascade	Yes	Yes

The ACE interconnections possible using the OPCG, ICG, and theREEOUT, UCNOUT, EHGOUT, and HIWAYOUT function blocks are summarized in the following table:

	ACE-UCN	ACE-ACE (different Experion cluster)	ACE-EHG (different Experion cluster)
Connect to	TPN Server	Inter Cluster Gateway	Inter Cluster Gateway
Whole arrays	Yes	Yes	Yes
Control cascade	UCNOUT	REEOUT	EHGOUT

When communicating between Experion clusters, it is required that the Inter Cluster Gateway block is configured and loaded in the secondary Experion cluster. The primary Experion cluster only requires that an OPC Gateway block is used. The primary Experion cluster could also use an ICG block if there is a need for bi-directional control cascade or data transfer.

In the primary Experion cluster, the OPC Gateway block needs to be activated in order to enable communication. The secondary Experion cluster's ICG block makes data available as soon as it is loaded. It only needs to be activated if it is also being used as a client.

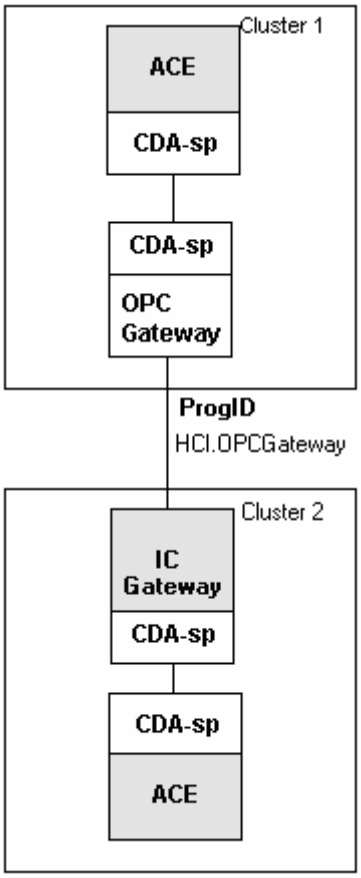
Without the use of Inter Cluster Gateway, the OPC Gateway block can be connected to the Experion OPC server in another Experion cluster to access data. The following table shows when an ICG might be needed.

	Experion OPC Server	Inter Cluster Gateway
Connect to	HwHsc.OPCServer	Hci.OPCGateway
Simple data	Yes	Yes
Whole arrays	No	Yes
Control cascade	No	Yes
Access Level (Continuous Control)	No	Yes

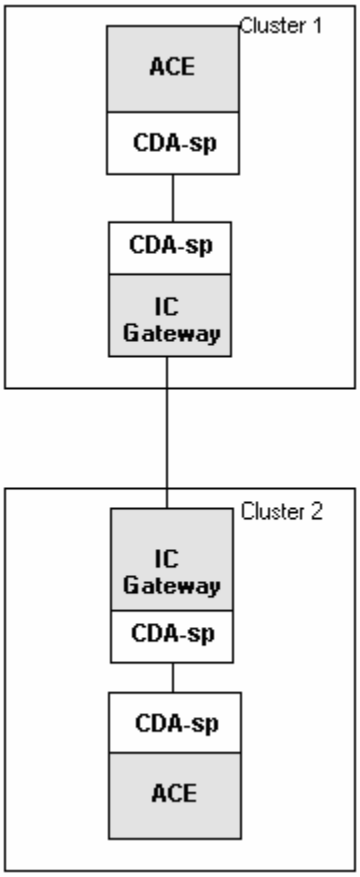
Reviewing Inter Cluster Gateway characteristics

The IC Gateway operates as a bridge/translator between the Control Data Access (CDA) communications protocol between two Experion PKS systems.

The IC Gateway depends on the CDA-supervisory platform (CDA-sp) for its communications link to the rest of the Experion PKS system, but it uses OPC to communicate between Experion clusters, as shown in the following figure.

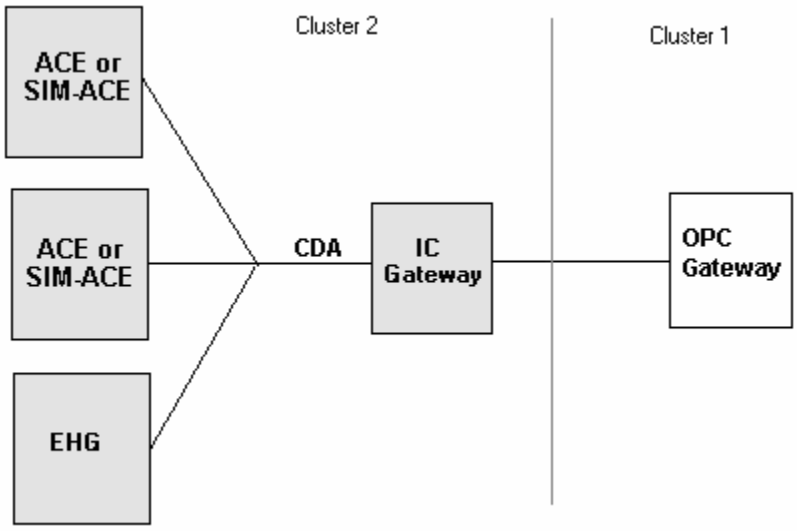


Both Experion clusters can contain an ICG if there is a need for bi-directional control cascade or data transfer, as shown in the following figure.



Inter Cluster Gateway connectivity

The IC Gateway provides a single point of connectivity to an Experion cluster, as shown in following figure.



Remote EEOUT (REEOUT) Function Block

Functional description

The REEOUT function block is used in conjunction with the Inter Cluster Gateway to support regulatory cascades between ACE nodes residing in two separate Experion clusters. It is described elsewhere in this document.

Controller Redundancy Functionality

Basic Redundancy Design Concepts

About Controller redundancy

If you have redundancy compliant Controller hardware, you can implement redundant Controller operation through a Redundant Chassis Pair (RCP). A RCP consists of two Controller chassis which include identical redundancy compliant modules in matching slot positions within their given chassis. The following figure shows a typical hardware configuration for a RCP.

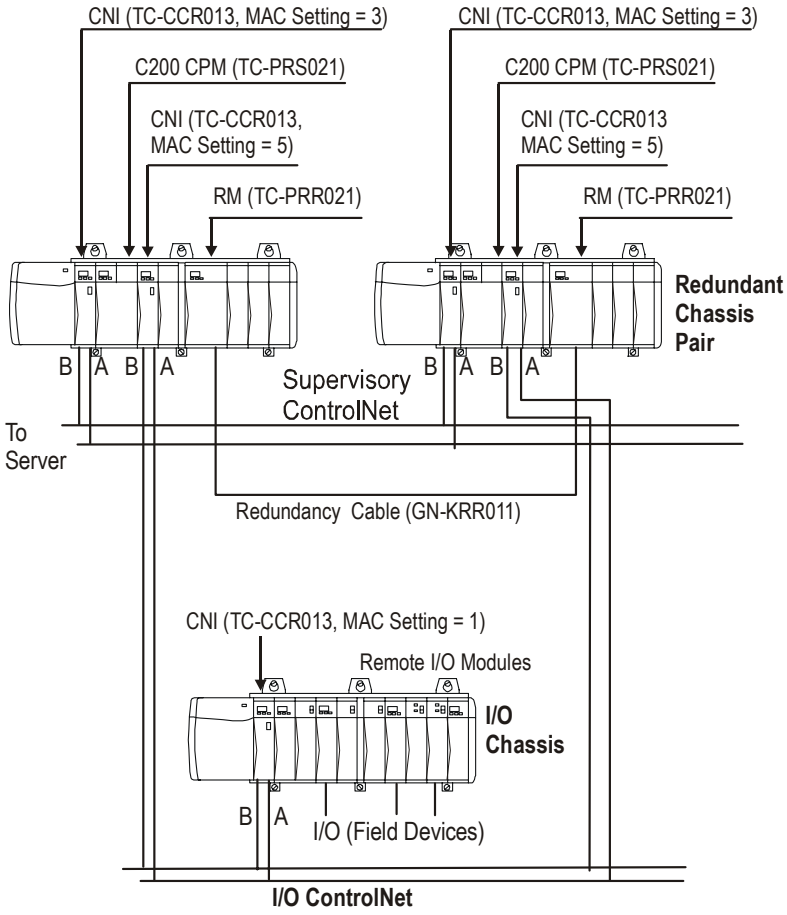


Figure 10 Typical RCP setup in 10-slot chassis.

The goal of Controller redundancy is to improve the availability of the Controller to perform its assigned control functions. The RCP does this by providing a pair of Controller chassis so a component failure in one chassis switches the handling of the assigned control functions to the other chassis. This is considered a dual redundant system, which is characterized by the following two main redundancy states.

- Primary — Refers to the chassis executing the assigned control functions.
- Secondary — Refers to the chassis in some state of readiness to assume the responsibilities of the Primary.

Switchover and Secondary readiness

A switchover describes the process where a Secondary chassis assumes the Primary state, and the Primary chassis assumes the appropriate Secondary state of readiness, depending upon what triggered the switchover. A switchover can be triggered immediately upon the detection of a fault in the Primary or upon the receipt of an operator command.

The ability of a Secondary chassis to take over the assigned control functions of the Primary depends upon which one of the following readiness states reflects its current state.

If Secondary Chassis State is . . .	Then, the Secondary Chassis . . .
Disqualified	Cannot assume the Primary state. This is a state of non-readiness.
Synchronized	Can assume the Primary state upon switchover. In this state, the database in the Secondary is aligned with the database in the Primary. The Secondary closely tracks database changes to maintain its synchronization with the database of the Primary. Otherwise, the Secondary will revert to a Disqualified state.
Standby	*The Standby state is not available at this time.*

Failure conditions and switchover

Conditions that result in switchover

The following table identifies failure conditions that result in a switchover. As noted in the table above, the Secondary control chassis must be in its **synchronized** state for a switchover to occur.

ATTENTION

When any failure that results in a switchover occurs in a Secondary, the Secondary chassis/FIM loses synchronization.

In addition to the failure conditions, these events are reported as diagnostic notifications:

- Loss of view of redundant partner on H1 network (applies to FIM link).
- Loss of private path connection from Primary to Secondary (lonely event).

Failure Conditions That Result in a Switchover
Loss of input power to Primary chassis power supply
Primary chassis power supply fails or is turned Off.
Integrated Control Protocol (ICP) backplane in Primary chassis fails
Any module in Primary chassis fails.
Both ControlNet drop cables to Primary ControlNet module are lost.
Both FTE links to Primary FTE Bridge module are lost.
Either Channel A or Channel B RJ-45 cable to Primary IOLIM is lost.
Inserting any module into a powered Primary chassis.
Removing any module from a powered Primary chassis.
Primary FIM's connection to Redundant Remote Terminal Panel is lost.

Conditions that do not result in a switchover

The following tables identify failure conditions that do not result in a switchover. The first table identifies conditions that are categorized as **non-switchover** triggers or that

trigger no action in the Secondary. The second table identifies conditions that are catergorized as **disqualification** triggers or that trigger disqualified state in the Secondary.

Failure Conditions That Do Not Result in a Switchover – Non-Switchover Triggers
Either ControlNet drop cable A or B to Primary ControlNet module is lost.
Both FTE links to Secondary FTE Bridge module are lost.
One FTE link to Primary FTE Bridge module is lost.
Either or both Channel A and/or Channel B RJ-45 cable(s) to Secondary IOLIM is(are) lost.
One or both H1 cables fail.
One or both H1 network conditioners fail.
The 24-Volt power supply for fieldbus devices fails.

Failure Conditions That Do Not Result in a Switchover – Disqualification Triggers
Loss of input power to Secondary chassis power supply
Secondary chassis power supply fails or is turned Off.
Integrated Control Protocol (ICP) backplane in Secondary chassis fails
Any module in Secondary chassis fails.
Remove any module from a powered Secondary chassis.
Insert any module into a powered Secondary chassis.
Both ControlNet drop cables to Secondary ControlNet module are lost.
Secondary FIM's connection to Redundant Remote Terminal Panel is lost.
Redundancy cable between Primary and Secondary Redundancy Modules is lost.
Data communication failures with Secondary chassis during qualification.
Data communication failures with Secondary chassis during steady-state operation.

Role of the Redundancy Module

The Redundancy Module (RM) serves as a high-performance, chassis-to-chassis, communications bridge for redundancy compliant modules in a RCP. It only provides the path for modules to synchronize themselves and coordinates the synchronization process. The RM does not determine what portions of a module's database get synchronized.

The communication path begins at a given module in a chassis, continues over the chassis backplane to the RM, from the RM in this chassis over the RM-to-RM private path to the partner RM in the partner chassis, and then over the partner chassis backplane to the partner module. The following figure is a simple graphic representation of the RM communication path.

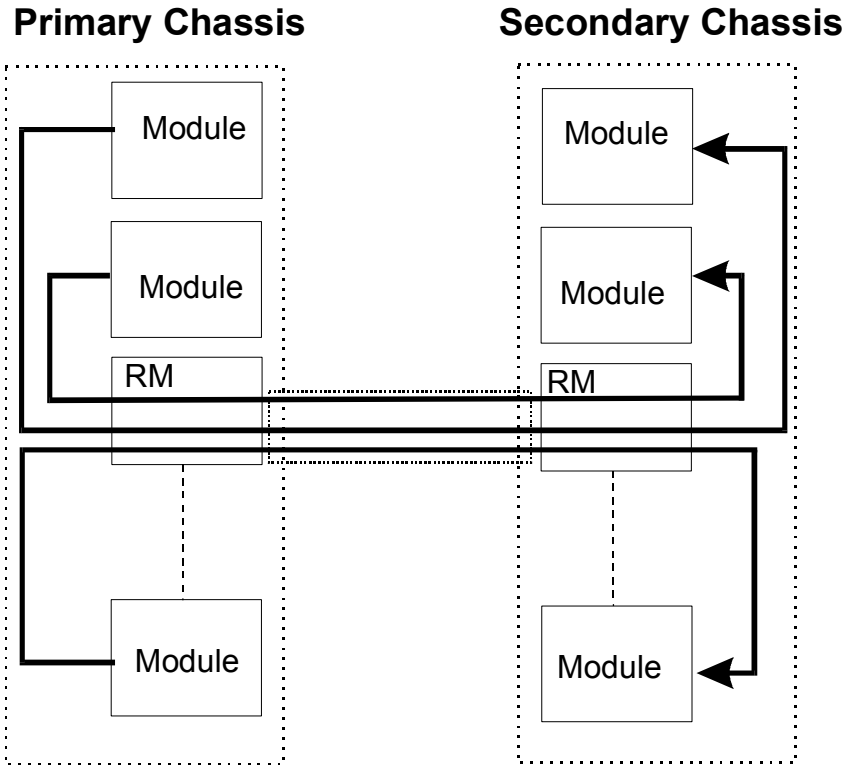


Figure 11 Typical redundancy communications path.

The RM functionality features the following major tasks.

- Gathers module status from all resident chassis modules.
- Gathers module status from its partner RM.
- Receives and logs event reports from resident chassis modules as well as from its partner RM.
- Resolves states based on gathered and reported data.
- Resolves contention situations with its partner RM.
- Coordinates state transition activity for the chassis.
- Provides redundancy-related read (status) and write (control) access to local controllers and remote network devices.

Control Module Independence

CM Independence Background

The design of the C200 controller eliminates some of the potential couplings between control strategies and reduces others. Some forms of coupling, in particular coupling that arises from a finite computing or a communication resource, cannot be entirely eliminated in any controller design. However, the impact can be moderated or controlled.

The C200 has a base software layer called Infrastructure Services, built up through a combination of software components including RTOS and Communication Services. They set up an environment where the execution and communication requirements of control strategies can be met.

C200 architecture also supports a layer of Application Programs with an additional layer of Control Modules. Application Programs and Control Modules work together to create the complete functionality of control strategies.

In software terminology, the Application Programs are what hold the classes for the control algorithms while the Control Modules hold the data object instances.

The separation of object instances from algorithm classes is one of the features of C200 design which allows for a high degree of independence between Modules. Modules can have tunable constants changed on-line. They can also be loaded and unloaded individually. Load and unload of one module has no impact on any other.

Another feature of C200 design which enhances independence is its execution scheduling. In the hypothetical controller, the control period could vary depending on flow through the application program for any particular execution. Different code paths could get executed under different executions causing the control period to vary.

In the C200 design, all control periods are based on a fundamental base cycle whose start time is regulated. While execution timing varies within any real-time software design, the cycle scheme used in C200 controls the amount of variation possible.

Given its design, the types of couplings that exist between Control Modules within a C200 controller are limited to the following.

- Load / Unload
  No coupling.
- Execution Timing
  No coupling.
- Memory Utilization

Modules are coupled in the sense that they all use memory from a common pool. In general, changes in the memory requirements of one module do not impact any other module. However, if the controller is very full then a module to be loaded may not fit into the remaining memory. This can happen if an application engineer has increased the module's configuration so that it requires more memory. Or it can happen if application engineers have changed other modules so that upon reload of this module, not enough memory remains.

By design, memory allocations never occur dynamically within a C200 controller. They only happen as the direct result of a user-initiated module load. In all cases, the failure of a module to load is known immediately by the return of an error message. There are no adverse consequences other than the fact that the module itself can not be loaded. Modules which have already been loaded and which are resident do not require any kind of re-qualification because a different module cannot be loaded.

In general, application engineers should plan the configuration of a C200 controller so that there is always some amount of reserve memory. Parameters TOTALMEM, USEDMEM and FREEMEM of the CPM block should be checked from time to time to determine how much memory remains unused.

CPU Utilization

Like memory, the CPU processing time available within the C200 can be thought of as a single resource pool from which all modules draw. Reliance on this common pool introduces a potential coupling between modules. However, the design of the control processing scheme within the C200 eliminates this coupling except in the case of overrun.

The processing scheme is based on 40 cycles, each one 0.05 seconds long. The start of each cycle is time regulated. The modules to run in each cycle are selected by the application engineer in a balancing procedure that distributes total load across the set of cycles. In a properly balanced C200 configuration, that is not overloaded, a module can

be added or removed without impact on any other module, despite the use of a shared CPU resource.

If the controller as a whole is overloaded, or if one or more cycle is overloaded, it is possible to incur processing overruns. An overrun occurs when cycle N does not finish its processing in time for the start of cycle N+1. If this happens, the start of cycle N+1 is postponed until the time originally designated for the start of cycle N+2. The net effect is to pause control processing by one cycle. No module's processing is skipped. Some timing operations are extended. No timing operations are shortened.

In general, a configuration that incurs one or a few overruns in the space of an hour suffers no degradation of control. However, application engineers should be conscious of the CPU resource when configuring a controller and should design control configurations that do not incur overruns. They should insure that the controller as a whole is not overloaded and that no individual processing cycle is overloaded. This activity does not require re-validation of individual modules. Rather it is a standard procedure that should be followed for the controller as a whole whenever its configuration is changed.

As general guidelines, the total, time-average, free CPU available within the controller should be kept above 20%. This can be determined from the value of the CPM block parameter CPUFREEAVG. The time-average CPU utilization of any individual cycle should be kept below 70%. This can be determined from the value of the CEE block parameter CPUCYCLEAVG(I). In addition, during configuration, application engineers should occasionally check the overrun statistics parameters CRCYCLEOVRN(I) and LSCYCLEOVRN(I) of the CEE block to see whether there have been any overruns within the controller or within any particular cycle of the controller.

If a configuration error is made which causes repetitive overruns which could degrade control, the C200 will report an overrun alarm. This alarm will clear once repetitive overruns cease.

Communication Bandwidth Utilization

Like all controllers the C200 controller uses communication bandwidth for data transfer. Types of data transfer include:

- Communication with IO devices.
- Communication with peer controllers.
- Communication with supervisory HI and control devices.
- Alarm and event reporting.

Communication bandwidth can be viewed as a common resource shared by all modules analogous to memory or CPU processing time. Reliance on this common pool introduces a potential coupling between modules.

Adverse effects from sharing communication bandwidth do not occur if the C200 controller is configured within its specified operating limits. Under these conditions, modules can be added and deleted at will and all modules operate using their needed communication bandwidth with no impact on other modules.

Data Reference

Data reference coupling between modules is an expected part of any control configuration and is consistent with the control mission. The extent to which there is or is not data transfer between modules is completely controlled by the application engineer.

However, the design of the C200 controller and the Experion system employ a feature that minimizes impact to running modules when they are referenced by other modules, by supervisory controllers or by H1 devices. This is that all modules and all algorithm blocks within modules have inherently external parameter linkage. This means that from the moment when the module is first loaded, its entire data content, as exposed through named parameters, can be accessed from outside the controller. When new displays are created or when existing displays are modified, no modules or application programs need change in order to publicize newly accessed data. Similarly, when supervisory or peer control strategies are created which access data for the first time, no change or load of the reference module is required.

Note in particular that because of inherently external parameter linkage, C200 modules do not need to be re-qualified if the read data set is changed after the initial qualification.

There is another characteristic supported by C200 modules and algorithm blocks that reduces the impact of data reference coupling. This is called built-in fail-safe handling. Built-in fail-safe handling is a system wide philosophy that requires every control block to respond safely to conditions of error outside of itself, or to a change of state in which it becomes non-operational.

The typical example is that for algorithm blocks, which support the PV (Process Variable) parameter, such as PID and Data Acquisition, the PV local value is set to IEEE754 Not-A-Number (NAN) in response to communication failure or de-activation. Other blocks, which read a NAN PV, know how to respond safely.

Because of built-in fail-safe handling, it is possible to de-activate and load a Control Module knowing that other modules, which may reference it, will respond with their build-in fail-safe behavior. This reduces the likelihood that an application engineer could cause a control mishap by loading a module.

Implications for Control Builder Functions

Redundant C200 CPM configuration

You can now configure C200 CPMs for redundant Controllers. This creates a partner CPM based on the uplink Media Access Control (MAC) address for the Controller and the numbered slot positions of the CPMs in the RCP. The corresponding redundant CPM icons appear in the Control Builder Project tree view.

RM configuration

You can configure the RM block and its partner RM through the Main tab in the RM/RCP dialog box in the Control Builder Project tree view only. The corresponding redundant RM icons appear in the Project view.

REFERENCE - INTERNAL

Please refer to *Creating Redundancy Modules</HH>* in the Control Building Guide for block configuration details

While the Configuration tab in the RM/RCP dialog box in Control Builder Project view shows the default values as Never for Auto-Synchronization and DISABLED for Program Command Recognition, these values are not written to the RM when the RM block is loaded from the Project view.

For a fully functional pair of RMs, some configuration changes applicable to both RMs can be made through the dialog box for either RM in the Monitoring view. In either case, the Primary RM eventually receives and executes the request, and directs the Secondary to make the same change. If the partner RM is not present, the local RM makes the changes regardless of the redundancy state.

You can also configure Description and Keyword text strings for use with Station displays through the RM profiles tab in the RM/RCP dialog box in the Project view only. You must call up the RM/RCP dialog box for the Primary RM to enter the text strings in the applicable Primary fields on the RM profiles tab. You must call up the RM/RCP dialog box for the Secondary RM to enter the text strings in the applicable Secondary fields on the RM profiles tab. These entries are written to the appropriate RM when the given RM block is loaded from the Project view.

The RM overwrites the unassigned Chassis ID parameter at powerup to be Chassis_A, if its redundancy state is Primary; or Chassis_B, if its redundancy state is Secondary. You can toggle the chassis ID assignment through a button on the Configuration tab or the Display tab in the RM/RCP dialog box.

RM monitoring

You can view relevant information about the operation of the RMs and the RCP and issue online commands through the RM/RCP dialog box in the Control Builder Monitoring tree view. You only need to double-click the Primary or Secondary icon for a loaded RM in the Monitoring tree view to call up the RM/RCP dialog box. As shown in following figure, the dialog box includes nine different categories of information under separately labeled tabs.

TIP

The Identification, Dependencies, and Template Defining tabs only appear when the optional Template license is activatied.

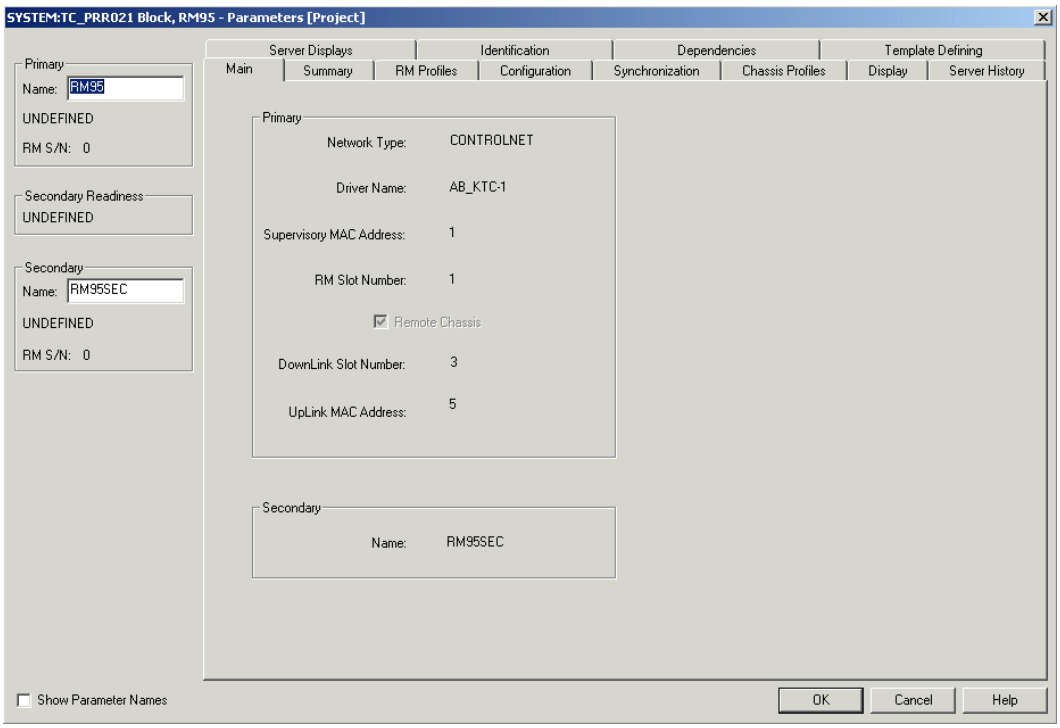


Figure 12 RM/RCP dialog box.

The following paragraphs give a brief description of the functions associated with the RM/RCP dialog box and each of its tabs.

RM/RCP dialog box

The RM/RCP dialog box includes the following status boxes in addition to its tabbed categories.

- Primary— Relates physical hardware data for chassis identification (Chassis_A) and RM serial number to the Primary redundancy state.
- Secondary Readiness — Shows readiness of Secondary chassis to assume the role of the Primary if a switchover occurs. The possible readiness states are:
  - UNDEFINED: Insufficient data to assess readiness. This is generally a startup state.
  - NOPARTNER: Partner (Secondary) is not visible through RM-to-RM private path and can not assume the role of the Primary.
  - DISQUALIFIED: Secondary is present and disqualified for operation as the Primary. It can not assume the role of the Primary.
  - SYNCHRONIZED: Secondary is synchronized with the Primary. It can now assume the role of the Primary.
  - STANDBY: Secondary is not synchronized but it can assume the role of the Primary. This state is not available in R120.
- Secondary — Relates physical hardware data for chassis identification (Chassis_B) and RM serial number to the Secondary redundancy state.

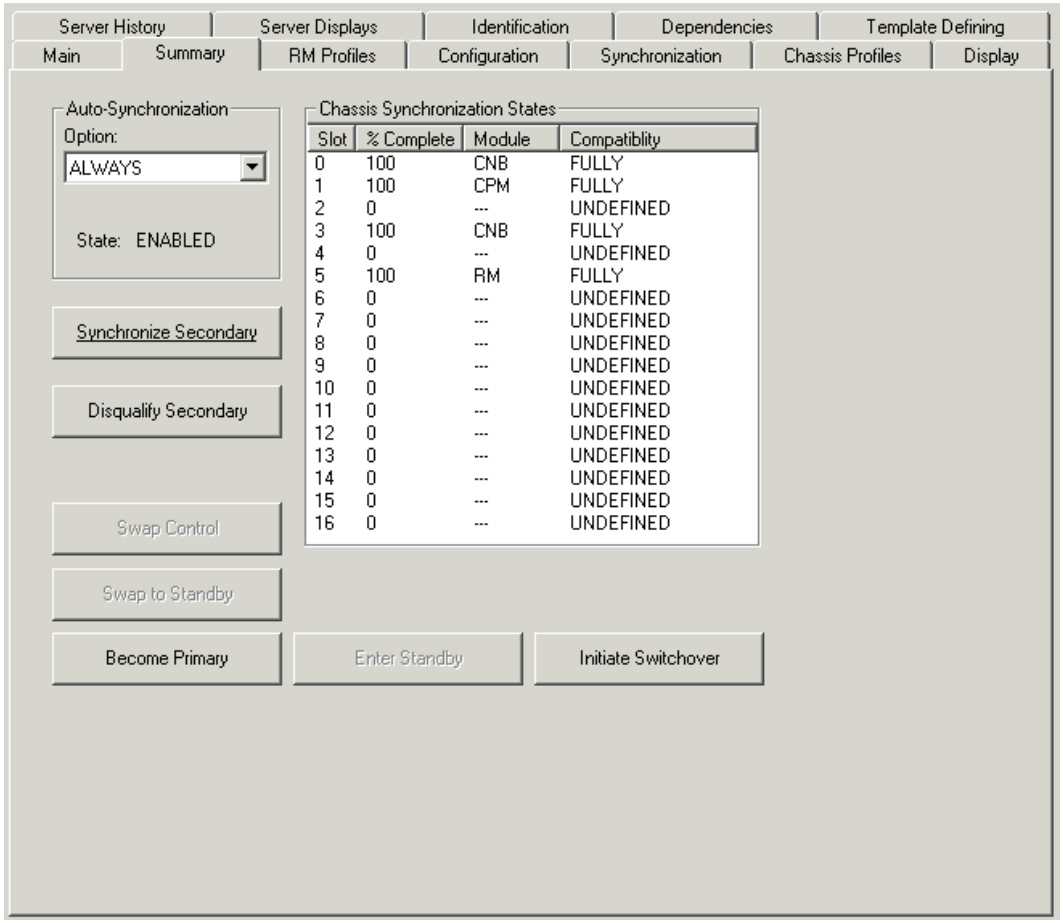
If a switchover occurs, the data in the Primary and Secondary boxes change places.

Main tab

The Main tab includes the following fields that must be initially configured through the Project view to create the Redundancy Module block in Control Builder. The data on this tab cannot be changed in the Monitoring view.

Summary tab

The Summary tab includes the following fields and buttons to provide an overview of synchronization configuration and status, and a means to initiate operator commands.

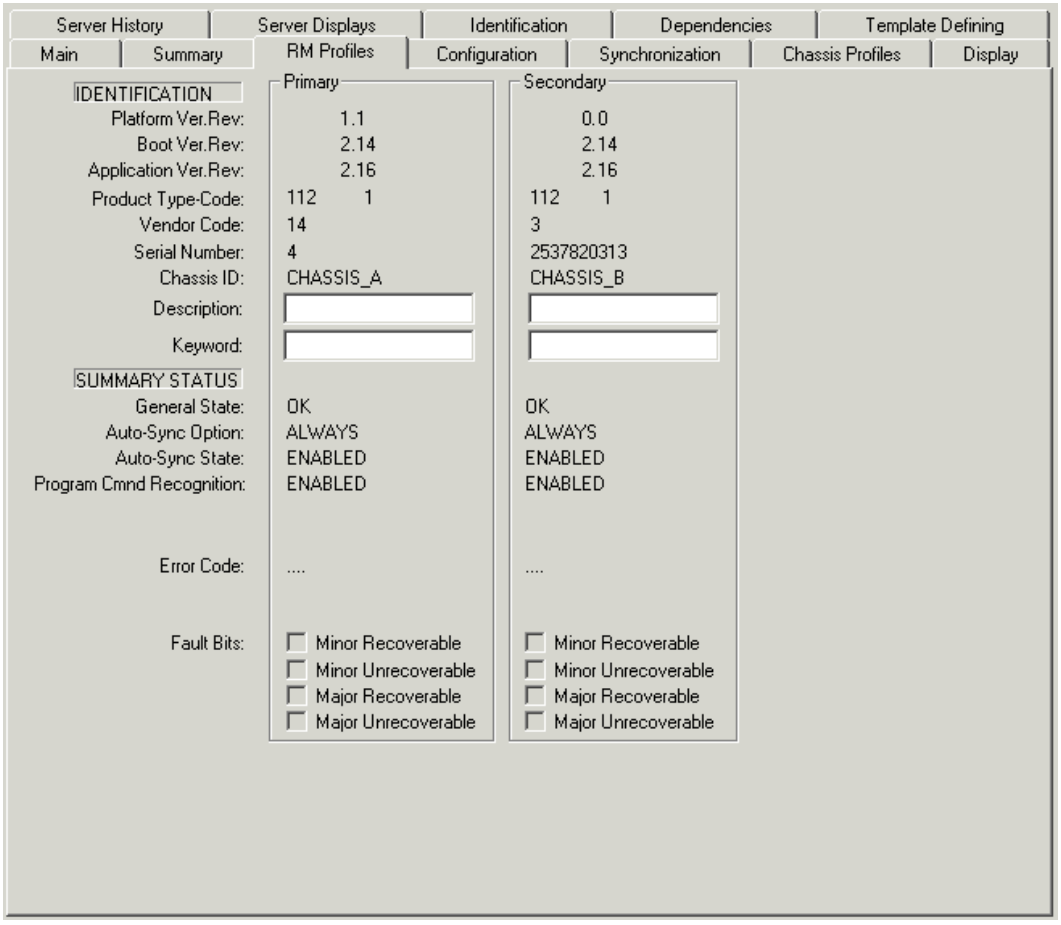


- Auto-Synchronization — This refers to the ability of the RCP to synchronize without user intervention. This is only applicable when the Auto-Synchronization state is Enabled and an Auto-Synchronization event is triggered.
  - Option: Shows the current configuration status and provides these on-line configuration selections.
    ALWAYS - Auto-Synchronization State is always Enabled.
    NEVER - Auto-Synchronization is always Disabled
    CONDITIONAL - Auto-Synchronization is Enabled upon receipt of any valid Initiate Synchronization command; or Disabled upon receipt of any valid Disqualify Secondary command.
  - State: Shows the current Auto-Synchronization state.
    ENABLED: An RM with a Disqualified Secondary will attempt to synchronize the Secondary when it receives any Auto-Synchronization event trigger.
    DISABLED: An RM ignores any Auto-Synchronization event trigger.
- Synchronize Secondary Button — Lets you send a synchronize command to the RM. If a Primary RM with a Disqualified Secondary accepts the command, it attempts to synchronize with its partner.
- Disqualify Secondary Button — Lets you send an abort synchronization command to the RM. If a Primary RM with a Qualified Secondary accepts the command, it aborts synchronization with its partner.
- Swap Control Button — Not active for R120.
- Swap to Standby Button — Not active for R120.
- Become Primary Button — Lets you send a change to Primary command to the RM. If a Disqualified Secondary RM with no partner (Primary not visible) accepts the command, it switches to the Primary state.

- Chassis Synchronization States — Provides a status overview of the synchronization activity on a module by module basis.
  - Slot: Numbered reference to the ICP chassis slot location.
  - % Complete: Shows status of synchronization effort for a module pair as a percentage (0 to 100). Modules synchronize at different rates. For example, a CPM can take as long as a minute to synchronize. The RM always shows 0%, since it does not perform a configuration data exchange during synchronization. The RMs synchronize their configurations at startup (or reconnect) and whenever a configuration change occurs.
  - Module: Identifies the module installed in a given slot with an abbreviation.
    UNK - Unknown
    - - - No module present
    CPM - Control Processor module
    CNI - ControlNet Interface
    RM - Redundancy Module
  - Compatibility: Shows module compatibility with respect to its partner.
    UNDEFINED: No module, no partner, or not yet assessed. (Note that the second slot for a doublewide module is shown as UNDEFINED.)
    INCOMPATIBLE: Modules have detected an incompatibility in one or more of these attributes.
    Vendor ID
    Product Type
    Product Code
    Revision
    FULLY: Modules are compatible. It is possible that minor module differences do not warrant incompatibility. If versions differ, the newer version is programmed to make this determination. .
- Initiate Switchover Button — Lets you send a switchover command to the RM. If a Primary RM with a Qualified (Synchronized) Secondary accepts the command, it immediately initiates the switchover.

RM Profiles tab

The RM Profiles tab includes the following fields to provide an overview of RM attributes, synchronization configuration and general RM status for both the Primary and Secondary RMs.



- IDENTIFICATION — List of attributes that identify the given RM.
  - Platform Ver. Rev: Shows version and revision numbers for the RM hardware.
  - Boot Ver. Rev: Shows version and revision numbers for the RM Boot firmware

- Application Ver. Rev: Shows version and revision numbers for the RM application firmware.
- Product Type-Code: Shows enumeration's of the RM product type (class) and product code (class member).
- Vendor Code: Shows enumeration of the RM manufacturer identification.
- Serial Number: Show unique identification number assigned to given RM.
- Description: Lets you enter a descriptive text string that is used by the Experion Server and visible on all RM alarms.
- Keyword: Lets you enter a definitive text string that is used on Station displays.

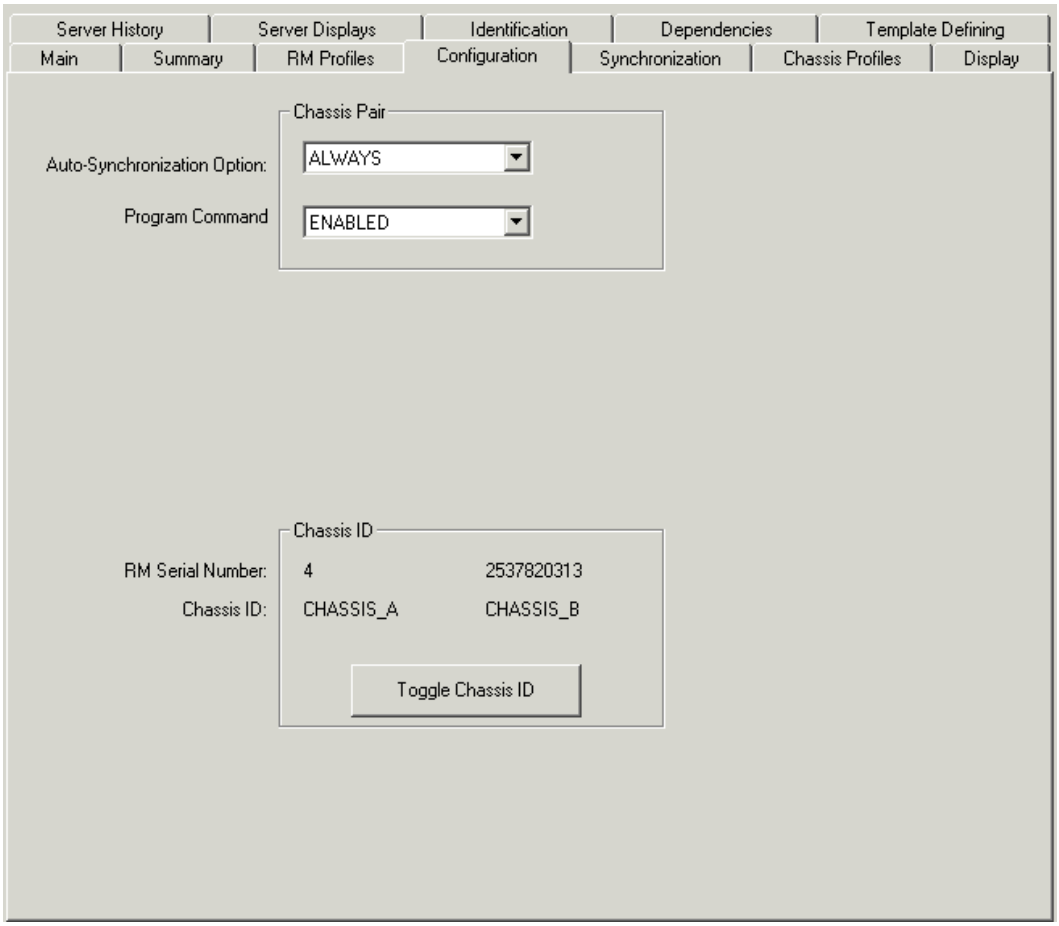
• SUMMARY STATUS — List of items that reflect the general status of the RM.

- General State: Shows the general operational state of the RM.
  UNDEFINED - General state of the RM has not yet been assessed.
  STARTUP - RM in process of startup.
  RELOAD - RM firmware in reload cycle.
  FAULT - RM in fault state.
  OK - RM is fully operational.
- Auto-Sync Option: Shows current Auto-Synchronization option selection - See Summary tab for more details.
  ALWAYS
  NEVER
  CONDITIONAL
- Auto-Sync State: Shows current state for Auto-Synchronization relative to an event trigger - See Summary tab for more details.
  ENABLED
  DISABLED
- Program Cmnd: Shows current configuration selection for allowing the RM to accept program commands from a controller module.
  ENABLED - RM accepts all valid commands from a Control Processor module.
  DISABLED - RM rejects any command from a Control Processor module and generates an error message.
- Error Code: Shows any detected error code in the format EXXX. Where XXX represents an alphanumeric code.

- Fault Bits: Shows status of four severity fault flags in the RM. A checkmark in a box indicates an active fault.

Configuration tab

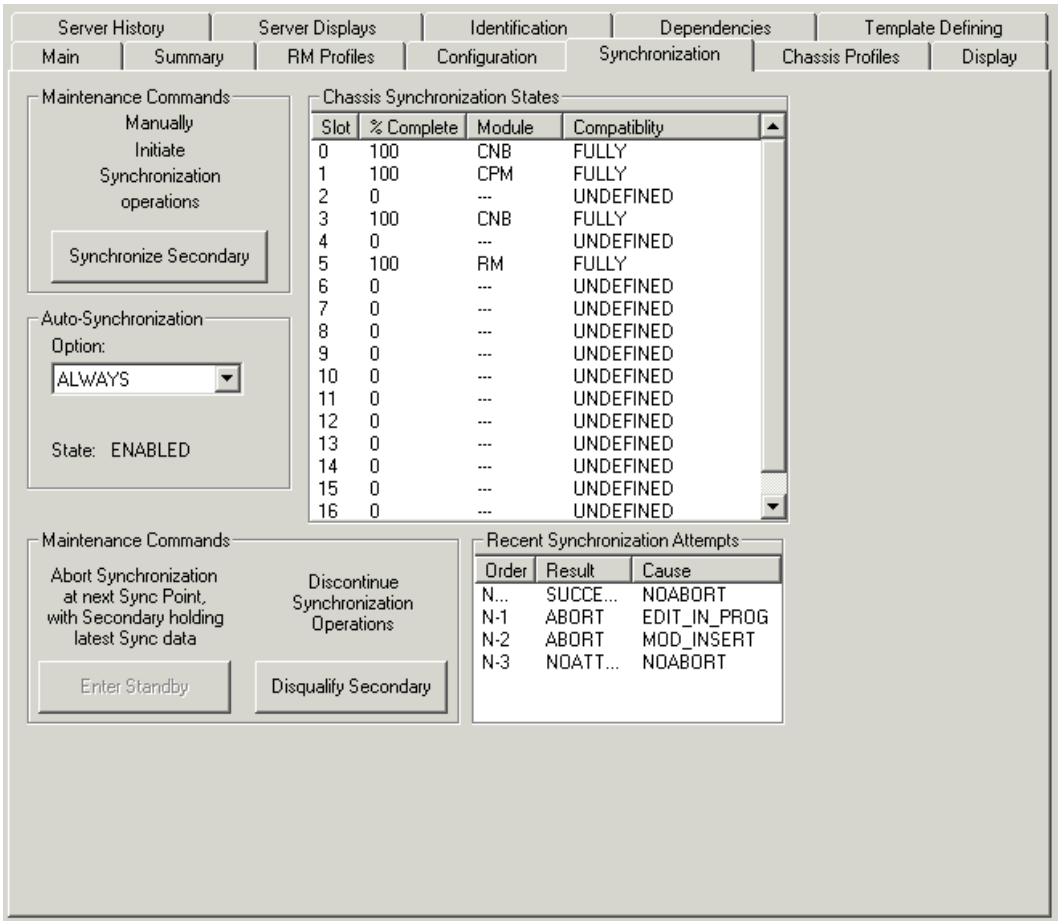
The Configuration tab includes the following fields and buttons for making configuration selections and toggling display positions.



- Chassis Pair — Includes configuration selections for Auto-Synchronization and Program Commands.
  - Auto-Synchronization Option: Shows current option selection and lets you make another selection. See the Summary tab for Auto-Synchronization details.
    ALWAYS
    NEVER
    CONDITIONAL
  - Program Command: Shows current command selection and lets you make another selection. See the RM Profiles tab for Program Command details.
    ENABLED
    DISABLED
- Chassis ID — Shows current chassis ID position relative to Primary (Chassis_A) and Secondary (Chassis_B) for given RM serial number.
  - RM Serial Number: Lists unique identification number assigned to given RM.
  - Chassis ID: Shows display position relative to Primary and Secondary states as Chassis_A and Chassis_B, respectively.
- Toggle Chassis ID Button — Lets you send a Toggle Display command to the RM. It switches the chassis ID labels between two RM serial numbers.

Synchronization tab

The Synchronization tab includes the following fields and buttons to provide a window to synchronization operations and a means to initiate operator commands. Since most of the buttons and display fields are the same as those on the Summary tab, only the Recent Sync Attempts field is described below. Refer to the Summary tab for the other button and field descriptions.



- Recent Synchronization Attempts — Lists the result and cause for the four most recent synchronization attempts.
  - Order: Shows the chronological order of the data as follows.
    N... Most recent data
    N-1 Second most recent data
    N-2 Third most recent data
    N-3 Fourth most recent data
  - Result: Shows the result of the synchronization attempt.
    NOATTEMPT - Indication of an empty location in the Result log. Usually only seen soon after an RM startup, when the RM doesn't have enough Synchronization attempts to fill its log.
    SUCCESS - Indication that attempt resulted in RCP reaching full synchronization.
    ABORT - Indication that attempt was aborted.
  - Cause: Shows a message that identifies the possible cause for the result of the given recent synchronization attempt. The following table lists the cause messages and gives a brief description of each for reference.

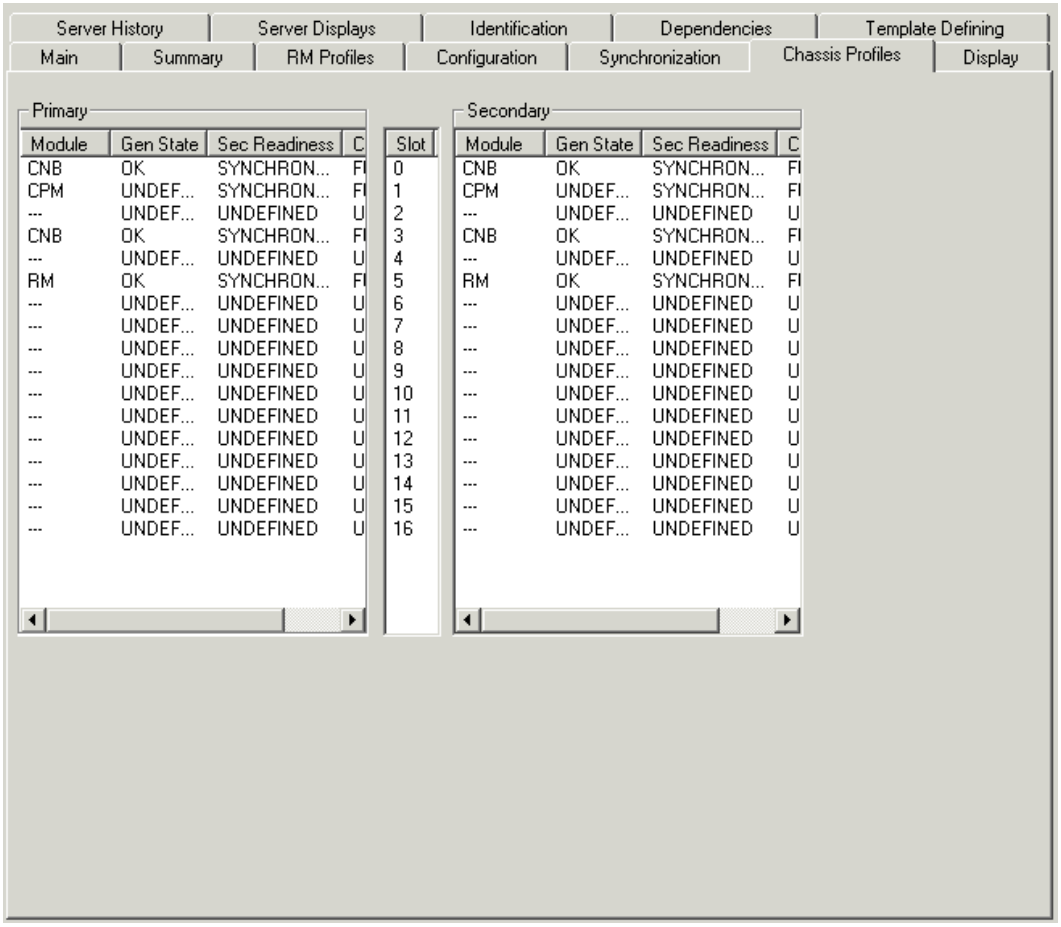
Cause Message	Description
BADSEC_EXIST	Module in Secondary chassis failed prior to the synchronization attempt. Repair or replace the failed module and retry.
COMM_DISCONN	Communications failed (disconnect) during the synchronization attempt. Try the following recovery procedures in the order presented and retry the synchronization attempt after each procedure. • Check redundancy cable installation. • Reseat each module in turn. • Replace each RM in turn. • Replace each module in turn. • Replace each chassis in turn.
COMM_NOEXIST	Communications failed before the synchronization attempt. Try the recovery procedures listed for the COMM_DISCONN message.

Cause Message	Description
CRSSLD_FAIL	The cross load of data from Primary partner modules failed. Try the recovery procedures listed for the COMM_DISCONN message.
EDIT_IN_PROG	An editing session is in progress. Stop configuration operations on all modules and retry the synchronization attempt.
LOC_MAJFLT	Major unrecoverable fault detected for module in the local chassis. Repair or replace the failed module and retry the synchronization attempt.
MODCONF_ERR	Module configuration error. Check module configuration for compliance with redundancy requirements and retry the synchronization attempt.
MOD_INSERT	Module was inserted into the chassis during the synchronization attempt. Check and correct module installation and pairing in the RCP and retry the synchronization attempt.
MODPAIRINCMP	A pair of modules is incompatible. Use other RM/RCP dialog box tab views to identify the incompatible pair. Use the Network Tools application to check module data and update firmware as required. Retry the synchronization attempt.
MOD_REMOVAL	Module was removed from the chassis during the synchronization attempt. Check and correct module installation and pairing in the RCP and retry the synchronization attempt.
NOABORT	Result was either NOATTEMPT or SUCCESS.
NRCMOD_EXIST	A non-redundancy compliant module is installed in the RCP. Use Network Tools application to check model numbers and version data for installed modules to identify non-redundancy module. Replace module with redundancy compliant one and retry the synchronization attempt.
PRTNR_MAJFLT	Major unrecoverable fault detected for the partner module. Repair or replace the failed module and retry the synchronization attempt.
SECMOD_FAIL	Module in Secondary chassis failed. Repair or replace the failed module and retry the synchronization attempt.

Cause Message	Description
SECRM_COMERR	Communications error with the RM in the Secondary chassis. Try the following recovery procedures in the order presented and retry the synchronization attempt after each procedure. • Check redundancy cable installation. • Reseat each RM in turn. • Replace each RM in turn. • Reseat each module in turn. • Replace each module in turn. • Replace each chassis in turn. .
SECRMMAJRFLT	Major recoverable fault detected for the RM in the Secondary chassis. Repair or replace the RM and retry the synchronization attempt.
SECRMMAJUFLT	Major unrecoverable fault detected for the RM in the Secondary chassis. Replace the RM and retry the synchronization attempt.
SECSUBSYSFLT	Secondary subsystem fault.
WRONG_CHS_ST	Chassis is in the wrong redundancy state for synchronization. The Primary chassis must have a Disqualifed Secondary to accept the synchronization attempt. Disqualify the Secondary and retry the synchronization attempt.
WRON_DEV_ST	Module is in the wrong device state. Individual module state does not support redundancy function. For example, the CPM is in its ALIVE state. Change device state as required and retry the synchronization attempt.

Chassis Profiles tab

The Chassis Profiles tab includes the following fields to provide an overview of the RCP status. Since the fields in the Primary and Secondary categories are identical, they are only described once here.



- Primary/Secondary — Provides a status overview of the RCP activity on a module by module basis for the given chassis.
  - Module: Identifies the module installed in a given slot with an abbreviation.
    UNK - Unknown

- - - No module present
CPM - Control Processor module
CNI - ControlNet Interface
RM - Redundancy Module

- Gen State: Shows the general operational state of the given module.
  UNDEFINED - General state of the module has not yet been assessed.
  STARTUP - Module in process of startup.
  RELOAD - Module firmware in reload cycle.
  FAULT - Module in fault state.
  OK - Module is fully operational.

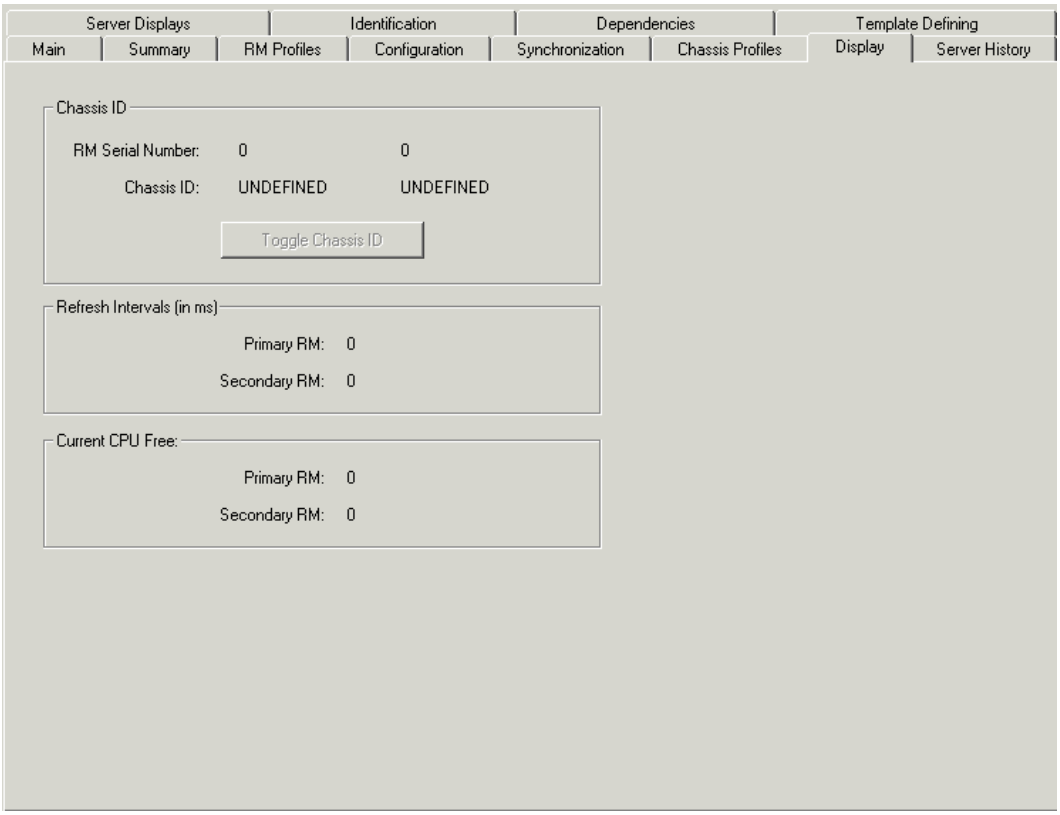
- Sec Readiness — Shows readiness of Secondary chassis to assume the role of the Primary if a switchover occurs. The possible readiness states are:
  UNDEFINED: Insufficient data to assess readiness. This is generally a startup state.
  NOPARTNER: Partner (Secondary) is not visible through RM-to-RM private path and can not assume the role of the Primary.
  DISQUALIFIED: Secondary is present and disqualified for operation as the Primary. It can not assume the role of the Primary.
  SYNCHRONIZED: Secondary is synchronized with the Primary. It can now assume the role of the Primary.
  STANDBY: Secondary is not synchronized but it can assume the role of the Primary. This state is not available in R120.

- Compatible: Shows module compatibility with respect to its partner.
  UNDEFINED: No module, no partner, or not yet assessed. (Note that the second slot for a doublewide module is shown as UNDEFINED.)
  INCOMPATIBLE: Modules have detected an incompatibility in one or more of these attributes.
  Vendor ID
  Product Type
  Product Code
  Revision
  FULLY: Modules are compatible. It is possible that minor module differences do not warrant incompatibility. If versions differ, the newer version is programmed to make this determination. .

Display tab

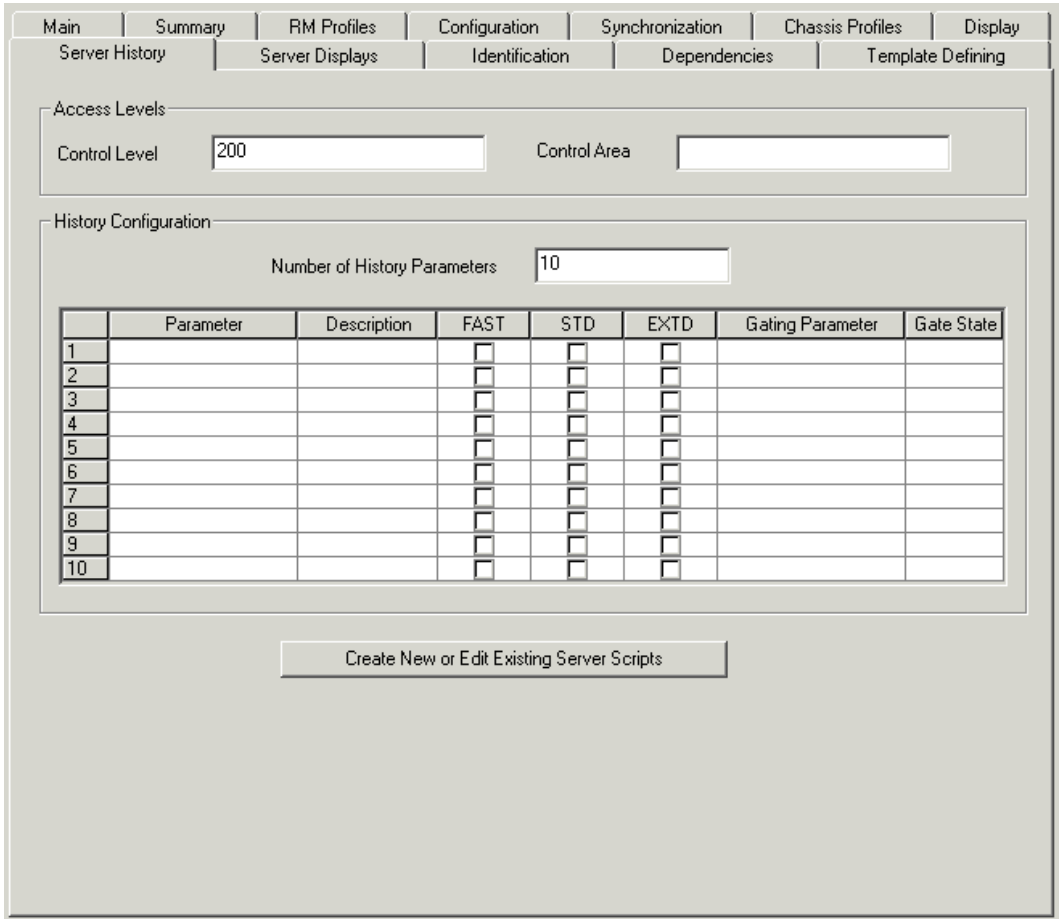
The Display tab includes the following fields and buttons for toggling display positions for the RM/RCP dialog box and checking the refresh intervals for the Primary and Secondary RMs in milliseconds.

Refer to the Configuration tab section for descriptions of the Chassis ID fields and Toggle Chassis ID button.



Sever History tab

The Server History tab includes the following fields for configuring the parameters related to history collection in the Experion Server applications. The entries are the same for both Primary and Secondary chassis.



Control Level - A system wide configuration parameter for Experion Server to limit control over selected display data in Station. A control level can be any number from 0 to 255. An operator can only control a point if their assigned control level is equal or greater than the assigned Control Level. The default value is 200.

Control Area - A system wide configuration parameter for Experion Server to restrict access to data in Station.

Number of History Parameters - Defines the number of history parameters to be included in the History Configuration table. Key in the desired number from 1 to 25 and press <Enter> to initiate changes in the Table.

Parameter - Enter a valid parameter name for a parameter associated with the given point that is to be collected and stored as hisotrical data at predetermined intervals. You can enter up to 25 different parameters for this given point.

Description - This read-only field provides a brief description of the entered parameter.

FAST - Check box lets you select the Fast type of history collection.

STD - Check box lets you select the Standard type of history collection.

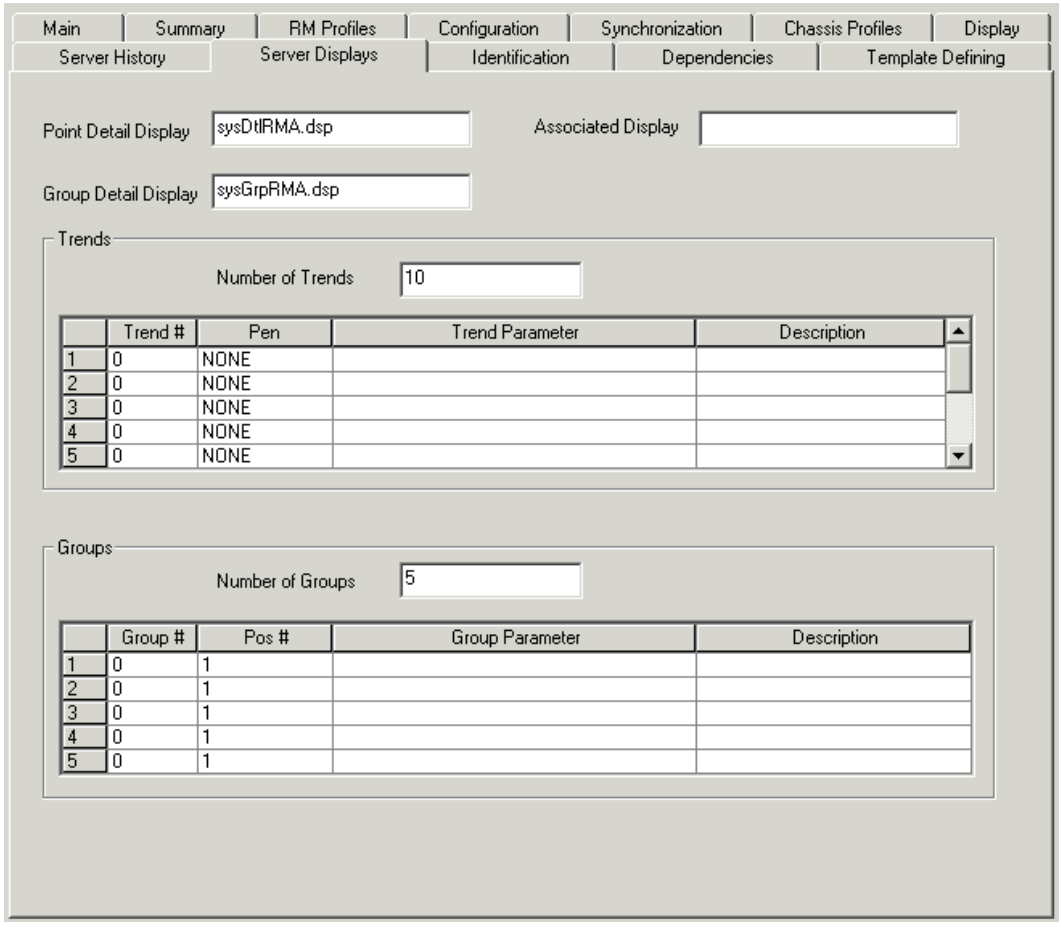
EXTD - Check box lets you select the Extended type of history collection.

Gating Parameter - Enter an optional gating parameter to define the conditions under which data for this parameter should be collected. This field is only active when associated valid parameter and history type are configured.

Gate State - Defines the gate state for the configured gating parameter. This field is only active when a gating parameter is configured.

Sever Displays tab

The Server Displays tab includes the following fields for configuring the parameters related to standard display functions in the Experion Server applications. The entries are the same for both Primary and Secondary chassis.



Point Detail Page — Links the RM data to a Detail display template for Station. The default display file name is sysDtlRMA.dsp

Associated Display — Links a custom schematic created in Display Builder to the RM Detail display. Enter page number or file name assigned to the custom graphic in Display Builder.

Group Detail Page — Links RM faceplates to a Group display template for Station. The default display file name is sysGrpRMA.dsp.

Number of Trends - Defines the number of trend parameters to be included in the Trends Configuration table. Key in the desired number from 1 to 25 and press <Enter> to initiate changes in the Table.

Trend # - Key in the desired Trend # (Number) to be associated with this trend parameter. Trend set displays are standard Station displays that provide a way of viewing historical data for points. Trend set displays complement other types of displays that can be used to view historical data such as point detail trends, group trends, and custom display trends.

Pen - Defines the color of the pen that will be used to trace the assigned parameter on the Station Trend display.

Trend Parameter - Enter a valid parameter name for a parameter associated with the given point that is configured for history collection. When a point parameter has been configured for history collection, it is sampled and stored in the server database history files. The Trend displays show the history data contained in these files. You can include this point as a member of up to 10 trends.

Description - This read-only field provides a brief description of the entered parameter.

Number of Groups - Defines the number of group parameters to be included in the Groups Configuration table. Key in the desired number from 1 to 25 and press <Enter> to initiate changes in the Table.

Group # - Key in the desired Group # (Number) to be associated with this group parameter. Group displays are single Station displays that provide a way of viewing up to eight related points. Each group display can contain a mixture of point types.

Pos - Defines the number of the position the configured parameter will occupy in the Station Group display.

Group Parameter - Enter a valid parameter name for a parameter associated with the given point that is configured in the system. You can include this point as a member of up to 10 groups.

Description - This read-only field provides a brief description of the entered parameter.

Auto-Synchronization events

The following events can trigger an Auto-Synchronization in a Primary RM with a Disqualified Secondary.

- The Secondary chassis becomes disqualified for one of the following reasons:
  - At power up, the Secondary chassis RM finds the Primary chassis RM and becomes a Disqualified Secondary. (Note that this also defines a simultaneous power up of the RCP.)
  - The RMs establish/re-establish an inoperative RM-to-RM connection and disqualify the Secondary as a result. For example, the redundancy cable installed between RMs in a powered RCP.
  - The Primary RM completes a switchover initiated by an Initiate Switchover command, or the insertion of a module into a Primary chassis with a Synchronized Secondary. These types of switchovers result in a Disqualified Secondary.
- Module detected in either chassis. Detection means that the module has completed a portion of its startup process and begins interacting with the RM. Other actions, such as a switchover caused by inserting a module, may precede Auto-Synchronization.
- Module goes invisible in the Secondary chassis.
- All major faults cleared in the Primary chassis.
- The redundancy's backplane hardware failure signal (SYS_FAIL_L) is deactivated. This applies to either chassis, and is probably caused by the removal of a failed module.
- Auto-Synchronization option changes to ALWAYS, with Auto-Synchronization state of DISABLED. This causes the Auto-Synchronization state to change to ENABLED.

Note that all of these events result in only one attempt at synchronization. If the synchronization function aborts, it takes another event trigger (or Secondary command) to initiate another synchronization attempt.

Control Mode Shed on Loss of I/O Functionality

Basic Control Mode Shed Design Concepts

About Control Mode shed on loss of I/O

If your control strategy includes cascaded loops, the ultimate secondary Regulatory Control type blocks will shed their Control Mode to Manual (MAN) upon the loss of I/O communications. This is in addition to going into initialization. The ultimate secondary block must be connected directly to an Analog Output Channel or a Pulse Width Modulator Channel block, so it can detect when communications fail with the Analog Output Module or Digital Output Module with a PWM Channel block.

This functionality applies to the family of Experion system I/O modules including Series A Chassis I/O, Series A Rail I/O, and Series H Rail I/O.

How it works

Any condition that breaks the communication path between the physical I/O module and the I/O function block will initiate a control mode shed. The most common causes of loss communications are as follows:

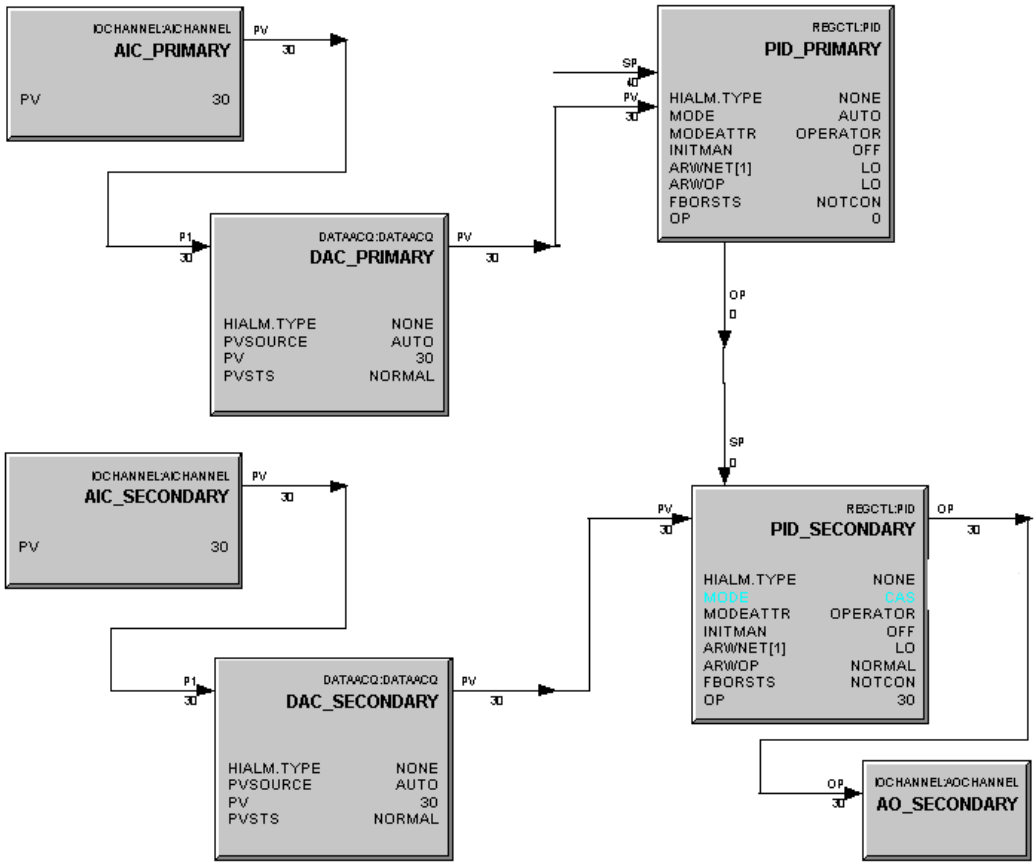
- Removing the I/O module from the chassis.
- Removing the local or remote CNI module from the chassis
- Disconnecting the ControlNet Cable.
- Cycling power to the remote chassis that contains an associated I/O module.
- The diagnostic failure of an individual slot. Only applicable for slot status conditions that truly indicate a broken output path. For example, a “Communication error” indicating a failure in communication to the IOM or an individual hardware failure on a slot. This means that a “Bad calibration” error would **not** initiate a shed (or back initialization for that matter), since it does break the output path.

ATTENTION

These common actions do **not** break the communications path or initiate a control mode shed:

- Activating/inactivating the IOM block.
- Activating/inactivating the CEE that contains the IOM block.
- Activating/inactivating the Control Module that contains an associated AOCHANNEL or PWMCHANNEL block.

For example, if the Analog Output module associated with the AO_SECONDARY block in the typical cascade control strategy shown below is removed from the chassis, the control MODE of the connected PID_SECONDARY block would shed to MAN (Manual). You could observe this action take place through the Monitoring tab in Control Builder or the associated Detail display in Station.



Implications for Operation

Resetting mode after I/O communications are restored.

When I/O communications are restored, the operator must manually reset the secondary regulatory block's control mode back to AUTO (Automatic) or CAS (Cascade) from MAN.

For example, if the AOM removed in the previous example is returned to the chassis, an operator would have to manually change the MODE of the PID_SECONDARY block from MAN to CAS through the Monitoring tab in Control Builder or the Detail display in Station.

Allowing Redundancy synchronization with lost I/O communications

The Control Mode shed on loss of I/O function allows redundant Controllers to synchronize in the presence of lost I/O communications.

For example, if you initiate a switchover in a synchronized Redundant Chassis Pair (RCP), the Secondary Controller becomes the Primary. If you now remove the up-link CNI from the chassis of the Primary Controller, any PID blocks in its control strategy change their control mode to MAN and go into initialization. The Secondary controller is allowed to synchronize with good I/O communications and an automatic switchover occurs making the Secondary the Primary again. The PID blocks in this Primary Controller's matching Control Strategy will now be in MANual mode. Hence, an operator would have to manually reset any secondary regulatory blocks that have shed to MAN mode back to AUTO or CAS mode.

Remote Cascade function between Controllers

The REMCAS (Remote Cascade) block provides automatic switching between a remote and backup cascade. This block is typically used with a PID block that normally gets its set point from a remote source, but sheds to a local source if there is a communications failure.

Note that when using a cascade control strategy between two controllers, or a controller and an ACE node, you do not need to use the REMCAS block but switching to a backup primary is not supported. The Regulatory Control (REGCTL) blocks in Experion do not contain the remote cascade function so the REMCAS block must be included in the control strategy to provide the function.

Use a REMCAS block between the controller/ACE nodes to handle the initialization and handshaking required to provide switching between remote and local sources

See the following links for more detailed information about the REMCAS block and its usage.

- REMCAS Block
- Peer-to-peer configuration example

Control Builder Export and Import Functionality

Basic Export/Import Design Concepts

About Control Builder Export and Import functions

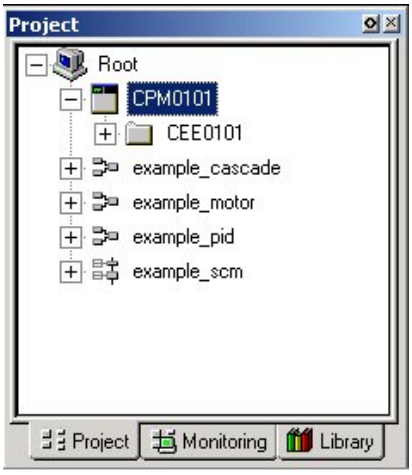
Control Builder includes Export and Import functions that are accessible through its File pull-down menu. These functions provide a dynamic copy and paste type functionality for the Engineering Repository Database (ERDB), which contains the control strategy that was configured through Control Builder.

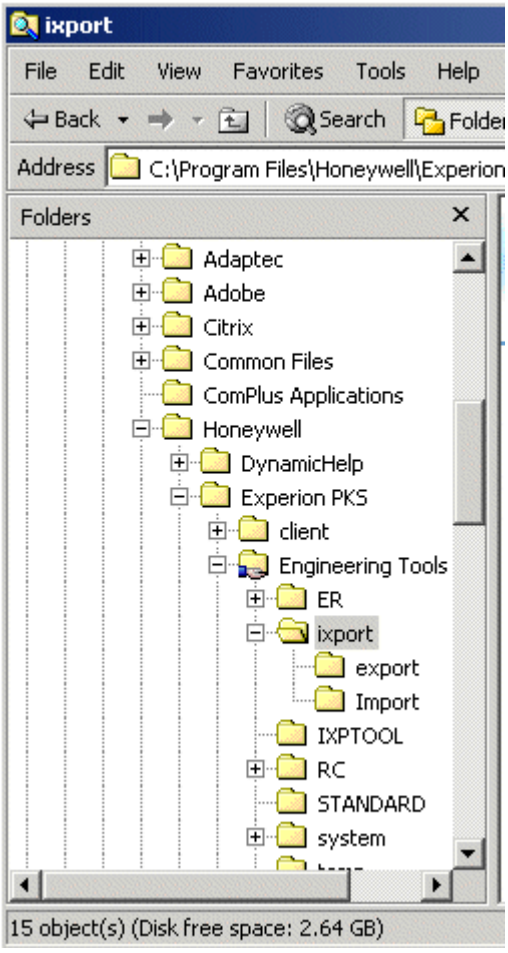
Export functionality

It may help to think of the Export function as a dynamic copy operation for the ERDB being accessed through Control Builder. This function lets you export a portion of the ERDB or the whole ERDB as viewed through the Project Tree in Control Builder. The exported or copied portion of the ERDB is automatically stored in the *c:\Program Files\honeywell\Experion\Engineering Tools\Ixport* directory location by default. The following figure shows how an exported ERDB viewed in the Project Tree appears in the Ixport file folder for example purposes only.

The export function lets you export all or a portion of the ERDB viewed in the Project Tree to the Ixport folder or a user-defined location.

The exported portions or all of the ERDB are stored in the Ixport folder at directory location *c:\Program Files\honeywell\Experion\Engineering Tools* by default.

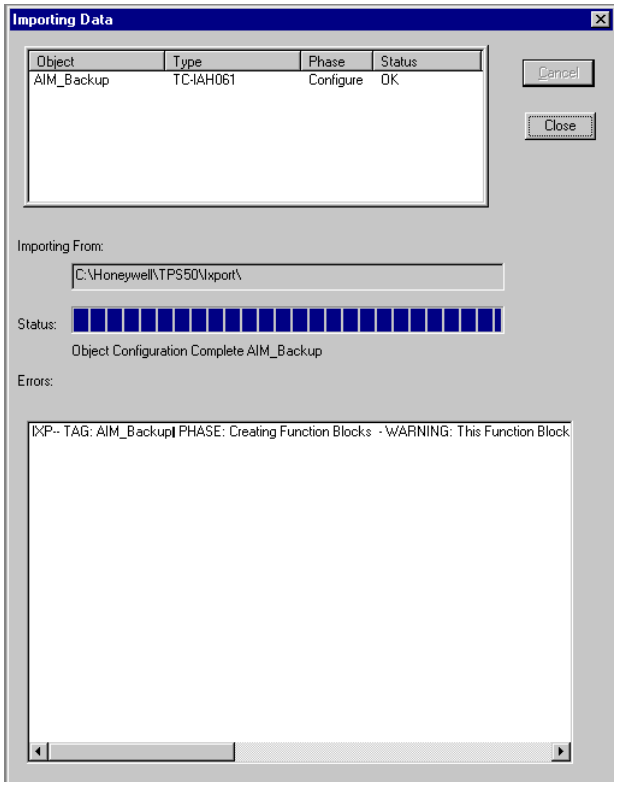




By default, the ERDB used by Control Builder is stored in SQL Server under the filename ps_erdb (default name). This is the ERDB that is being viewed in the Project Tree.

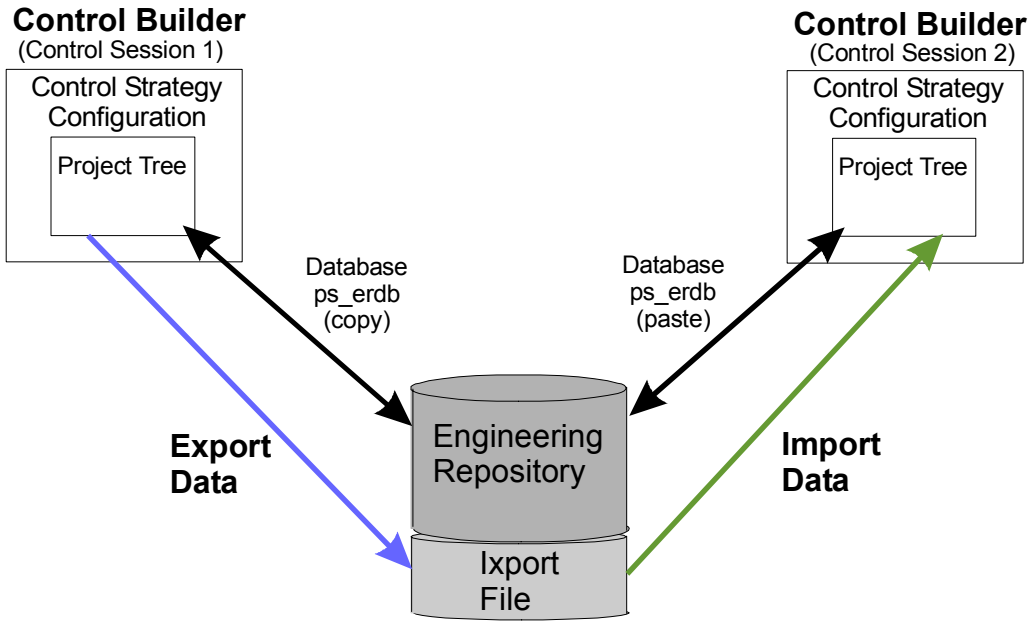
Import functionality

It may help to think of the Import function as a dynamic paste operation for the ERDB being accessed through Control Builder. This function lets you import a portion of an exported ERDB or the whole exported ERDB as viewed through the Import dialog box. The imported or pasted portion of the exported ERDB is automatically written to the Project Tree, which is associated with the **ps_erdb** file in the *SQL Server*. If an imported block has the same name as one already in the Project Tree, the configuration data for the same named block in the Project Tree will be overwritten by configuration data for the imported block. The following figure shows the informational type messages generated for the Import of an exported block named AIM_Backup into the Project Tree for example purposes only.



Export/Import functional overview

The Control Builder Export and Import functions work together to copy/export configuration data from one ERDB and paste/import the exported data to another compatible ERDB in Control Builder. This scenario involves storing the exported data in a designated file location on the hard drive, associating the Control Builder with another ps_erdb file, and importing desired exported configuration data to a new ERDB file through the Import dialog box.



Some Operation Considerations

Export/Import usage notes

The following notes outline some general considerations associated with using the Export or Import function.

- It is always a good idea to make a backup copy of your primary ERDB in the Er folder. The default backup filename is ps_erdb.bak.
- Always use the primary ERDB for export or import functions, when using Redundant Servers.
- Can not initiate an export or import function while Engineering Repository (ER) synchronization is in progress for Redundant Servers.
- Can not run an export and import function at the same time.
- Can not run more than one import function at a time.
- Can not run more than one export function at a time.
- Do not change a Control Session while an import or export function is running.

SCM and CM Chart Visualization Functionality

Basic Chart Visualization Design Concepts

About Chart Visualization

A SCM chart configured and loaded to the Controller through Control Builder is now viewable through the corresponding SCM Detail display in Station. You can also view a CM chart configured and loaded to the Controller through Control Builder in its corresponding CM Detail display in Station.

This chart visualization function uses ActiveX® technology to embed an active document in the SCM or CM Detail display. This means an operator can actively monitor an active SCM or CM chart through the corresponding Detail display in Station.

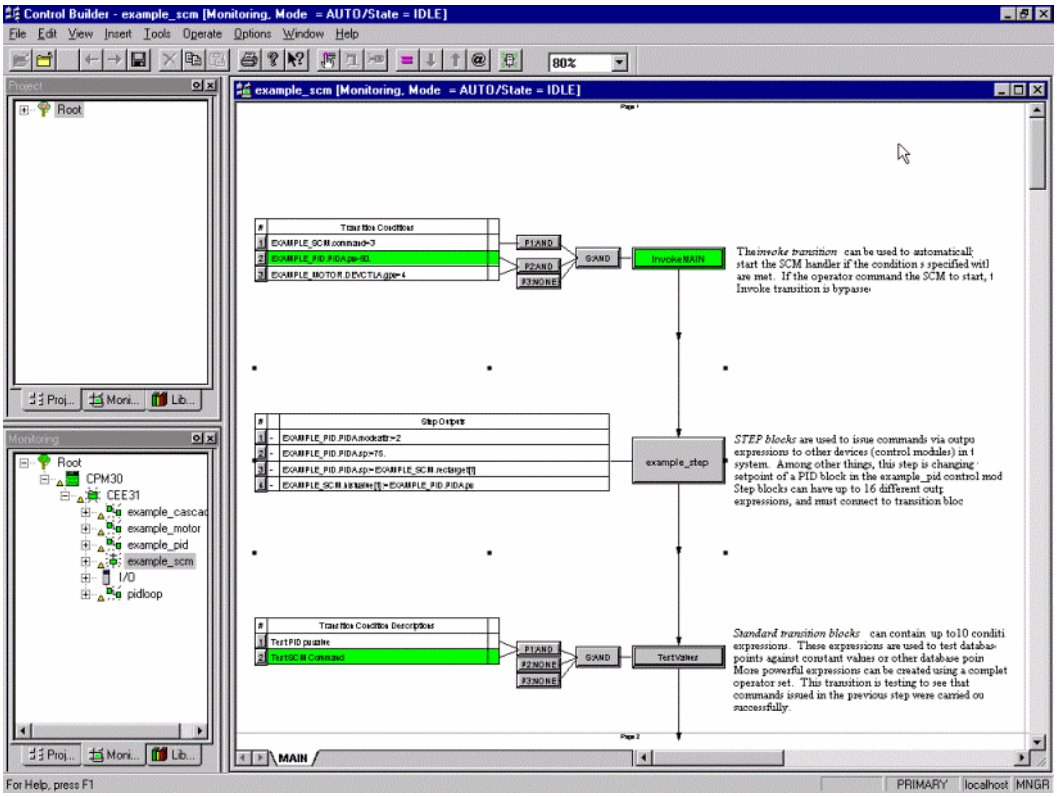
You must have both Engineering Tools and Server/Client software components installed on the computer for the chart visualization function to work.

How SCM chart visualization works

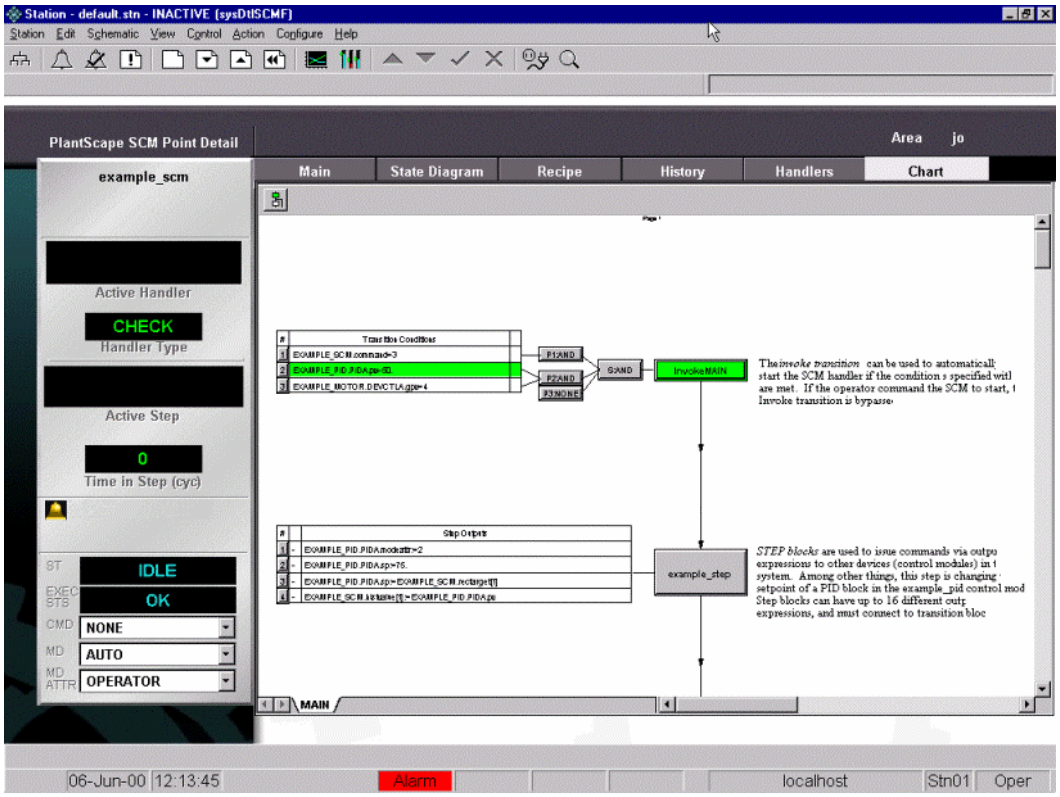
The ActiveX Controls needed to support the display and interaction of a Control Builder SCM or CM chart in the associated Detail Display in Station are loaded as part of the software. The software also includes updated SCM and CM Detail displays The SCM display includes a Tab labeled "Chart". The redesigned CM display includes a Tab labeled "Control Module"

Since the SCM and CM Detail displays mirror the SCM and CM chart configurations in Control Builder, you first configure the SCM and CM charts in Control Builder as usual. Once the SCM and CM charts are configured and loaded to a Controller, you can view them in the Monitoring Tab of Control Builder or in the associated Detail display in Station.

The following figures show how a configured SCM chart named example_scm appears in the Monitoring Tab of Control Builder and in the Chart Tab of the example_scm Detail display in Station for example purposes only.

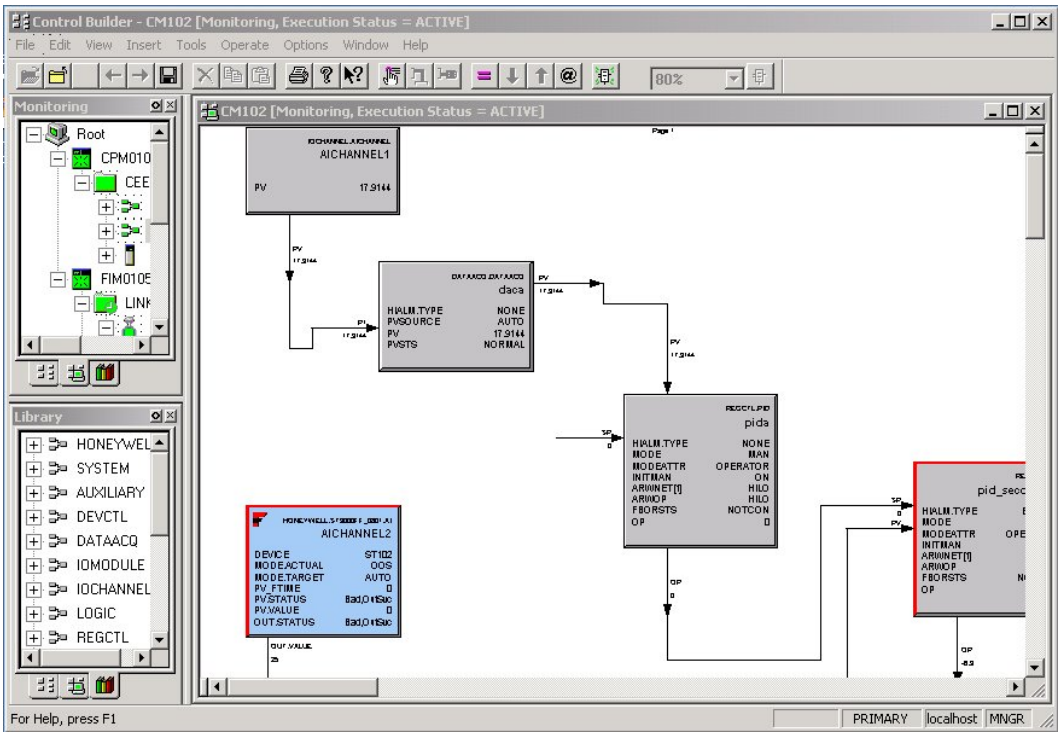


View of example_scm chart in Control Builder Monitoring Tab.

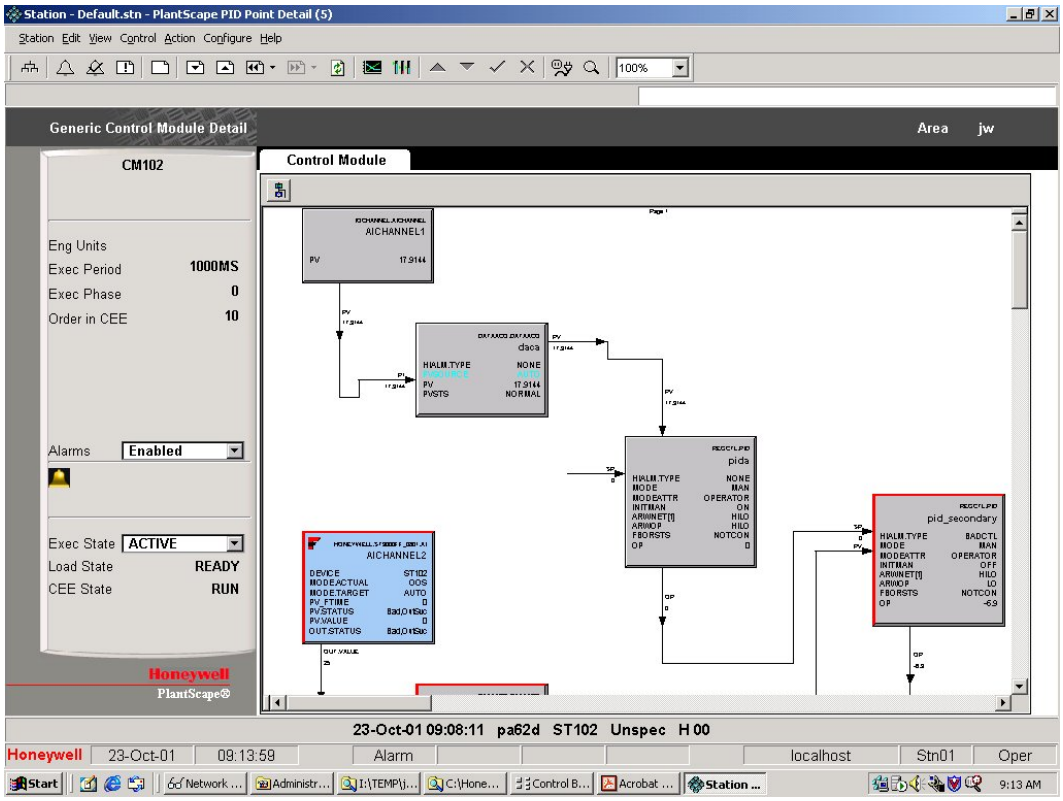


View of example_scm chart in SCM Detail display Chart Tab in Station.

The following figures show how a configured CM chart named CM102 appears in the Monitoring Tab of Control Builder and in the Control Module Tab of the CM102 Detail display in Station for example purposes only.



View of CM102 chart in Control Builder Monitoring Tab.



View of CM102 chart in CM Detail display in Station.

Some SCM and CM Chart Operation Considerations

Detail display interaction notes

ATTENTION

You can only monitor parameters shown on SCM and CM charts in Detail Displays. You must access charts through the Monitoring mode in Control Builder to initiate allowable parameter changes.

- A tool button on the chart Detail display lets you change the scale factor of the chart for viewing as well as cancel or resume the Chart Automatic Tracking feature for an SCM chart display.
- Open only one SCM or CM chart for display in Station at a time.
- You can display the same SCM or CM chart on multiple Stations at the same time.
- You can display different SCM or CM charts on multiple Stations at the same time.
- Since the operator security level can be changed through Station, it is possible that the security level for Station will be different than the security level for Control Builder.
- A communications failure will result in question marks (?) being shown in place of live values until the fault is cleared.

I/O Link Interface

I/O Link Interface

Seamless integration

The I/O Link supports both the:

C200 controller: where the I/O Interface Module (IOLIM) seamlessly integrates the Process Manager I/O subsystem consisting of Input/Output Processor card file, Input/Output Processors (IOPs) and power supply with an Experion system. It lets user easily configure IOPs as part of a control strategy through the system's Control Builder application.

The IOLINK connection for process Manager I/O can be established through C200 controller and IOLIM or C300 controller.

C300 controller: where the IOLINK contains a function that enables programming and reprogramming the executable image (rather than substitution of a removable hardware component).

PM I/O and Series C I/O cannot exist on the same link.

I/O Functions

The function of the IOPs remains the same.

Process Manager I/O: in conjunction with Field Termination Assemblies (FTAs), the IOPs perform input and output scanning and processing for field I/O.

Series C I/O: in conjunction with Input/Output Termination Assemblies (IOTSs), the IOMs perform input and output scanning and processing for field I/O.

A redundant I/O Link is standard for added security.

A word about Point form

A configurable parameter called Point Form (PNTFORM) let users decide whether an IOP point was to be a **Full** or **Component** type point in a Process Manager environment. A **Full** point form meant that the point would have alarm-related parameters. A **Component** point meant that the point should be used as an input to the **Full** point, and also for those points that handle the outputs from the **Full** points. While the PNTFORM parameter does appear on the IOP configuration forms in Control Builder, the default selection for most IOPs is **Component** and it is not user configurable.

IOP validation

Database security is provided to prevent an operator form starting an IOP that has an invalid database. After initial configuration each IOP must be set valid. Users with an engineer access level or higher can initiate the validation check through the Validate IOP Database button on the Control Builder configuration form for an IOP in the Monitoring mode. The IOP must have a valid database before it can be set to RUN.

I/O Link Cable Status and Statistics

Periodic swap functionality

When the periodic channel swap function is enabled, it swaps the active receive channel at one minute intervals to monitor its health and update its status as follows.

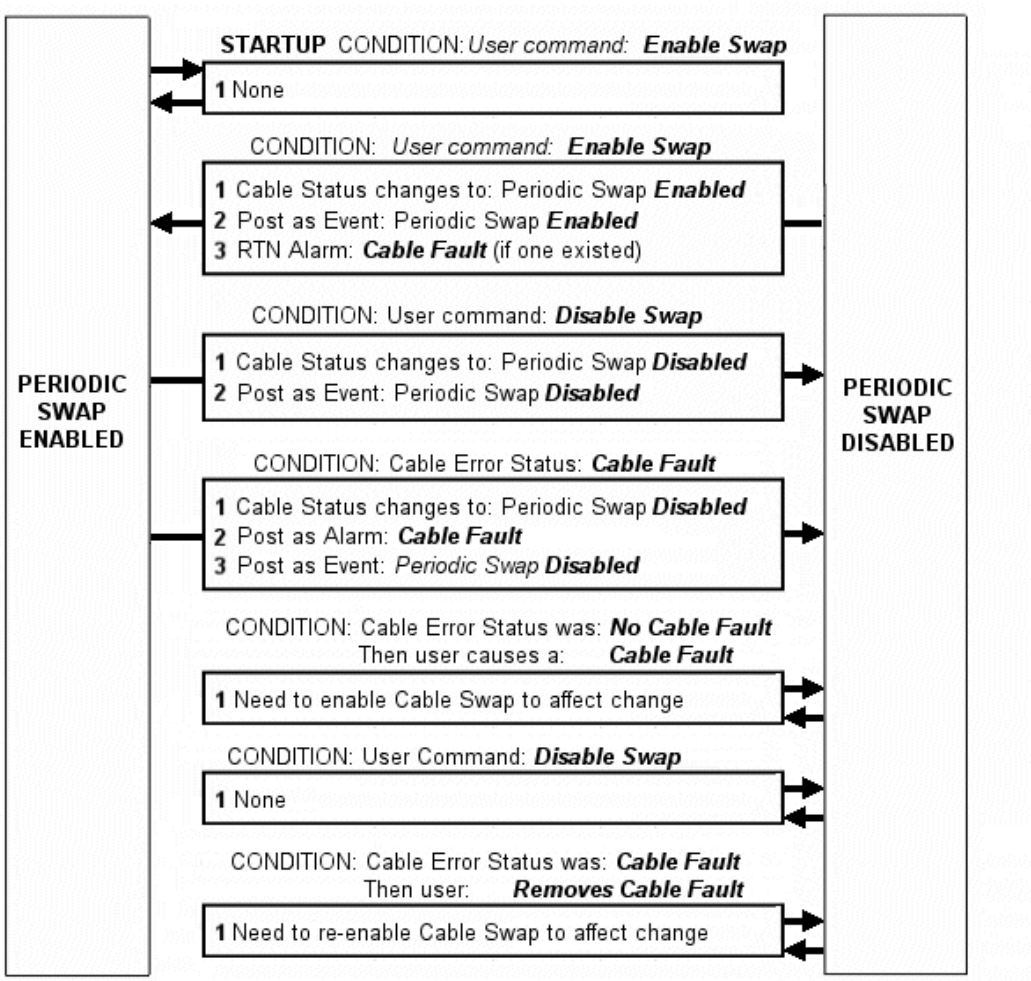
If Status is . . .	Then, Cable is . . .
Ok	Good. No errors detected during last pass of periodic channel swap.
Warning	Questionable. Some cable errors detected during last pass of periodic channel swap. The errors detected did not exceed the periodic swap threshold setting. The default setting for the PERSWAPTHRES parameter is 10 errors per minute.
Error	Bad. Either the cable errors detected during the last pass of periodic channel swap or the sum of errors detected for the last 10 passes of periofic channel swap on this cable exceeds the periodic swap threshold setting of 10 errors per minute.

When the status of a cable transistions to the **error** state, a cable eror alarm is generated and the periodic channel swap function is automatically disabled. The status of the other cable changes to **unknown**, since cable statuses cannot be validated until the periodic channel swap function is re-enabled.

Once the cable fault is corrected, the periodic channel swap is enabled, and the cable status returns to Ok, the cable status alarm returns to normal.

Periodic swap transition diagram

The following diagram illustrates the states and conditions associated with the periodic swap behavior and its associated I/O Link Interface Module annunciations.



To enable periodic channel swap

Use one of the following procedures to issue the enable command to the entire associated I/O Link or just the selected IOP. These procedures assume that Control Builder is running and the control strategy is loaded.

For entire I/O Link

Step	Action
1	In the **Monitoring** tab, expand the hierarchy for the applicable IOLIM block and double click the IOLINK icon to open its configuration form.
2	On the **Main** tab, click the arrow button in the **COMMAND** box and select **ENB PRESWAP**.
3	Click the **Yes** button to confirm the action.
4	Click the **OK** button to close the form.
5	This completes the procedure.

For selected IOP

The SWAP PRIMARY command is only allowed when the IOP partners are synchronized, the primary is connected to a Field Termination Assembly (FTA) and the secondary is connected to a FTA.

Step	Action
1	In the **Monitoring** tab, expand the hierarchy for the applicable IOLIM block and double click the IOLINK icon to open its configuration form.
2	Click the **I/O Status Summary** tab to open it.
3	Scroll the IOP/IOM Number grid to locate the applicable IOP/IOM number. Click the appropriate row in the **IOM Commands** column. Click the arrow button in the row and select **SWAP PRIMARY**.
4	Click the **Yes** button to confirm the action.
5	Click the **OK** button to close the form.
6	This completes the procedure.

Viewing cable status and statistics

You can view a variety of cable status data and statistics through the tabs on the I/O Link block configuration form in the Control Builder **Monitoring** tab. This procedure assumes that Control Builder is running and the control strategy is loaded.

To view channel/cable status and statistics parameters

Step	Action
1	In the **Monitoring** tab, expand the hierarchy for the applicable IOLIM block and double click the IOLINK icon to open its configuration form.
2	Click the **Statistics** tab to check various link interface and performance related parameters. Click the **Show Parameter Names** check box in the lower left-hand corner of the form to display the actual parameter name associated with a field. Refer to the *Control Builder Parameter Reference* document to find a definition for a given parameter.
3	Repeat Step 2 for the **I/O Link Status** and **I/O Status Summary** tabs.
4	Click the **OK** button to close the form.
5	This completes the procedure.

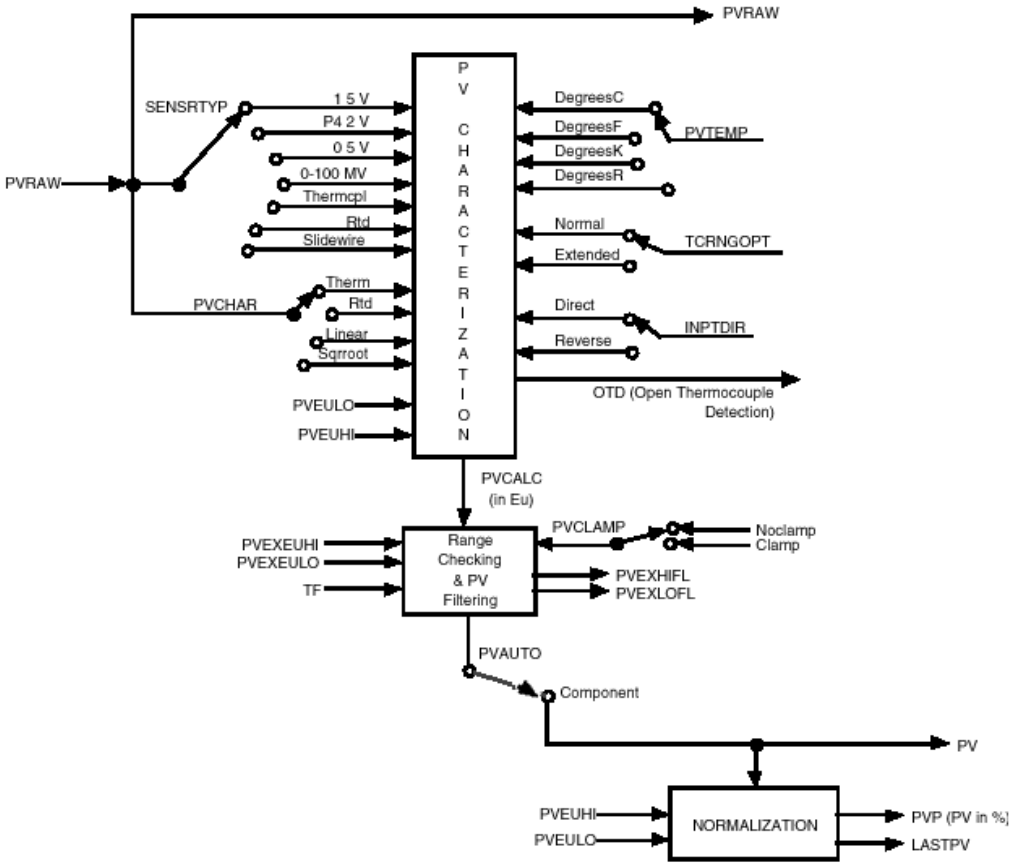
Process Manager Input/Output Functionality

High and Low Level Analog Input Points

Function

The analog input point converts an analog PV signal received from a field sensor to engineering units for use by other data points in the control strategy, as shown in the following figure. To accomplish this function, the analog input point performs the following functions.

- Analog-to Digital Conversion
- PV Characterization
- Range Checking and PV Filtering
- PV Source Selection
- Alarm Detection



High-level points are located in the High Level Analog Input (HLAI and HLAIHART) IOP. One type of low level point is located in the Low Level Analog Input (LLAI) IOP. This type is generally used for control points. The other type is located in either the Low Level Multiplexer (LLMUX) or the Remote Hardened Multiplexer (RHMUX) IOP. This type is generally used for data acquisition points. The type of analog input point needed is based on the type of field sensor that is providing the input to the point and the characterization options selected by the user as listed in following table.

Sensor Type (SENSRTYP)	AI Processor Type	PVCHAR Options	PVRAW(1)	PVCALC	Bad PV Direction(2)
0 to 5 Volts	HL and LL HL and LL HL HL	Linear Square Root Thermocouple RTD	Percent	EU	Range check on PVCALC
0.4 to 2 Volts	HL and LL HL and LL HL HL	Linear Square Root Thermocouple RTD	Percent	EU	Range check on PVCALC HLAI checks for open input
1 to 5 Volts	HL and LL HL and LL HL HL	Linear Square Root Thermocouple RTD	Percent	EU	Range check on PVCALC HLAI checks for open input
Slidewire	HL	Linear	Raito	EU(3)	Bad slidewire source, range check on PVCALC
0 to 100 mV	LL, LLMUX, RHMUX	Linear	Millivolts	EU(3)	Range check on PVCALC
Thermo-couple	LL, LLMUX, RHMUX	Thermocouple	Microvolts	EU	Open thermocouple, and range check on PVCALC
RTD	LL, LLMUX	RTD	Milliohms	EU	Range check on PVCALC

Sensor Type (SENSRTYP)	AI Processor Type	PVCHAR Options	PVRAW(1)	PVCALC	Bad PV Direction(2)

Legend: EU = Engineering Units
HL = High Level Analog Input, (Includes HLAIHART)
LL = Low Level Analog Input
LLMUX = Low Level Analog Multiplexer Input
PVCALC = Calculated Process Variable
PVCHAR = Process Variable Characterization
PVRAW = Process Variable received from field and converted to digital form by the A/D converter.
RHMUX = Remote Hardened Analog Input Multiplexer.

Notes:

e) PVRAW is the voltage signal at the IOP Field Termination Assembly as a percentage of the voltage range for the sensor type. The exceptions are as follows:

1) For a thermocouple sensor type, PVRAW is in microvolts after reference junction compensation. If an open thermocouple is detected, PVRAW is set to NaN.

1) For an RTD sensor type, PVRAW is in milliohms after lead-wire compensation. If an open RTD * is detected, PVRAW is set to NaN.

1) An external power source is used to excite the slidewire. The power source and the slidewire are connected to separate analog input points. One power source input can be used with several slidewire inputs.

For slidewire input, PVRAW is the slidewire ratio (Vin/Vsrc).
Where: Vin is the FTA voltage input for this data point (slot)
Vsrc is the FTA voltage source at the slidewire source slot specified by
parameter SLWSRCID. If Vsrc is zero (fails the under-range check), PVRAW
is set to NaN.

1) For a 0 to 100 millivolt sensor type, PVRAW is the FTA voltage input for the slot.

f) If the diagnostics determine that the A/D converter has failed, PVRAW of the slot is set to NaN.

g) The normal operating range for PVRAW is configured by the user (for a thermocouple, 0 percent equals PVRAWLO and 100 percent equals PVRAWHI; for a slidewire, 0 equals PVRAWLO and 1 equals PVRAWHI).

* Note that LLMUX FTAs provide three wire terminations for each sensor. Open wire detection for RTDs is limited in that the FTA will detect an open wire between the "A" and "C" leads, but not the "B" lead.

PV Characterization

The PV signal received from the field is characterized based on the entries that the user makes for the SENSRTYP, PVCHAR, PVTEMP, INPTDIR, and TCRNGOPT parameters as shown in the previous figure. The input PV signal is first converted to a raw PVsignal (PVRAW) whose units can be percent, ratio, millivolts, microvolts, or milliohms depending on the entry made for the SENSRTYP parameter. The PVRAW signal is then converted to the engineering units. The engineering unit conversions that are performed in the HLAI, HLAIHART, LLAI and LLMUX points are listed in the previous table and described in the following sections.

TIP

The LLMUX points include points built against the LLMUX IOP as well as the RHMUX IOP. The RHMUX does not have a unique point type.

The RHMUX IOP does not support RTD.

Linear Conversions

The PVRAW value is converted to a floating-point number the output value of the linear conversion is PVCALC, which is calculated based on the raw input span (for slidewire and 0-100 mV sensor types only), and the engineering unit span. The state of the input direction parameter (INPTDIR) is taken into consideration during the calculation of PVCALC as follows:

- For slidewire and 0 to 100 millivolts sensor types, when INPTDIR is Direct:

$$PVCALC = \frac{(PVRAW - PVRAWLO)}{(PVRAWHI - PVRAWLO)} * (PVEUHI - PVEULO) + PVEULO$$

- For 0 to 5 Volts, 0.4 to 2 Volts, and 1 to 5 Volts sensor types, when INPTDIR is Direct:

$$PVCALC = \frac{PVRAW}{100} * (PVEUHI - PVEULO) + PVEULO$$

- For slidewire and 0 to 100 millivolts sensor types, when INPTDIR is Reverse:

$$PVCALC = PVEUHI - \frac{(PVRAW - PVRAWLO)}{(PVRAWHI - PVRAWLO)} * (PVEUHI - PVEULO)$$

- For 0 to 5 Volts, 0.4 to 2 Volts, and 1 to 5 Volts sensor types, when INPTDIR is Reverse:

$$PVCALC = PVEUHI - \frac{PVRAW}{100} * (PVEUHI - PVEULO)$$

Square root conversion

The square-root calculation is applied to the PVRAW input such that 100 percent of span equals 1.0 . The square-rooted value is then converted to Engineering Units based on the configured PV engineering-unit range values. (For example, square root of 100 percent equals 100 percent; square root of 50 percent equals 70.71percent.) The output value of the square-root conversion is PVCALC, which is calculated based on the state of the input direction parameter (INPTDIR) as follows:

- If PVRAW is equal or greater than 0.0 and INPTDIR is Direct:

$$PVCALC = \sqrt{\frac{PVRAW}{100}} * (PVEUHI - PVEULO) + PVEULO$$

- If PVRAW is less than 0.0 and INPTDIR is Direct:

$$PVCALC = -\sqrt{\frac{PVRAW}{100}} * (PVEUHI - PVEULO) + PVEULO$$

- If PVRAW is equal or greater than 0.0 and INPTDIR is Reverse:

$$PVCALC = PVEUHI - \sqrt{\frac{-PVRAW}{100}} * (PVEUHI - PVEULO)$$

- If PVRAW is less than 0.0 and INPTDIR is Reverse:

$$PVCALC = PVEUHI - \sqrt{\frac{-PVRAW}{100}} * (PVEUHI - PVEULO)$$

Thermal conversion

Thermal linearization is performed on thermocouple and RTD input types, and is selectable by parameter PVCHAR.

The range of the thermocouple type used with the LLAI or LLMUX points can be increased by selecting Extended as the entry for the TCRNGOPT parameter. The LLAI and LLMUX points calculate the reference junction compensation from the measured reference junction output level. This value is stored and then later converted back to microvolts, with respect to 0 degrees C, for each thermocouple that is to be compensated. The cold-junction reference compensation (RJTEMP) parameter is expressed in microvolts for the specified thermocouple and is added to the microvolt value for PVRAW.

REFERENCE - INTERNAL

Refer to *PVEXEULO* for normal and extended range details.

For an RTD, the LLAI and LLMUX points calculate the lead-wire compensation and then subtract the value from PVRAW. The maximum allowable lead-wire resistance and intrinsic safety barrier resistance for the RTDs are listed in the following table.

ATTENTION

Proper compensation for lead-wire resistance depends on the resistance being equal in each leg of the RTD. This includes resistance due to lead-wire resistance and intrinsic safety barriers. No provision is made to compensate for lead-wire resistance mismatch or intrinsic-safety-barrier resistance mismatch. Both the lead resistance and the intrinsic-safety-barrier resistance are allowed simultaneously when connected to an RTD in a Division 1 area.

RTD Type	Maximum Allowable Lead Resistance		Maximum Allowable Intrinsic Safety Barrier Resistance	
	Entire Loop	***Per Leg***	***Entire Loop***	***Per Leg***
Pt: 100 ohm DIN Characterization	20 ohms	10 ohms	18 ohms	18 ohms
Pt: 100 ohm JIS Characterization	20 ohms	10 ohms	18 ohms	18 ohms
NI: 120 ohm Edison Type 7 Characterization	20 ohms	10 ohms	18 ohms	18 ohms
Cu: 10 ohm SEER Standard Characterization	20 ohms	10 ohms	0.0 ohm	0.0 ohm

Smart Transmitter Interface Point

Smart Transmitter support

The Smart Transmitter Interface (STI) point provides an interface to Honeywell's family of Smart Transmitters. The STI point can support the following Smart Transmitter types:

- ST3000 Smart Pressure Transmitter for differential, gauge, and absolute pressure measurements
- STT3000 Smart Temperature Transmitter for temperature, millivolts, and ohms measurements, and
- MagneW 3000 Smart Magnetic Flow Transmitter for flow measurements

The STI points are located on the STI IOPs. Each STI IOP has a maximum of 16 inputs, and it can communicate bidirectionally with up to 16 Smart transmitters, regardless of the mix of transmitter types (pressure, temperature, or flow) using Honeywell's Digitally Enhanced (DE.) protocol.

The bidirectional digital communication allows the user to configure, view, and modify the transmitter database through the Analog Input Channel block associated with the STI IOP in the Monitor mode of Control Builder or the IOP's Detail display in Station. This digital protocol allows a more precise PV value to be transferred, thereby permitting more accurate control of the process. In addition, the transmitter can also send a secondary variable such as the transmitter temperature, cold junction temperature, or totalized value, depending on the transmitter type.

Multivariable transmitter support

The STIMV IOP supports all the Smart Transmitters listed above and multi-PV Smart Transmitter types such as the following:

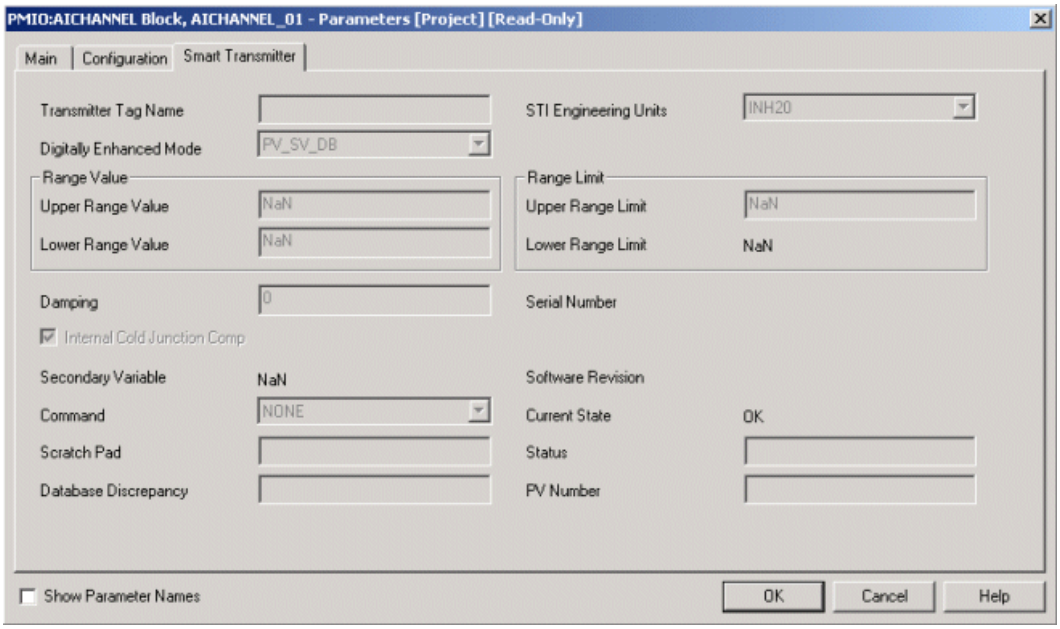
- SCM3000 Smart Flow Transmitter (Coriolis method)
- Drexelbrook SLT Level Transmitter
- SMV 3000 Multivariable Pressure Transmitter
- SGC 3000 Gas Chromatograph

An STIMV IOP allows up to four multi-PV transmitters or a mix of multi-PV and single PV transmitter inputs that total no more than 16. A multi-PV transmitter is configured as if it were in "n" contiguous slots where "n" equals the number of PVs expected. The STITAG parameter value for each contiguous slot must be identical.

Transmitter parameters and database access

The transmitter database can be configured through the Smart Transmitter tab on the Analog Input Channel block's configuration form in Monitoring mode of Control Builder and loaded to the transmitter. The transmitter database can also be uploaded to the STI IOP as required, when the STI point is in the inactive state. During normal operation (when the STI point is in the active state), each time that the transmitter broadcasts the PV value to the STI IOP, it also sends the one byte of its database (depending on the selected DECONF mode) to the STI IOP. This allows the STI IOP to compare the stored database to the newly received database to check for database mismatches. If a mismatch is detected, the PV is set to NaN and the status is set to DBChange. The user can easily correct the discrepancy by downloading the database stored in the STI IOP.

As shown in the following figure, all key transmitter parameters can be accessed from the AI Channel block configuration form. (The following illustration is used for example purposes only and reflects a Project mode rather than a Monitor mode condition.)



STI parameter comparisons

The following table shows the comparison between parameters in the Smart Transmitter databases and those in the STI IOP for reference. These parameters are accessible through the associated AI Channel block configuration form with Control Builder in Monitor mode.

Smart Transmitter Database Parameter	Corresponding STI IOP Parameter	Remarks
Upper Range Value and Lower Range Value	URV LRV	Define the operating range of the transmitter. These values correspond to the values for PVEUHI and PVEULO, respectively.
Upper Range Limit and Lower Range Limit	URL LRL	These parameters are the respective built-in maximum and minimum limits of the transmitter and they cannot be changed. The LRL is a read-only parameter. The URL must be configured to match the URL value of the transmitter.
PV Damping	DAMPING	PV damping at the transmitter. Refer to STI PV range checking and filtering.
Tag Identifier	STITAG	Transmitter identifier.
Software Version	STISWVER	Revision level of the software in the transmitter. This is a read-only parameter.
Serial Number	SERIALNO	Serial number (PROM) of the transmitter. This is a read-only parameter.
Secondary Variable	SECVAR	For a pressure transmitter, the secondary variable is the meter-body temperature of the transmitter. For a temperature transmitter, the secondary variable is the cold-junction temperature. For a flowmeter, the secondary variable is the totalized value. This is a read-only parameter.
Linear / Square Root Characterization	PVCHAR	Refer to STI PV characterization.

Smart Transmitter Database Parameter	Corresponding STI IOP Parameter	Remarks
Communication Mode	DECONF	Refer to the following section.
Cold Junction Compensation Active	CJTACT	Applicable to STT 3000 only.

Transmitter communication mode

You can select the transmitter's communication mode through the Digitally Enhanced Mode (DECONF) parameter on the AI Channel block configuration form the possible selections are listed in the following table for reference.

If DECONF Selection is . . .	Then, Transmitter Communicates . . .
ANALOG	nothing - selection is not supported.
PV	only the Process Variable (PV) - 4-byte format.
PV_SV	the PV and the Secondary Variable (SV) - 4-byte format.
PV_DB	the PV and the database - 6-byte format.
PV_SV_DB	the PV, SV, and database - 6-byte format.

ATTENTION

We recommend using the PV_DB or PV_SV_DB mode, since they support database mismatch detection and on process mismatch recovery.

STI IOP commands

You can issue the following commands through the Command (COMMAND) parameter on the Smart Transmitter tab of the AI Channel block's configuration form. The Control Builder must be in Monitor mode and the Point Execution State (PTEXECST) must be Inactive.

If Command Selection Is . . .	Then, Action Initiated Is . . .
NONE	none
DnLoadDb	downloads the transmitter parameters from the STI point database into the transmitter.
UpLoadDb	uploads the transmitter database from the transmitter into the STI point.
Set_LRV	sets the Lower Range Value.
Set_URV	sets the Upper Range Value.
Cor_LRV	corrects the Lower Range Value.
Cor_URV	corrects the Upper Range Value.
Cor_Inpt	corrects the zero point for the Process Variable (PV).
RstCor	sets all input calibration parameters to the default values.
Calibrate	Refer to the Transmitter's *User Manual.*
ResetErr	Refer to the Transmitter's *User Manual.*
Restart	Refer to the Transmitter's *User Manual.*
Shutdown	Refer to the Transmitter's *User Manual.*

Point states

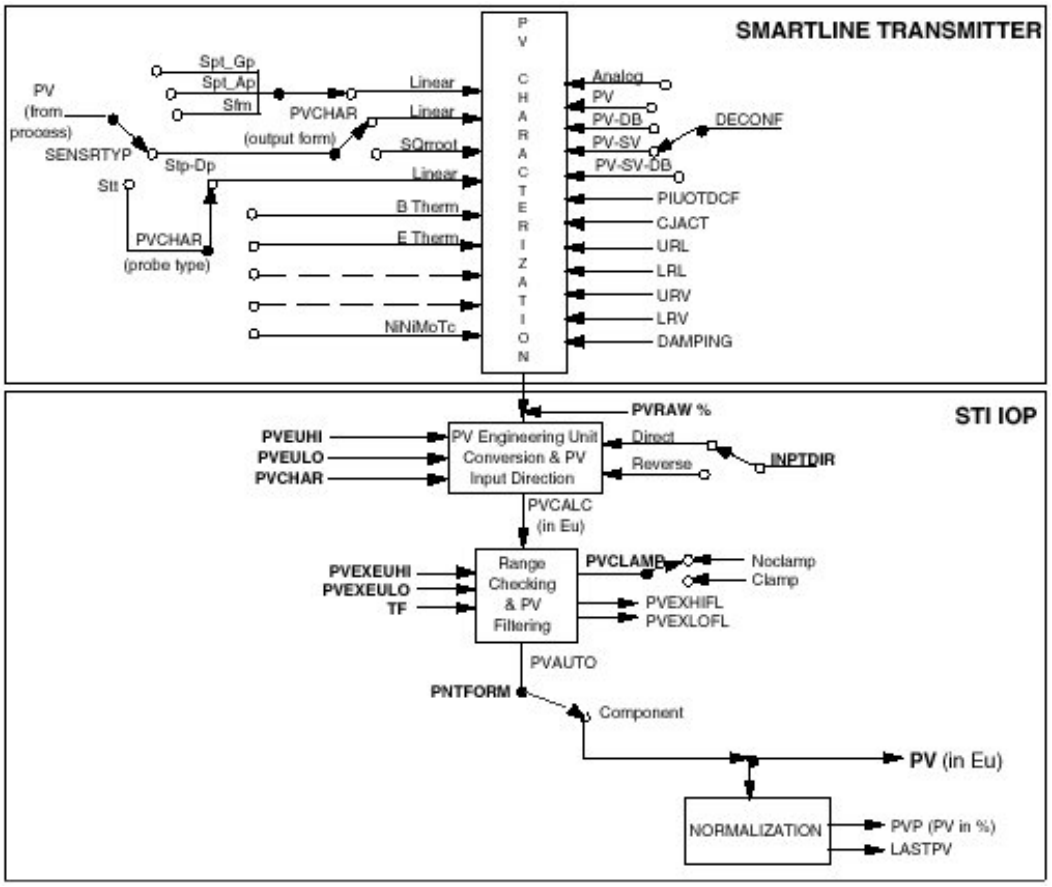
You can view the status of the STI IOP and the transmitter through the Current State (STATE) read-only parameter on the Smart Transmitter tab of the AI Channel block's configuration form. The following table lists the possible states for reference.

If Current State Is . . .	Then, Existing Condition Is . . .
None	off net.
Loading	loading of the database is occurring between the STI point and the transmitter.
Loadcomp	loading of the database between the STI point and the transmitter has completed successfully.
Loadfail	loading of the database between the STI point and the transmitter has failed.
Calib	the STI point is calibrating certain parameters at the transmitter.
Calcomp	calibration has completed successfully.
Calfail	calibration has failed.
OK	Normal State - The STI and the transmitter are okay. Transmitter is updating the Process Variable (PV) value at the STI point. State remains OK when the point is made inactive.
DBChange	a database mismatch exists between the STI point and the transmitter. Transmitter is not updating the PV value at the STI point. State remains Dbchange when point is made inactive

STI IOP functions

The STI IOP performs the following functions as illustrated in the following figure.

- PV Characterization
- Range Checking and PV Filtering



STI PV characterization

The PV signal (PVRAW) received from the transmitter has been characterized by the transmitter in terms of linear or square root characterization and damping. For the STT 3000, PVRAW is further characterized based on the entries that the user makes for the SENSRTYP, PVCHAR, and INPTDIR parameters as shown in the previous figure. The following table lists the PV characterization options available for the various transmitter (sensor) types.

Transmitter (Sensor) Type (SENSRTYP)	PVCHAR Options	PVRAW(1,3)	PVCALC	PV Detection(2)
Spt_Dp (Differential Pressure)	Linear Square Root	Percent in Water	EU	Range check on PVCALC
Spt_Gp (Gauge Pressure)	Linear	Percent in Water	EU	Range check on PVCALC
Spt_Ap (Absolute Pressure)	Linear	Percent in Water	EU	Range check on PVCALC
Stt (Temperature)(4)	Linear	Percent Millivolts	EU	Range check on PVCALC
	Thermocouple	Percent degrees C	EU	Open thermocouple detection, and range check on PVCALC.
	RTD	Percent degrees C	EU	Range check on PVCALC
	RTD Ohms	Percent Ohms	EU	Range check on PVCALC
Sfm (Flow)	Linear	Percent cubic meters per hour	EU	Range check on PVCALC

Transmitter (Sensor) Type (SENSRTYP)	PVCHAR Options	PVRAW(1,3)	PVCALC	PV Detection(2)

Legend: EU = Engineering Units
PVCALC = Calculated Process Variable
PVCHAR = Process Variable Characterization
PVRAW = Process Variable received from transmitter and multiplied by 100 by the STI IOP.

Notes:

a) PVRAW is a percentage of the configured range for the sensor type. For Multivariable transmitters, PVRAW Engineering Units are different for each PV slot.

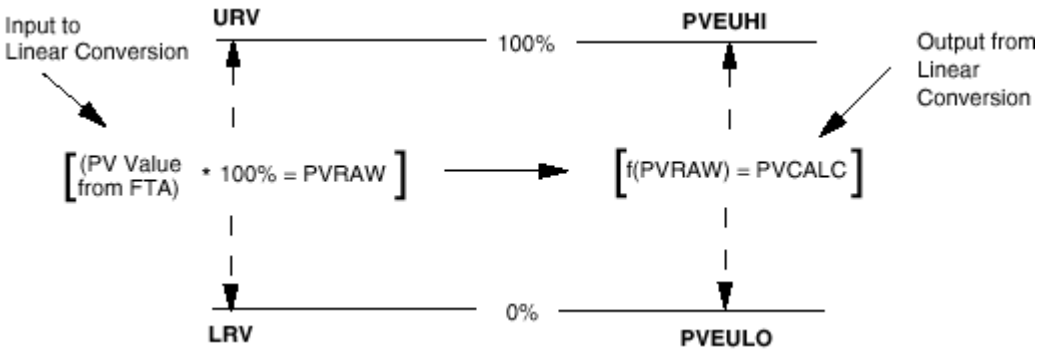
b) If the transmitter gross status indicates Output mode or Bad, PVRAW of the STI point is set to NaN, and PVSTS is set to Bad.

c) The normal operating range for PVRAW (0 percent equals PVRAWLO, 100 percent equals PVRAWHI) is configured by the user.

d) For the supported temperature ranges, refer to the definition of the PVCHAR parameter in the *Control Builder Parameter Reference*.

STI linear conversion

If the entry for PVCHAR is Linear, the PVRAW input from the FTA is calculated as a proportion of the input span in percent, as determined from upper and lower range values URV and LRV. This proportion is then used in generating an identical proportion of the output span, as determined from PVEULO and PVEUHI shown in the following figure. The URV and LRV values are the 100 percent and 0 percent values that correspond to the PVEUHI and PVEULO values, respectively.



STI square root conversion

If square root is selected, this function is performed by the smart transmitter in its computation of PVRAW. The value for PVCALC is then determined in the same manner as linear conversion. These conversion equations are provided below.

- For INPTDIR = Direct:

$$PVCALC = \frac{PVRAW}{100} (PVEUHI - PVEULO) + PVEULO$$

- If INPTDIR = Reverse:

$$PVCALC = \frac{PVRAW}{100} (PVEULO - PVEUHI) + PVEUHI$$

STI thermal conversion

Thermal linearization is available for the thermocouple and RTD inputs of the Stt (temperature) transmitter. Thermal linearization is selectable by parameter PVCHAR.

The STI point calculates the reference junction compensation from the measured reference junction output level. This value is stored and then later converted back to millivolts, with respect to 0 degrees C, for each thermocouple that is to be compensated. The external cold-junction reference compensation (CJTACT) parameter is expressed in millivolts for the specified thermocouple and is added to the millivolt value for PVRAW.

For an RTD, the STI point calculates the lead-wire compensation and then subtracts the value from PVRAW.

STI PV range checking and filtering

PV range checking ensures that the PVCALC output of PV characterization is within the limits defined by parameters PVEXEULO and PVEXEUHI. If either of the limits is violated, the output of the range check is set to BadPV if clamping has not been specified. If clamping has been specified, the output of the range check is clamped.

If the range-checked and filtered value is less than the value specified by the user-configured LOCUTOFF parameter, the final output called PVAUTO is forced to PVEULO.

PV filtering can be implemented at the STI IOP, or at the Smartline Transmitter. At the STI IOP, first-order filtering is performed on PVCALC, as specified by the user through parameter TF (filter lag time). At a transmitter, filtering is performed on the PV

depending on the value entered for the DAMPING parameter using the SFC. The user should decide the type of filtering required based on the following guidelines:

- The DAMPING parameter allows for better control accuracy because more PV samples are used in calculating the filtered PV value at the transmitter.
- TF can be changed on-process from the DI Channel block configuration form. To change the DAMPING value requires the point to be made inactive and requires the database to be downloaded to the transmitter after the change has been made.

For better control accuracy, the use of the DAMPING value is preferred over the TF value. The transmitter accepts only certain DAMPING values from the STI IOP, and the value received must first be converted to one of the predefined DAMPING values that reside in the transmitter. This conversion is accomplished automatically by the STI IOP by finding the DAMPING value that is nearest to the desired DAMPING value.

DAMPING values differ between the Smartline Transmitters. The valid DAMPING values for each transmitter type are contained in the following table.

Valid DAMPING Values in Seconds		
Pressure (Spt)	***Temperature (Stt)***	***Flow (Sfm)***
0.0	0.0	0.0
0.16	0.3	0.5
0.32	0.7	1.0
0.48	1.5	2.0
1.00	3.1	3.0
2.0	6.3	4.0
4.0	12.7	5.0
8.0	25.5	10.0
16.0	51.1	50.0
32.0	102.3	100.0

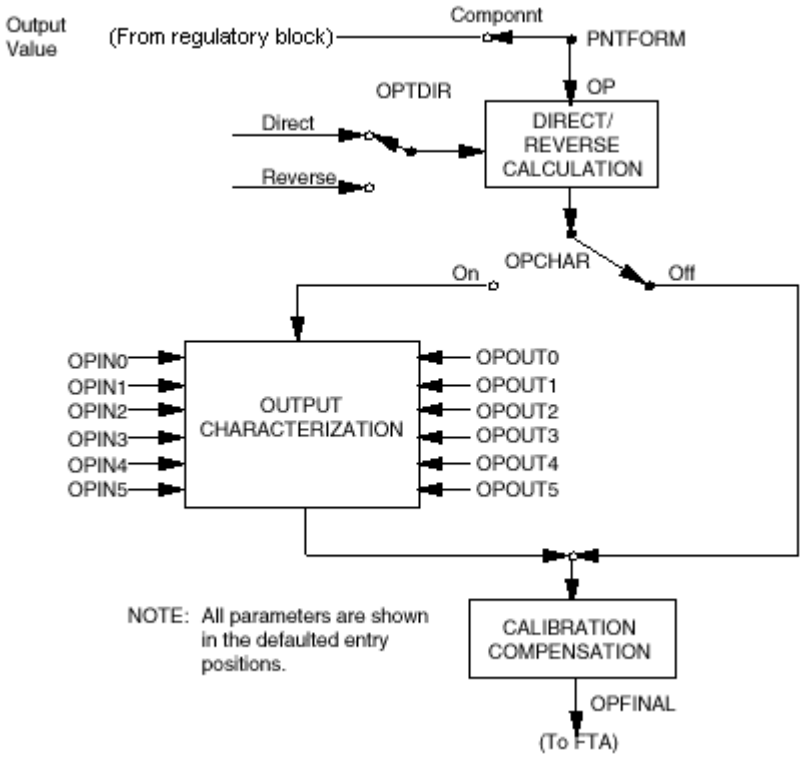
Analog Output Point

AO functions

The analog output point converts the output value (OP) to a 4-20 mA output signal for operating final control elements such as valves and actuators in the field. The OP parameter value can be controlled from a regulatory function block contained in a Control Module. To convert the OP value to a 4-20 mA signal, the analog output point performs:

- Direct/reverse Output Function
- Nonlinear Output Characterization

An option allows redundant Analog Output points. The following figure is a functional diagram of the analog output point.



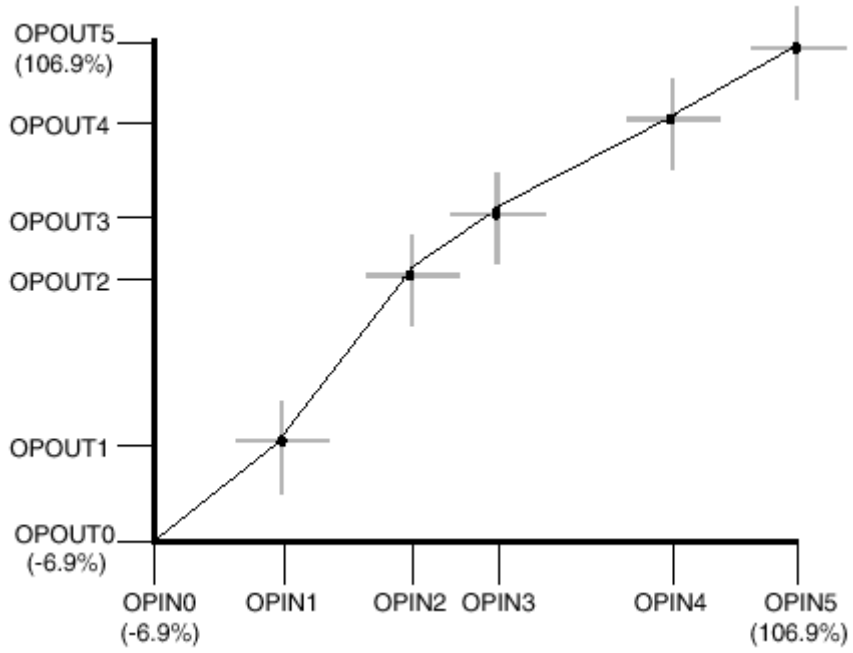
AO direct/reverse output

Parameter OPTDIR allows the user to specify whether the output of the data point is direct acting (where 4 mA equals 0 percent and 20 mA equals 100 percent or reverse acting (where 4 mA equals 100 percent and 20 mA equals 0 percent). The default mode is direct acting.

AO output characterization

Output characterization allows the user to specify an output transfer function, using configurable X-Y coordinates that provide five linear segments as shown in the following figure. The length of each segment is variable according to the coordinates that can be entered as applicable constants for OPOUT1-4 and OPIN1-4 parameters, which are real numbers.

As shown in the following figure, the end points of the curve are fixed at coordinates OPOUT0,OPIN0 (at -6.9 percent) and OPOUT5,OPIN5 (at 106.9 percent). These coordinates are fixed at these values to ensure that neither the characterization function nor its inverse can provide output values, which are outside the -6.9 to 106.9 percent range.



AO calibration compensation

The final stage of output processing in the analog output point is calibration compensation. This is accomplished in the data point using internal offset and scale constants. The output value OPFINAL is then routed to the field through the appropriate FTA.

ATTENTION

Slot or module level soft failures can prevent a point (or points) from outputting to the field. The regulatory control point will initiate a "Bad Output" alarm (If configured) when any connection is broken. If all configured point connections to the field are broken, the regulatory control point driving that analog output slot goes into initialization.

Digital Input Point

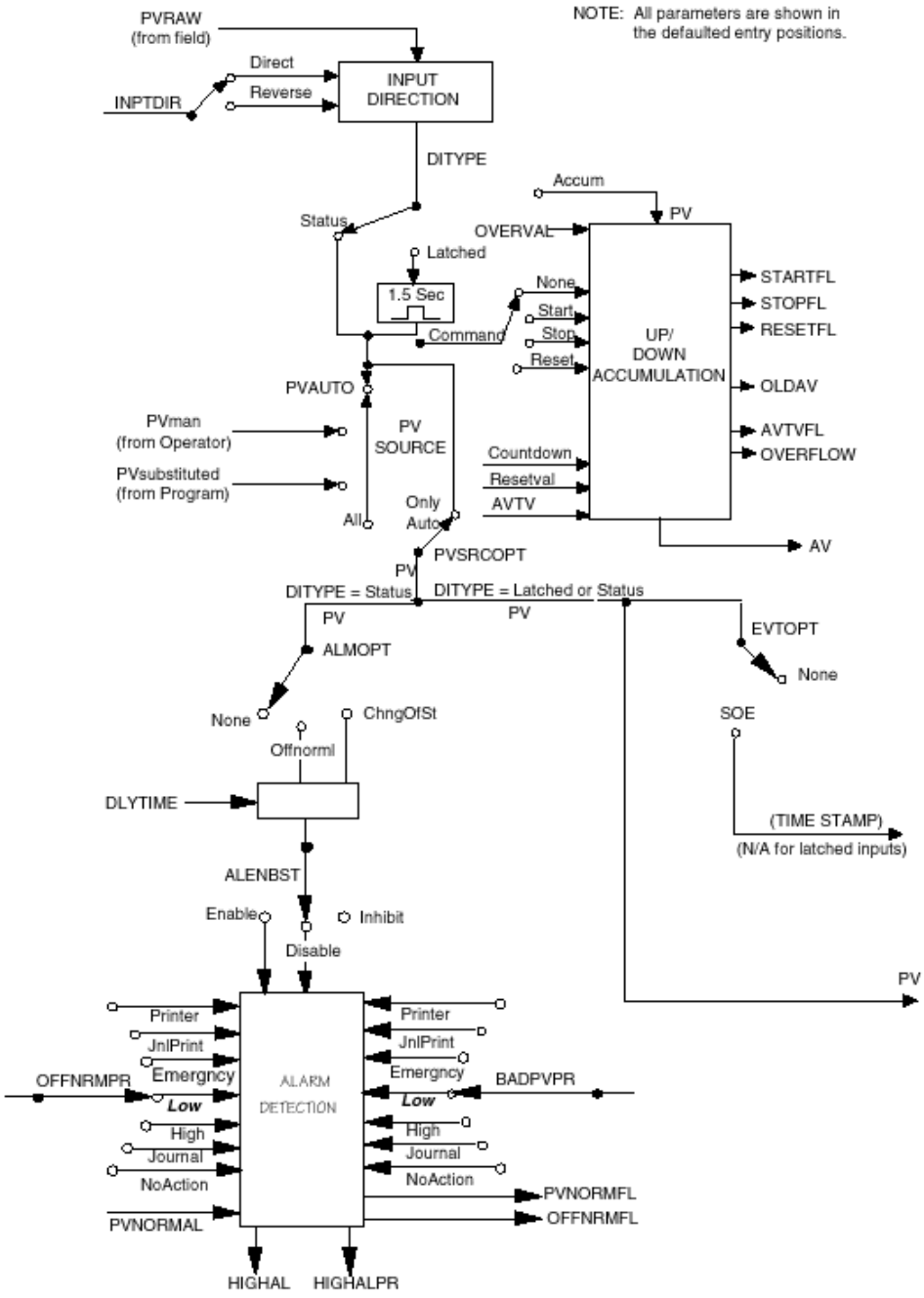
DI functions

A digital input point converts a digital PVRAW signal received from the field to a PV that can be used by other data points in the control strategy. A functional diagram of the digital input point is shown in the following figure.

Control strategies can test for a bad Digital Input PV. Parameter BADPVFL is set ON when:

- The PV source has been switched to Substituted, and the point is inactive or the module status is Idle.
- The PV source is AUTO and the PV is not being updated, because either the point is inactive, the module is idle, there is a slot soft failure, or the FTA is missing.

The digital input point is a single-input point that can be configured as a status input or a latched input, as described in the following sections.


NOTE: All parameters are shown in the defaulted entry positions.
PVRAW (from field)
Direct
Reverse
INPTDIR
INPUT DIRECTION
DITYPE
Accum
PV
Status
OVERVAL
Latched
1.5 Sec
None
Start
Stop
Reset
Command
UP/ DOWN ACCUMULATION
STARTFL
STOPFL
RESETFL
OLDAV
AVTVFL
OVERFLOW
AV
PVAUTO
PVman (from Operator)
PVsubstituted (from Program)
PV SOURCE
All
Only Auto
Countdown
Resetval
AVTV
PVSRCOPT
PV
DITYPE = Status
DITYPE = Latched or Status
PV
PV
ALMOPT
EVTOPT
None
None
ChngOfSt
Offnorml
SOE
DLYTIME
(TIME STAMP)
(N/A for latched inputs)
ALENBST
Enable
Inhibit
Disable
PV
Printer
JnlPrint
Emergncy
Low
High
Journal
NoAction
OFFNRMPR
ALARM DETECTION
BADPVPR
PVNORMAL
PVNORMFL
OFFNRMFL
HIGHAL
HIGHALPR

DI status point

For this digital input type, the PVAUTO value represents the state of the raw input signal after the direct/reverse conversion is performed. The status digital-input point can be configured for PV source selection, detection of off-normal alarms, and for reporting any PV state changes to the system. The status digital input point is selected by entering Status for the DITYPE parameter.

The current state of the PV input is represented on the Station Detail Display as an indicator light. The lights are lighted or extinguished depending on the current state of PVRAW and the input direction as configured through the INPTDIR parameter. The current PV state is also available to be used as an input to logic slots, and other Control Builder control functions.

DI PV source selection

The PV source parameter (PVSOURCE) option determines the source of the PV for a status input point. The source can be the PV input from the field (PVauto), the PV state entered by the operator (PVman), or it can be supplied by a user program (PVsub). PVSOURCE has no effect on the latched and accumulation options of the digital input point. If PVSOURCE is PVauto, PV tracks PVRAW.

DI off-normal alarming

Off-normal alarming can be selected for the digital input point through the ALMOPT parameter. An off-normal alarm is generated when the input PV state is different than the configured normal (desired) state for the point as specified by the PVNORMAL parameter.

Digital Input Status points (and Sequence of Events points) can be configured for Change Of State (COS) alarm reporting through the ALMOPT parameter. The alarm is generated when the input changes state in either direction.

COS alarms are removed from the Alarm Summary display following acknowledgement. The Point does not remain in alarm so there is no Return-to-Normal. Point Detail or Group displays will never show a point in COS alarm.

Older digital input IOPs may need to have a new firmware chip for COS reporting. Check the IOP's detail display. For COS reporting the Digital Input IOP firmware revision must be 5.0 or later.

Note that when a point with COS reporting is changed from Inactive to Active, a COS alarm is generated if the PV equals 1. There is no COS alarm if the PV equals 0. The same alarming occurs if the point is active and the IOP is put into Run mode.

Alarm delay

When off-normal alarming has been configured and an off-normal alarm is detected, the event is reported to the system. Further off-normal alarms for the same data point are not reported until the time delay (0 to 60 seconds) specified by the DLYTIME parameter expires. When the time delay expires, the time-delay function is disabled and the off-normal alarm for the data point can again be reported.

For Change of State alarms, when a PV state change occurs, a COS alarm is produced and the delay timer is started. When DLYTIME expires, two situations are possible:

- The PV is in the same state and future state changes are immediately alarmed.
- The PV is in the opposite state (it may have changed many times during the DLYTIME period) so a second COS alarm is produced and the timer starts again.

Event reporting

The EVTOPT parameter for the status input allows the user to optionally specify that a time stamp be added to the reported PV state change. For a status input point, EVTOPT has the two possible entries: None or SOE. The SOE specifies that a time stamp is added to the PV state change to establish a sequence of events.

DI latched input point

To capture the occurrence of momentary digital inputs, such as from pushbuttons, requires the user to configure the digital input point as a latched digital input point. Configuring the point as a latched point is accomplished by entering Latched for the DITYPE parameter.

When configured as a latched input point, an input pulse that is on for a minimum of 40 milliseconds is latched true for 1.5 seconds. This ensures that any control function that needs to monitor this event will execute at least once during the time that the signal is latched on.

The current state of the latched PV input is represented as indicator light on the Station Detail display. The lights are lighted or extinguished depending on the current state of PVRAW and the input direction as configured through the INPTDIR parameter. The current PV state is also available to be used as an input to other control strategies. .

DI sequence of events point

Sequence of Events (SOE) points are used to report the order of occurrence of digital state changes. The SOE digital input IOPs can use the same type FTAs as digital input cards, but the best overall performance is obtained when using the high resolution 24 Vdc Digital Input FTAs.

SOE events are recorded in a journal with a timestamp so that you can determine, for example, which event started an upset and the progression of events thereafter. The record includes the point ID, point descriptor, state text unit, and time of occurrence to one millisecond resolution.

SOE definitions

In practice, the capture of physical digital state changes is less precise than the nominal 1 millisecond time stamping that is degraded by other system factors. This means that time stamps in logs must differ by multiple milliseconds before the correct order can be determined. The following definitions will be helpful reference in applying the specifications associated with SOE generation in the IOP and I/O Link Interface Module (IOLIM).

Term	Definition
Resolution (Tres)	The smallest increment of real time that can separate two consecutive SOE time-stamped events. The resolution of the field-connected equipment which first stamps an event limits the inherent resolution of the system.
Sequence Stamp Difference (SSD)	The minimum difference in logged time stamps that guarantees the order of two time stamps.
Minimum Physical Event Separation (MPES)	The smallest interval between two ideal (bounceless) physical events, so the events can be correctly ordered by the Sequence Stamp Difference.
Skew (Tskew)	The sum of all factors which cause differing time stamps to be applied to the same physical event, if wired into two different places in the system.

SOE resolution considerations

The IOLIM synchronizes the I/O Link every 2 seconds. Consider the following values when determining SOE resolution. In general terms, the resolution is X milliseconds; the time stamps that differ by Y milliseconds or more indicate true event sequence; and real-world events must occur Z milliseconds or more apart to be assigned time stamps that differ by Y milliseconds.

For points in one IOP

- Skew: 0+0.2+0.04+0.2+0.2+0.1+0.5 = 1.24 msec
- SSD: N = 7, (7+1) * 0.200 = 1.6 msec
- MPES: 1.6 + 1.24 = 2.84 msec

For points in one IOLIM

- Skew: 0+0.2+0.04+0.2+0.2+0.2+0.5 = 1.34 msec
- SSD: N = 7, (7+1) * 0.200 = 1.6 msec
- MPES: 1.6 + 1.34 = 2.94 msec

For points in redundant IOLIMs

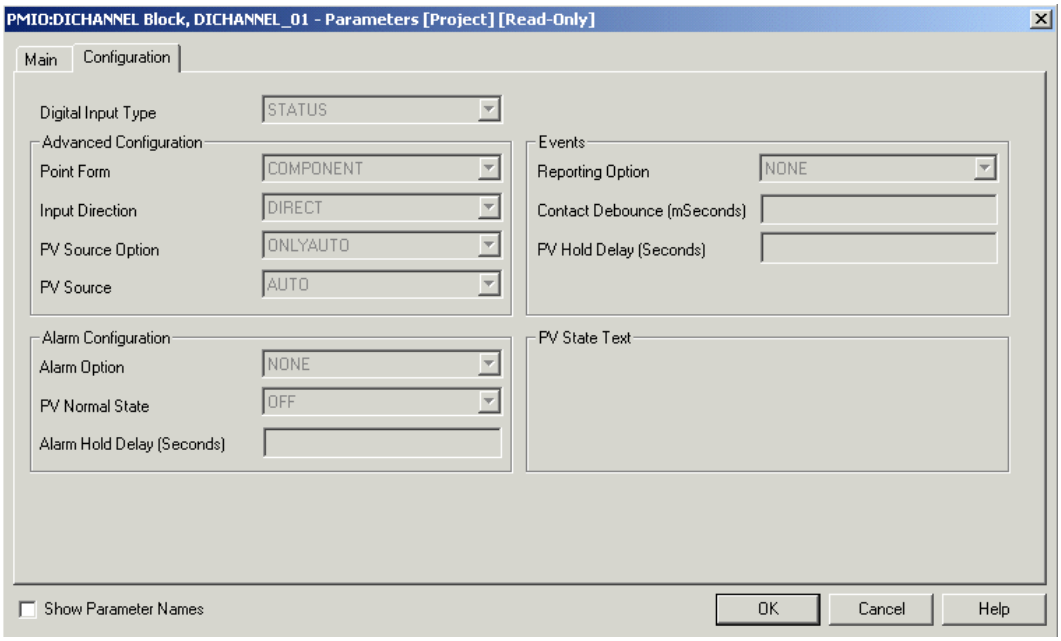
- Skew: 0.5*+0.4+0.04+0.2+0.2+0.2+0.5=2.04
- SSD: N = 11, (11+1) * 0.200 = 2.4
- MPES: 2.4 + 2.04 = 4.44

*Estimated value.

This means that time stamps that differ by 2, 2, 3 milliseconds in the SOE journal show order for events that differ by 3.8, 3.8, 5.8 milliseconds, respectively

DI SOE configuration considerations

Digital input SOE IOP points are configured through configuration tab of the DI Channel block for loaded control strategy and Control Builder in Monitoring mode. The configuration considerations are similar to those described for the conventional digital input point with the following additions for events, as shown in the following figure. (The following illustration is used for example purposes only and reflects a Project mode rather than a Monitor mode condition.)



Choose STATUS as the Digital Input Type (DITYPE). You can use the DISOE IOP as a conventional digital input, if you choose LATCHED.

The Contact Debounce time (DEBOUNCE) parameter specifies the time interval used to debounce an input from mechanical contacts of a field input source. It is defined as the length of time following an input state change during which the input must remain unchanged in the new state to declare it a valid event. DEBOUNCE has a range of - 50 milliseconds in one-millisecond increments. The default value of 10 ms should suffice for most contacts. If not, choose a value slightly longer than the manufacturer's specified contact bounce time.

The following figures illustrate the debounce operation.

This waveform represents the field input. Tick marks across the waveform indicate the 200 microsecond scan intervals of the DISOE IOP hardware. Assume that the input state changes at point A.

At point B, the state change is detected. At this point, the current time and old state are recorded. The debounce timer is started.

- If the input remains at a steady state until the debounce timer expires, then an event is generated with a timestamp corresponding to the time of detection (point B).

- If the input changes before the debounce timer expires (point c), then the change of state event detected at B is discarded, the timer is restarted and runs for the full debounce time:
  - If a new input detected at D remains in a steady state until the debounce timer expires (point F), then an event is generated with a timestamp corresponding to the original time of detection (point D).
  - If the input has returned to the old state (dotted line at E) when the debounce timer runs out (point F), no event is generated.

The PV Hold Delay (PVCHGDLY) parameter specifies the time of separation in seconds for reporting two consecutive PV change events from the same input source. It is intended to prevent repeated rapid reporting of PV change events (i.e., chattering). It can be configured over a 0 to60 second range in one second increments. It applies to points configured as SOE only. Ideally, PVCHGDLY and DLYTIME should have the same value.

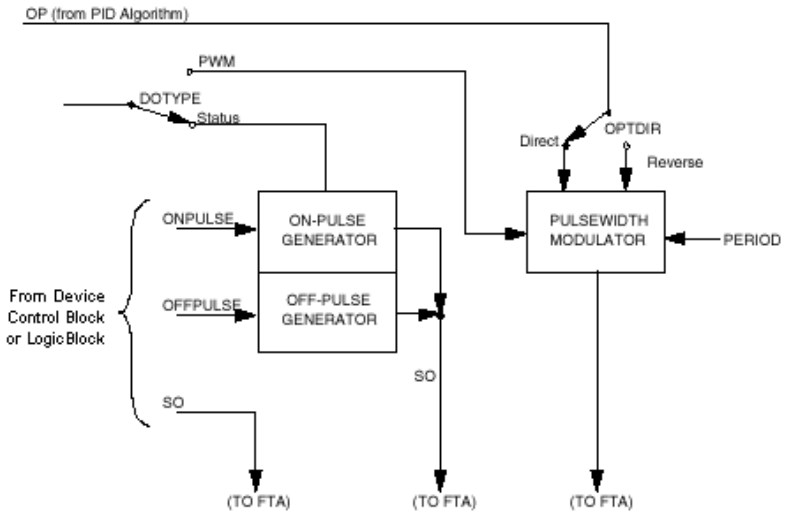
When a PV state change is detected, the change is reported and the PV change delay timer is started.

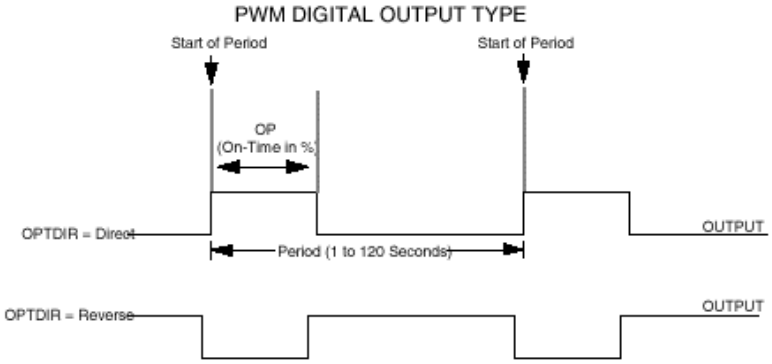
- If the PV does not change before the delay timer runs out, no further action is taken.
- If the PV state changes only once before the delay timer runs out, this second event is noticed and time stamped with the actual time of detection, but it is not reported until the delay timer expires.
- If the PV state changes more than once before the PV change delay timer runs out, only the last state change is noticed and time stamped but not yet reported. When the delay timer expires:
  - If the PV state is different from the state that started the timer, this event is reported.
  - If the PV state is the same as the original PV state that started the timer, no event is reported.

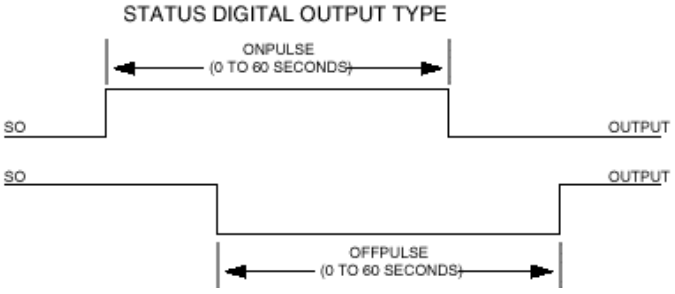
Digital Output Point

DO functions

The digital output point provides a digital output to the field based on the origin of the input and the configured parameters. A functional diagram of the digital output point is shown in the following figure. The digital output point does not have any modes.


OP (from PID Algorithm)
PWM
DOTYPE
Status
Direct
OPTDIR
Reverse
ONPULSE
ON-PULSE GENERATOR
PULSEWIDTH MODULATOR
PERIOD
From Device Control Block or Logic Block
OFFPULSE
OFF-PULSE GENERATOR
SO
SO
(TO FTA)
(TO FTA)
(TO FTA)


PWM DIGITAL OUTPUT TYPE
Start of Period
Start of Period
OP
(On-Time in %)
OPTDIR = Direct
OUTPUT
Period (1 to 120 Seconds)
OPTDIR = Reverse
OUTPUT


STATUS DIGITAL OUTPUT TYPE
ONPULSE
(0 TO 60 SECONDS)
SO
OUTPUT
SO
OUTPUT
OFFPULSE
(0 TO 60 SECONDS)

There are two types of digital output points: pulse-width modulated (PWM) output and status output. Selection of the output type is accomplished through the DOTYPE parameter shown in the previous figure. The PWM type is used in combination with regulatory control block algorithms to provide true proportional control. The status type output is the normal configuration for digital outputs that are linked to Device Control block points. Actual output action can be status, latched or momentary, depending on the configuration of the Device Control point. The default for untagged component DO points is Status.

Pulse Width Modulated (PWM) Output Type

The pulse width modulated output type can receive its input from a regulatory PID block through a user-configured output connection. The length of the pulse is derived from the OP parameter provided by the regulatory point. Because OP is in percent, the percent value becomes the percent on time for the pulse whose period (1 to 120 seconds) is specified by the PERIOD parameter, as shown in the timing diagram in previous figure.

The output direction of the output signal can be configured to be direct or reverse acting by using the OPTDIR parameter.

The pulse on time for direct and reverse acting outputs is calculated as follows:

For direct action:

$$\text{Pulse On-Time} = \frac{OP\% * PERIOD}{100}$$

For reverse action:

$$\text{Pulse On-Time} = \frac{100\% - OP\% * PERIOD}{100}$$

If the value of OP is less than 0 percent, it is clamped to 0 percent; an OP with a value greater than 100 percent is clamped to 100 percent.

Status Output Type

The status output type can be controlled from a Device Control block output, a logic slot output, or a Position Proportional block, as determined by the output connection. The output latch function is obtained by linking Device Control block output connections to the SO parameter. Pulsed operation (pulse-on or pulse-off) can be obtained by linking the output connections to the ONPULSE and OFFPULSE parameters, respectively.

The ONPULSE parameter sets SO to On for the specified duration. At the end of the pulse time, SO is set to Off. If ONPULSE is specified as 0.0, SO is immediately set to Off. This also applies to the OFFPULSE, except that the OFFPULSE sets SO to Off.

If SO is received from a logic slot, the SO output of the digital output point tracks the SO output provided by the logic slot.

Initialization request flag

When ON, this parameter indicates (for Status Output type points) that control strategies in the Controller cannot manipulate the output. Parameter INITREQ is set ON when:

- a PWM type output is configured
- a Status Output type is configured and:
  - the point is inactive
  - the module is idle
  - there is a soft failure such that the point is not working.

Component Categories and Types

Overview

About categories

For data organizational purposes, we divide the Control Builder components into these two major categories:

- *Hardware Relation Category*
- *Functional Relation Category*

These categories are explained in the *Experion Control Builder Components Reference* in *Hardware Relation Category* and *Functional Relation Category*. In brief, the Hardware Category consists of self-standing type blocks and the Functional Category consists of container and component type blocks. Please note that self-standing type blocks are referred to as physical equipment type blocks, and container and component type blocks are referred to as functional type blocks.

Function block types and Data Organization

The remainder of this document is organized according to the following grouping of function block types.

- Regulatory Control
- UCN Interface
- Exchange Functions
- Auxiliary Functions
- Data Acquisition Functions
- Pulse Input
- Device Control
- Logic Functions
- Utility Functions
- Sequential Control

To minimize repetition of data, this document does not include topics specific to the physical equipment type blocks (self-standing FBs) and the Control Module FB, and it does not list all the parameters associated with a given FB. This information can be found in the *Experion Control Builder Components Reference*.

Regulatory Control

Regulatory Control Blocks

Functional overview

Experion Regulatory Control blocks offer a veritable "toolkit" of process control functions. Each block is configured to perform a single control function, and multiple blocks may be connected to form complex control strategies.

The following table presents the various functions that can be performed through the configuration of the associated Regulatory Control block.

Function	Block	Description
Auto Manual	AUTOMAN (Auto Manual) Block	Provides a "bumpless" output following initialization or mode changes; typically used in a cascade where one of the upstream blocks may not accept an initialization request.
Enhanced Regulatory Control Calculator	ENHREGCALC (Enhanced Regulatory Control Calculator) Block	Provides enhancements over the REGCALC block. The enhancements include: • Expands existing arrayed input parameters XSTS and X from six to ten. • These arrayed parameters are added to correspond to each of the ten inputs. – Input Description – Enable/Disable Switch – XSUB Substitute Parameter See the block section for other enhancements.
Fanout	FANOUT Block	Sends one input to many outputs, whereby a different gain and bias may be assigned to each output.
Incremental Summer	INCRSUMMER	Calculates the sum of the incremental changes in up to four input values. The

Function	Block	Description
		output is obtained by adding the sum of the changes in all the inputs after each input is multiplied by its respective scale factor.
Override Selector	OVRDSEL (Override Selector) Block	Selects one input from many based on the highest or lowest input value; the OVRDSEL block always forces the unselected inputs to track the selected inputs, but provides two methods for doing so: • Propagate override feedback data to the unselected inputs. • Continually initialize the unselected inputs.
Proportional, Integral & Derivative	PID Block	Provides an implementation of the PID algorithm, using the Ideal form, whereby the following combinations of control terms may be configured: • Proportional-only (acts on the error PV -SP) • Integral-only (acts on the error PV - SP) • Proportional & integral (act on error PV - SP), & derivative (acts on changes in PV) • Proportional & derivative (act on changes in PV), & integral (acts on the error PV - SP) • Proportional, integral & derivative (act on the error PV - SP)
Proportional, Integral & Derivative with Profit Loop PKS	PID-PL (Profit Loop PKS) Block	Provides PID control using a model predictive control package called Profit Loop PKS that incorporates robust control techniques to enhance control performance despite process model uncertainty and measurement error.
Proportional, Integral	PIDER (PID with	Operates as a proportional-integral-

Function	Block	Description
& Derivative with External Reset Feedback	External Reset Feedback) Block	derivative (PID) controller that accepts a reset feedback signal, a tracking value, and a tracking control switch. It supports the same Ideal form of calculating the PID terms as the PID block. It also prevents windup when the secondary does not propagate windup status or control initialization data back to the primary of a remote (foreign) controller
Proportional, Integral & Derivative with Feedforward	PIDFF (PID with Feedforward) Block	Provides the same classic PID function as outlined above with the ability to accept a “feedforward” signal. You can configure the feedforward signal to be added to or multiplied by the PID’s incremental output to meet varying control requirements.
Position Proportional Control	POSPROP (Position Proportional) Block	Provides two digital outputs for pulse control of a final control element. It accepts process variable (PV) and set point (SP) inputs.
Pulse Count	PULSECOUNT Block	Provides pulse waveform generation on its four main outputs using a pulse control algorithm that relates the waveform to the configurable period and requested pulse time parameters. Typically, used in conjunction with a POSPROP block.
Pulse Length	PULSELENGTH Block	Provides pulse waveform generation on its four main outputs using a pulse control algorithm that relates the waveform to the requested pulse time parameter. Typically, used in conjunction with a POSPROP block.
Ramp Soak	RAMPSOAK Block	Provides an output that follows the user configured sequence of ramp/soak pairs. Each ramp/soak pair consists of a configurable soak value or ramp target value, a soak time and a ramp rate. Typically, used in conjunction with a PID block.
Ratio and Bias	RATIOBIAS Block	Provides a calculated output based on

Function	Block	Description
		the ratio of the input variables plus a fixed and/or a floating bias. Typically, used between two PID blocks to implement a form of ratio control.
Ratio Control	RATIOCTL (Ratio Control) Block	Accepts the actual value of the controlled flow (X1), the actual value of the uncontrolled flow (X2) and the target ratio between the flows (SP), and calculates the target value of the controlled flow (OP) and the actual ratio between the flows (PV) as outputs.
Regulatory Control Calculator	REGCALC (Regulatory Control Calculator) Block	Lets you write up to eight expressions for creating custom algorithms for Calculated Variable (CV) calculations. Provides an interface to windup, initialization and override feedback processing, so you can add user-defined control blocks to your control strategies.
Remote EEOut	REEOUT Block	Used in conjunction with the Inter Cluster Gateway to support regulatory cascades between ACE nodes residing in two separate Experion clusters.
Regulatory Summer	REGSUMMER	Calculates an output value which is the sum of up to four inputs. Each of the inputs may be individually scaled. The output may be scaled by an overall gain, and an overall bias may be added to the result.
Remote Cascade	REMCAS (Remote Cascade) Block	Provides automatic switching between a primary (remote) and backup (local) cascade; typically used with a PID that normally gets its set point from a remote source, but sheds to a local source when the remote cascade is broken.
Switch	SWITCH Block	Operates as a single-pole, 8-position rotary switch that may be positioned by the operator, user program or another block. Typically used to assign different primary to a secondary; allows user to select one from as many as 8 inputs and

Function	Block	Description
		outputs the selected value.

Common regulatory control functions for C200 and ACE controllers without insertion points

ATTENTION

The Regulatory Control blocks can interact with Sequential Control Modules (SCM) to provide interactive control action. Please refer to the *Sequential Control Module Use'sr Guide*for more information on this function.

Listed below are the major functions performed by all Regulatory Control blocks along with a brief functional description for each. Functional descriptions for each block are given in the following subsections.

Major Function	Description
Input Processing	Provides these functions for all Regulatory Control function blocks to be used as needed. PV Processing – Regulatory control blocks that have PV input use this function to fetch the input value, status, and update the appropriate PV parameters. SP Processing – Regulatory Control blocks that have a SP input use this function for SP limit checking. SP/PV Processing – Regulatory Control blocks that have SP and PV inputs use this function for SP target value processing, PV tracking, deviation alarming, and advisory deviation alarming. These functions are configurable. PUSHSP Parameter Processing - You can use the PUSHSP parameter only to enable SP value to be pushed from another block. This parameter helps in configuring cascade control offering of inter-cluster peer-to-peer regulatory controls. When the PushSP option is enabled on the block, no wired or parameter reference connection should be made to the SP pin on the block.

Major Function	Description
Mode Processing	"Mode" identifies who may store to certain Regulatory Control parameters (for example, SP and OP). Possible choices of who may store are the operator, another function block, or a user program. Mode Processing checks for the following conditions and sets the mode as appropriate: External requests for mode switch, and Mode switch requests due to safety interlocks.
Initial Control Processing	If a Regulatory Control block is in a Cascade strategy, this function checks if the cascade has been broken. If it has, this function initializes the block and builds an initialization request for its primary or primaries.
Algorithm Calculation	This involves calculations that are unique to each Regulatory Control block (for example, PID calculation). These calculations are described more fully for each individual block in the remainder of this section.
Output Processing	This function derives the control output (OP) from the algorithm's calculated variable (CV). Among other things, it applies an output bias, compares against output limits, and if necessary, clamps OP to those limits. This function also performs OP limit alarming.
Feedback Propagation	If a Regulatory Control block is in a cascade strategy, this function may propagate windup, initialization and override information to upstream blocks. This information is used to constrain the output of the upstream blocks when a limiting condition exists downstream. Windup status is used by PID blocks to turn integral control on or off; and override is used to prevent windup when an output of a PID has been disconnected from the process (by an OVRDSEL block).

Common regulatory control functions for ACE controllers with insertion points

In systems running R210 or greater, you can insert Control Alogrithm Block (CAB) programs at the following predefined locations in a regulatory control block that is associated with a control strategy to be run in an ACE Controller.

Insertion Type	Description
Post_Input	Insert CAB instance after input processing
Pre_Alg	Insert CAB instance before the execution of the algorithm.
Ctl_Alg	Insert CAB instance to replace the regulatory control algorithm.
Post_Alg	Insert CAB instance after algorithm processing.
Post_Ctlout	Insert CAB instance after the execution of control output processing.

The following illustration shows where you can insert CAB programs in relation to the regulatory control block's major functions. This means that you can use CAB programs to enhance aspects of the block's execution but still take advantage of the more complex system functions such as initialization, anti-reset windup, range-checking and alarming integral to the native block.

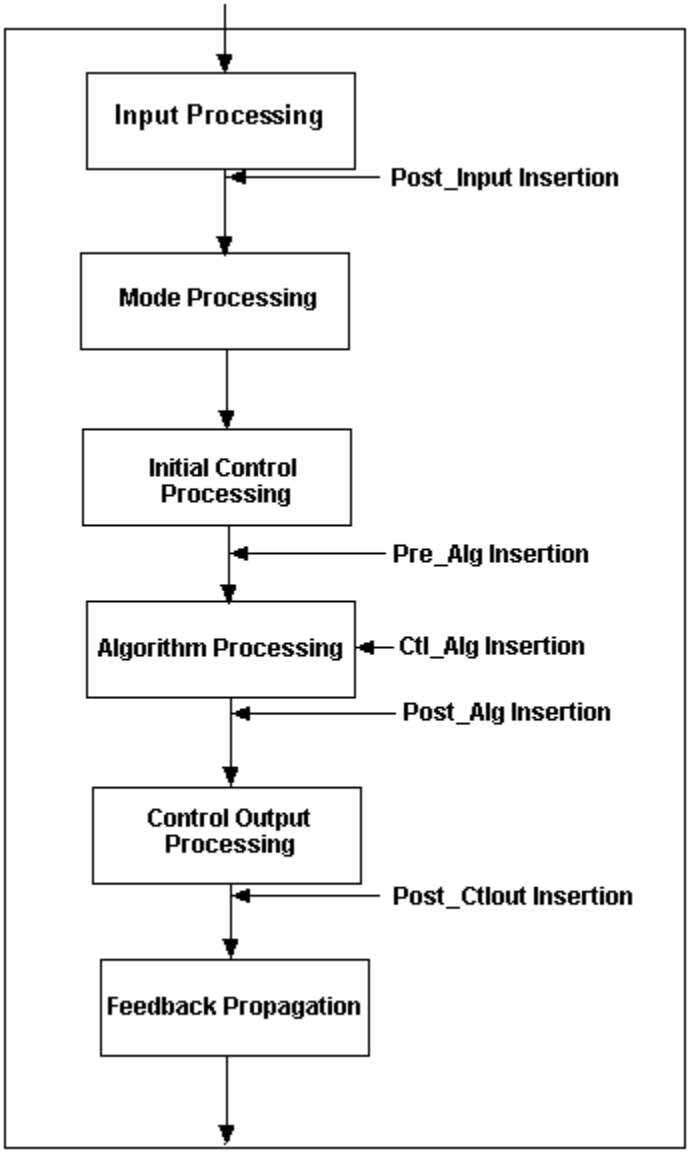


Figure 13 CAB insertion locations in regulatory control block major functions

CAB insertion configuration considerations for regulatory control blocks

- You cannot use CAB insertion programs with the following Regulatory Control blocks.
  - POSPROP
  - PULSECOUNT
  - PULSELENGTH
  - REGCALC
- You can insert up to 10 CAB programs in a requlatory control block.
- You can use CAB instances for standalone operation or as programs whose execution is inserted into the flow of other compatible blocks. For standalone operation, you must configure the CAB for an Access Level of PROGRAM. For insertion program operation, you must configure the CAB for an Access Level of CONTCONTROL.
- You must insert CAB instances in the same Control Module that contains the regulatory control block.
- If you insert multiple CAB programs at the same point, the order in which the insertions are configured determines their execution order. During configuration, the ORDERINCM parameter of the inserted CAB instance changes automatically to match that of the calling regulatory control block and the INSERTION parameter of the inserted CAB instances is set to TRUE.
- CAB instances configured for insertion execute only when they are called during the regulatory control block execution and are not executed as part of the normal Control Module execution.
- CAB instances configured for insertion should normally have no outside pin connections configured. If you need to share CAB instance data with blocks other than the one with insterted CAB programs, you can use parameter connectors or direct wire connections to configured pin connections for custom data parameters on the CAB instance. See the *Pin connections to inserted CAB instances* section for more information.
- The Control Builder application will not allow you to configure the same CAB instance as an insertion by more than one Regulatory Control block.

Insertion type functional characteristics for regulatory control blocks

The following table summarizes the functional characteristics for a given insertion type.

Insertion Type	Function
Post Input (Post_Input)	Provides the capability of performing custom functions such as: • Custom calculations on inputs, • Custom timeout processing function, and • Enable/disable external mode switching based on input values. It is normally used to perform calculations on the fetched input data. This means that Post Input insertion programs must check if the fetched input data is valid before using the data in calculations. The fetched data is valid only if the associated input status is Normal. For example, before performing calculations using X1, the parameter X1STS should be checked for Normal status. If the status is not normal, the fetched X1 value is not valid
Pre Algorithm (Pre_Alg)	Provides the capability of implementing custom initialization strategies..
Control Algorithm (Ctl_Alg)	Provides the capability of replacing the standard regulatory control block algorithm with user custom one written in a CAB program. If you use a CAB program as the algorithm, the program must calculate and store the Regulatory Control point output value CV. You must use parameter references in the CAB instance to acquire inputs for the CAB program. You should include the following items in the algorithm program: • Bad input handling (CV gets set to bad when input is bad). • CV calculation for the various control states (FORWARD, MAN and INIT).

Insertion Type	Function
	• INITVAL calculation. • Override feedback handling, if the block is part of an override strategy. • Input timeout processing, if required. • Floating bias implementation, if output transfers are to be bumpless. See the following paragraph *Functional requirements for custom CAB algorithm for requlatory control block* for more information.
Post Algorithm (Post_Alg)	Provides the capability of implementing custom calculation for output values.
Post Control Output (Post_Ctlout)	Provides the capability of implementing custom windup calculations. You can use parameters ARWNETIN and ARWOPIN to modify windup states.

Functional requirements for Custom CAB algorithm for requlatory control block

The following table summarizes the functional requirements you need to consider when creating a custom CAB algorithm for a regulatory control block.

Function	Requirement
Error Handlling	The CAB program must not abort when the regulatory control block goes into a bad control state. The output value CV must be set to NaN, when the value of the required input is bad. This casues the regulatory control block to go into a bad control state. The regulatory control block algorithm generates a BADCTL alarm during control output processing when CV is NaN.
Initialization	The CAB program algorithm must calculate the initialization value (INITVAL) that is to be propagated to its primaries during initialization. The algorithm must perform CV calculations based on the control state of the regulatory control block, which can be determined by reading the value of the CTLSTATE parameter. If the CTLSTATE value is INIT, the regulatory control block is in its initialization state.
Windup	The CAB program must set the COMPUTEARW parameter to True to initiate recomputation of windup when process conditions change. The windup state of the regulatory control block is computed after the control output has been computed.
Override Feedback	If the regulatory control block is part of an override strategy, the CAB program must set the appropriate override status and calculate the override value to be propagated to its primary. The override status to be propagated is written to the parameter PRIMDATA.ORFBSTS and the override feedback value is stored in the PRIMDATA.ORFBVAL parameter. The following is a high level implementation detail for the override propagation. BEGIN IF PRIM.INITIALIZABLE { if primary is initializable} PRIMDATA.OROFFSET = SECDATAIN.OROFFSET PRIMDATA.HISELECT = SECDATAIN.HISELECT PRIMDATA.PROPOVRD = TRUE IF ORFBSTS = NOTSELECTED {If the override status is

Function	Requirement
	not selected } PRIMDATA.ORFBVAL = SECDATAIN.ORFBVAL Calculate ORFBVAL for primary and store the value in PRIMDATA.ORFBVAL ELSE {if the status is selected } PRIMDATA.ORFBVAL = NaN IF CV = Nan {if the CV value is bad, propagate Notconnected status and a feedback value of NaN } PRIMDATA.ORFBSTS = NOTCONNECTED PRIMDATA.ORFBVAL = NaN ELSE { If CV is good, propagate the status received from secondary } PRIMDATA.ORFBSTS = SECDATAIN.ORFBSTS END
Floating Bias	The standard Regctl algorithms include a floating bias function so that output is bumpless during mode changes. If this function is required, the floating bias functionality has to be implemented in the CAB program. The standard regulatory control block algorithms store the calculated floating bias value in the parameter OPBIAS.FLOAT. This parameter allows CAB write access, so it is available for use in CAB programs.
Timeout	If the input timeout function is needed, the CAB program must implement the timeout processing as part of the program.

CAB insertions for Regulatory Control block parameters

In R210 system or greater, the access level/lock for the following regulatory control block parameters is now Continuous Control (CONTCONTROL). This provides write access for CAB programs to these parameters. This means you can modify and/or enhance other built-in regulatory control block features by inserting user written CAB programs for one or more of these parameters. The CAB programs must be configured for an access level of Continuous Control (CONTCONTROL).

Parameter	Description	Associated Block
INITREQ[1..8]	Initialization Request Flags. There is one flag for each primary.	All Regulatory Control blocks that support insertion points
INITVAL[1..8]	Initialization Value. There is one value for each primary.	All Regulatory Control blocks that support insertion points
OPBIAS.FLOAT	Floating Output Bias in Engineering Units	ENHREGCALC, PIDER, RATIOCTL, REGSUMMER
OPBIAS[1..8].FLOAT	Total Output Bias for function blocks with Multiple Outputs	FANOUT
PV	Process Variable	PID, PIDFF,
PVSTS	Status of Process Variable.	PID, PIDFF, RAMPSOAK, RATIOCTL
TMOUTFL	Timeout Flag. Indicates if an initializable input has timed-out.	All Regulatory Control blocks that support insertion points
X [..]	Input Value for blocks with multiple inputs	INCRSUMMER OVRDSEL, REGSUMMER, SWITCH

Parameter	Description	Associated Block
XKB[..]	X-Input Gain (K) and Bias (B)	ENHREGCALC
X1	Input Value	AUTOMAN, FANOUT, RATIOBIAS RATIOCTL, REMCAS
X2	Input Value	RATIOCTL, REMCAS
X1STS	Status of Input Value	AUTOMAN, FANOUT RATIOCTL
X2STS	Status of Input Value for blocks with multiple inputs	RATIOCTL

Pin connections to inserted CAB instances

Normally, inserted CAB instances do not have outside pin connections to their custom data parameters (CDPs). Inserted CAB instances usually share data only with their calling block, which is the block that is using the CAB as an insertion program. In this case, block connections are created between the calling block and the inserted CAB blocks, during the load of insertion points configuration for configured parameter references. The parameter references ensure that data flow occurs in the proper sequence with respect to the execution of the calling block and the CAB program.

If your application calls for inserted CAB instances to share data with blocks other than the calling block, you can configure pin connections for custom data parameters on the CAB instance. If pin connections are configured, be aware that the data transfer operates as follows:

- Pin connections always transfer data into a CAB insertion program just before execution of the calling block.
- Pin connections always transfer data out of a CAB insertion program just before execution of the block that is pulling the CAB custom data parameter (CDP).

Insertion status and fail alarm for regulatory control blocks

The following table summarizes failure considerations when a CAB program encounters an exception or termination condition. When the program completes execution normally, the CAB instance returns an execution status of NORMAL. When any of the insertions is in a non-normal status, the insertion fail flag (INSFAILFL) is set to TRUE. The flag is reset to FALSE, when all the abnormal programs return to normal status

Failure Type	Considerations
Program Termination	CAB programs in general must not be repetitious. If repetitive calculations must be used, be sure you carefully design and thoughly test them to be sure that excessive execution time is not required. CAB programs that take longer than one half of the ACE base cycle (250ms), are automatically terminated. CAB programs can also terminate, if the CAB program consumes excessive memory during runtime. When a program terminates, the CAB instance issues a terminated status to the calling master. You must correct the program defect and reload the CAB instance before ractivating the control strategy. The CAB remains in the terminated condition until it is restarted by an operator.
Exception Failures	If the inserted CAB program encounters an exception failure condition, the program is terminated in the current cycle but will execute in the next cycle. The insertion fail flag (INSFAILFL) is set to TRUE. The flag will be reset to FALSE, when the program runs till completion and returns a normal status. Note that the CAB instance generates an exception event on the first encounter only and does not regenerate the event on the next execution, if it remains in the exception state

Handling of insertion failure on regulatory control block

If the CAB program inserted on a regulatory block fails (returns a non-normal status), the block sets its control initialization condition (INITMAN) to True and skips all the remaining processing of the block.

For example, if a Ctl_Alg insertion program fails, control output processing, windup processing, and so on is not performed. The parameter INITMAN is set to TRUE, which causes the regulatory control block to go to INIT state.

CAB insertion on regulatory control block configuration examples

Single CAB insertion

The following figure and its companion callout description table show a sample configuration that uses a PID block with an insertion type from a single CAB instance.

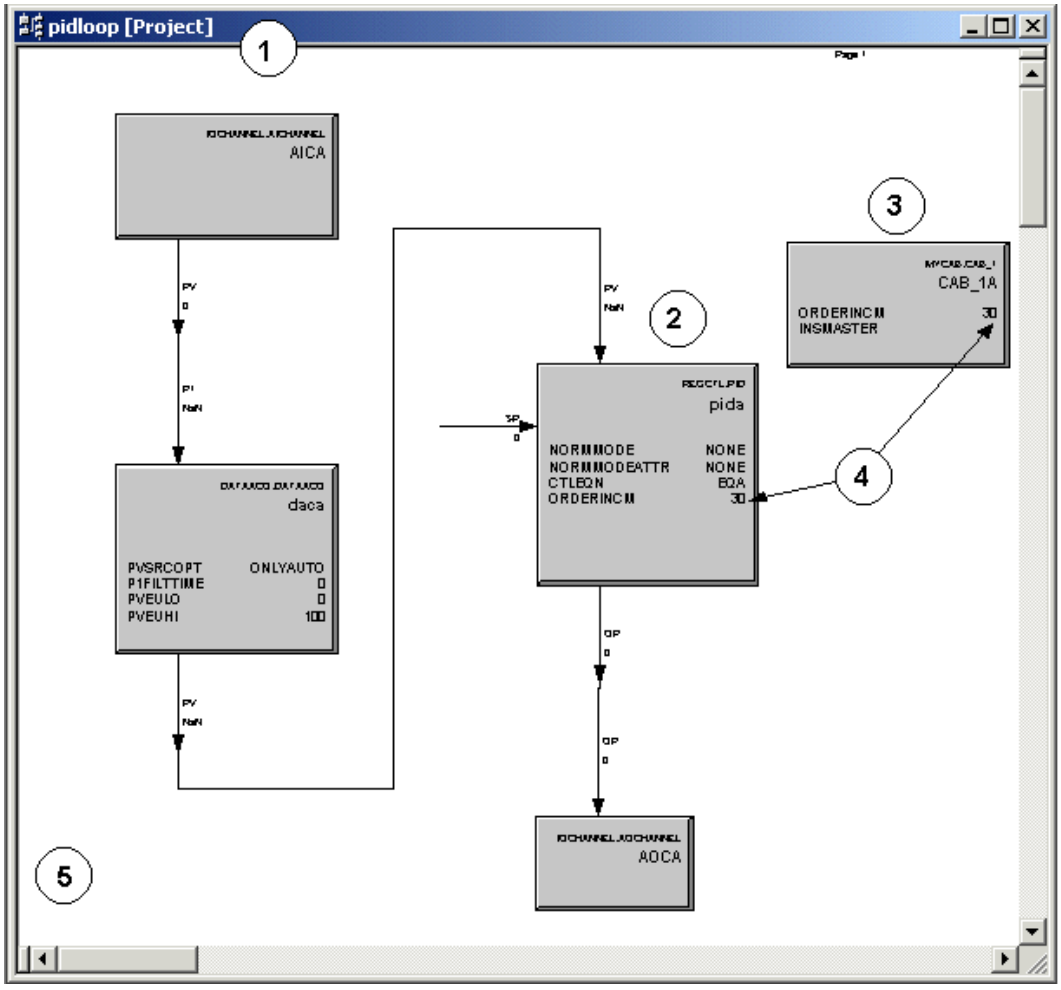


Figure 14 Configuration example using single CAB insertion for requlatory control block

Callout	Description
1	The Control Module contains a PID block named **pida**.
2	The **pida** block is configured to include an insertion type Ctl_Alg from a CAB instance named **CAB_1A**.
3	The CAB instance named **CAB_1A** is added to the Control Module containing the **pida** block.
4	During configuration, the ORDERINCM parameter for **CAB_1A** is changed to match the setting of ORDERINCM for the **pida** block. Also, the insertion point parameter (INSMASTER) for **CAB_1A** is turned On or set to True and shows the tag name of the **pida** block on the CAB configuration form.
5	Once the Control Module and its components is loaded to an ACE controller and activated, the **pida** block controls the execution of the **CAB_1A** instance as requried. If the **CAB_1A** instance runs successively with no failures, the cycle is repeated during every Control Module execution cycle. In this case, the **CAB_1A** instance returns a NORMAL status. During Control Module execution, only the **pida** block is executed.The **CAB_1A** instance executes only when the **pida** block calls the execute function. If the **CAB_1** instance is not used for an insertion point and the INSMASTER parameter is turned Off or set to False, it is included in the Control Module execution list and runs normally during each cycle. In this case, no tag name appears in the Insertion Point field on the block's configuration form and the CAB must be re-configured for an Access Level of PROGRAM.
Exception Failure Scenario	
If the CAB program encounters an exception failure (example divide by zero), it returns an exception status to the calling master **pida**. This causes **pida** to go into an INIT state, INITMAN is set to true, and it skips the rest of its processing. On the next execution cycle, **pida** remains in INIT unless the CAB returns a NORMAL status. When the exception condition clears, the **pida** block executes normally. Note that the CAB instance generates an exception event on the first encounter only and does not regenerate the event on the next execution, if it remains in the exception state.	
Termination Failure Scenario	

Callout	Description
If the CAB program encounters a termination failure (example execution time is too long), **CAB_1A** returns a termination status to the calling master **pida**. In this case, **pida** sets INITMAN to true and skips the rest of its processing. On the next execution cycle, **pida** will remain in INIT unless the problem has been corrected and the program returns a NORMAL status. With a termination failure, the CAB instance does not execute again unless manually restarted. You should correct the defect in the program before reactivating the control strategy. If you just try restarting a terminated CAB program, it will most likely terminate again.	

Multiple CAB insertions

The following figure and its companion callout description table show a sample configuration that uses a AUTOMAN block with insertion types from four CAB instances.

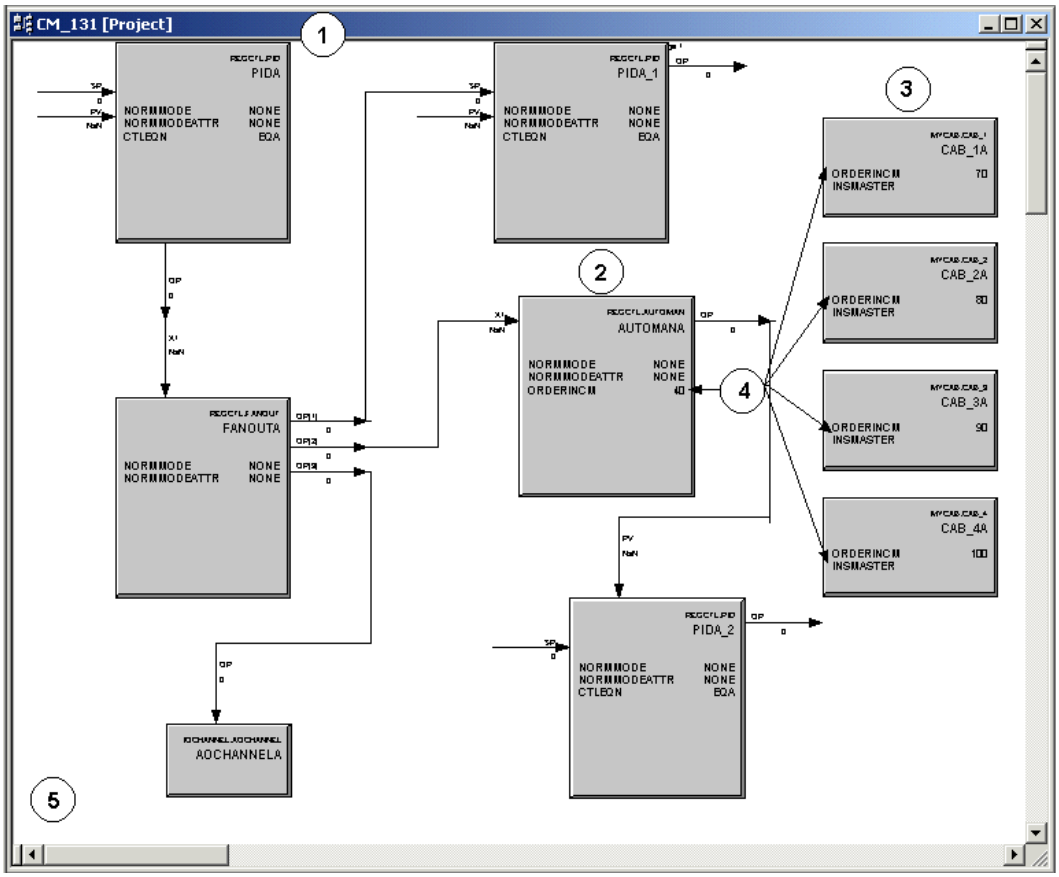


Figure 15 Configuration example using multiple CAB insertions for regulatory control block

Callout	Description
1	The Control Module contains a AUTOMAN block named **AUTOMANA**.
2	The **AUTOMANA** block is configured to include an insertion type Post_Input from a CAB instance named **CAB_1A**, insertion type Ctl_Alg from a CAB instance named **CAB_2A**, insertion type Post_Alg from a CAB instance named **CAB_3A**, and insertion type Post_Ctlout from a CAB instance named **CAB_4A**
3	The CAB instances named **CAB_1A**, **CAB_2A**, **CAB_3A**, and **CAB 4A** are added to the Control Module containing the **AUTOMANA** block.
4	During configuration, the ORDERINCM parameters for **CAB_1A, CAB_2A, CAB_3A,** and **CAB_4A** are changed to match the setting of ORDERINCM for the **AUTOMANA** block. Also, the insertion point parameters (INSMASTER) for all four CAB instances are turned On or set to True and show the tag name of the **AUTOMANA** block on their configuration forms.
5	Once the Control Module and its components is loaded to an ACE controller and activated, the **AUTOMANA** block controls the execution of all CAB instances as requried. If all CAB instances run successively with no failures, the cycle is repeated during every Control Module execution cycle. In this case, all CAB instances return a NORMAL status. If any of the CAB instances return a non-normal status, the insertion fail flag (INFAILFL) is set to TRUE and the INITMAN parameter of **AUTOMANA** block is set to TRUE, causing the block to go into its INIT state. The execution steps that follow the failure are skipped. If a CAB instance is not used for an insertion point and the INSMASTER parameter is turned Off or set to False, it is included in the Control Module execution list and runs normally during each cycle. In this case, no tag name appears in the Insertion Point field on the block's configuration form and the CAB must be re-configured for an Access Level of PROGRAM.
CAB _1A Failure Scenario	
If **CAB_1A** fails, none of the steps after input processing are executed in the current cycle including the execution of the other three insertion programs **CAB_2A**, **CAB_3A** and **CAB_4A**. This means that the output values (CV and OP) will remain unchanged and will not be recalculated based on the new input values fetched in the current cycle.	

Callout	Description
CAB _2A Failure Scenario	
If **CAB_2A** fails, the execution of **CAB_3A**, **CAB_4A**, control output processing and windup processing are skipped. If this happens, the CV value will remain at the old value	
CAB _3A Failure Scenario	
If **CAB_3A** fails, control output processing, windup processing and **CAB_4A** are skipped. In this case, the output value OP will not be computed to match the CV value calculated in the **CAB_2A** program.	
CAB _4A Failure Scenario	
If **CAB_4A** fails, windup and feedback processing are skipped. In this case, the windup states will not be recomputed and propagated to the primaries..	

Two CAB insertions of same type

The following figure and its companion callout description table show a sample configuration that uses a DATAACQ block with the same insertion type from two CAB instances.

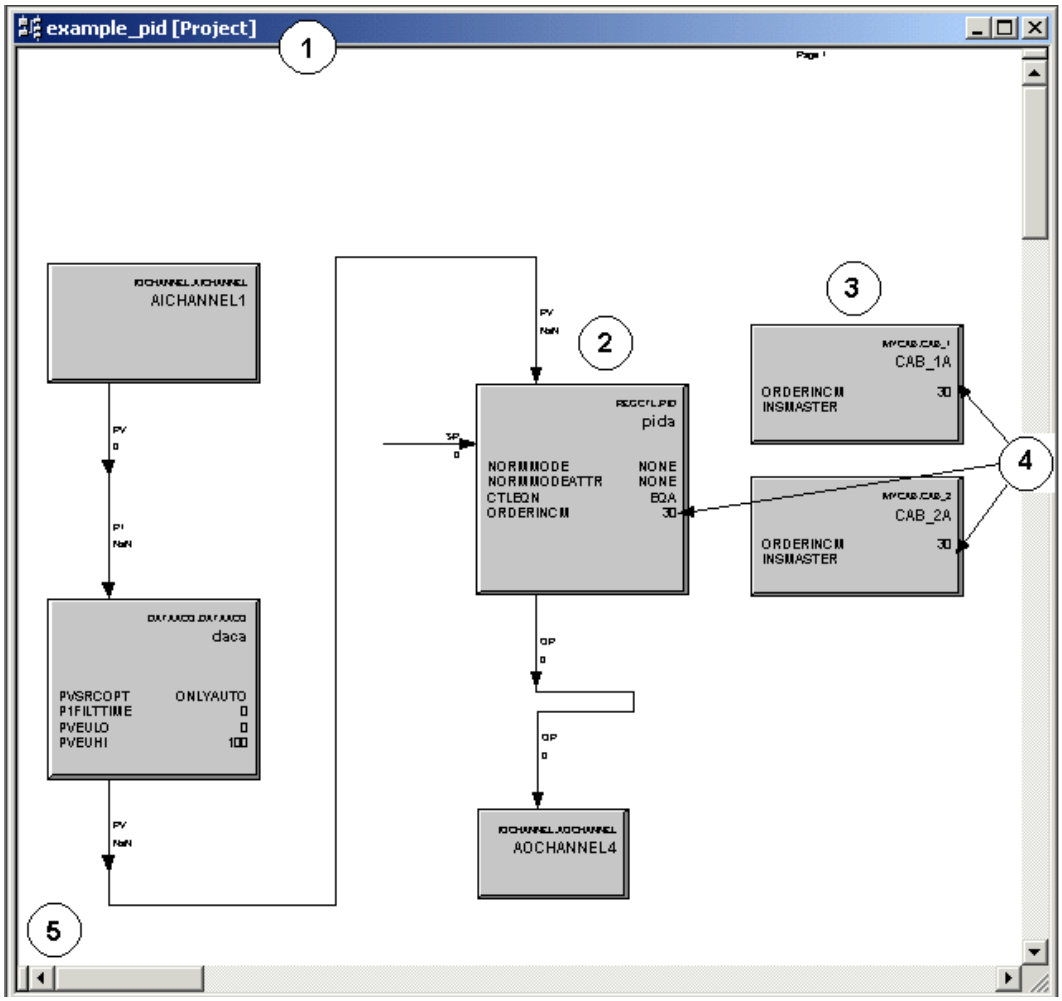


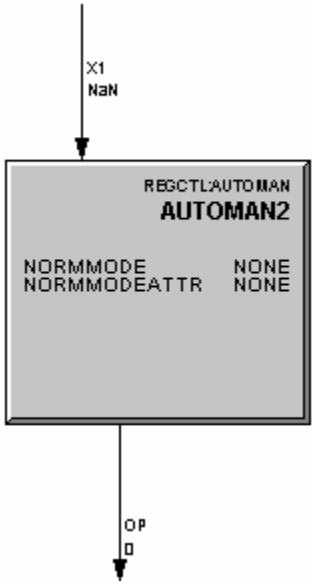
Figure 16 Configuration example using two CAB insetions of the same type for regulatory control block

Callout	Description
1	The Control Module contains a PID block named **pida**.
2	The **pida** block is configured to include insertion type Post_Alg from CAB instances named **CAB_1A** and **CAB_2A**
3	The CAB instances named **CAB_1A** and **CAB_2A** are added to the Control Module containing the **pida** block.
4	During configuration, the ORDERINCM parameters for **CAB_1A** and **CAB_2A** are changed to match the setting of ORDERINCM for the **pida** block. Also, the insertion point parameters (INSMASTER) for both CAB instances are turned On or set to True and the show the tag name of the **pida** block on their configuration forms.
5	Once the Control Module and its components is loaded to an ACE controller and activated, the **pida** block controls the execution of both CAB instances as requried. Since the CAB instances are both inserted at Post_Alg, the instances will be executed in the order in which they were configured. For example, if CAB instance **CAB_1A** was added to the CM berfore **CAB_2A**, CAB_1A is executed first. If both CAB instances run successively with no failures, the cycle is repeated during every Control Module execution cycle. In this case, both CAB instances return a NORMAL status. If either of the CAB instances returns a non-normal status, the insertion fail flag (INFAILFL) is set to TRUE and the INITMAN parameter of **pida** block is set to TRUE, causing the block to go into its INIT state. The execution steps that follow the failure are skipped. For example, if **CAB_1A** execution returns a non-normal status, **CAB_2A** will not be executed and the subsequent steps namely windup and feedback processing are skipped. If a CAB instance is not used for an insertion point and the INSMASTER parameter is turned Off or set to False, it is included in the Control Module execution list and runs normally during each cycle. In this case, no tag name appears in the Insertion Point field on the block's configuration form and the CAB must be re-configured for an Access Level of PROGRAM.

AUTOMAN (Auto Manual) Block

Description

The AUTOMAN (Auto Manual) block applies a user-specified gain and bias to the output. The user-specified values can be fixed or external. A fixed value is stored manually or by a program, and an external value comes from another function block. It looks like this graphically:



Each AUTOMAN block supports the following user configurable attributes. The following table lists the given name of the “Tab” in the parameter configuration form and then briefly describes the attributes associated with that Tab. This data is only provided as a quick document reference, since this same information is included in the on-line context sensitive Help.

Configuration Tab	Description
Main	• Name – Block (Tag) name of up to 16 characters long. Must be unique within the CM block containing it. • Description (DESC) – Block descriptor of up to 24 characters long. • Engineering Units (EUDESC) – Lets you specify a text string of up to 16 characters to identify the variable values associated with this block. For example, you could specify DEGF for temperature values in degrees Fahrenheit. This name is used on any associated displays and generated reports. • Execution Order in CM (ORDERINCM) – Specifies the execution order of the block in the CM relative to other blocks contained in this CM. Enter as a number between 1 to 32767. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. • High Limit (XEUHI) – Lets you specify the high input range limit that represents 100% full-scale input for the block. The default value is 100. • Low Limit (XEULO) – Lets you specify the low input range limit that represents the 0 full-scale input for the block. The default value is 0 (zero). • Normal Mode (NORMMODE) – Lets you specify the MODE the block is to assume when the Control to Normal function is initiated through the Station display. Selections are MANual, AUTOmatic, CAScade, BackupCAScade and NONE. All selections are not valid for a given block. The default selection is NONE. • Normal Mode Attribute (NORMMODEATTR) – Lets you specify the mode attribute (MODEATTR) the block is to assume when the Control to Normal function is initiated through the Station display. Selections are NONE, OPERATOR and PROGRAM. The default selection is NONE. • Mode (MODE) – Lets you set the block's current MODE. The selections are MANual, AUTOmatic, CAScade, BackupCAScade, NONE, and NORMAL. All selections are not valid for a given block. The default selection is MANual. MODE identifies who may store values to the

Configuration Tab	Description
	block's initializable inputs or output. Blocks strictly enforce the MODE assignment. Some function blocks perform automatic mode switching (or mode shedding), while others require manual intervention. The block's MODE is derived at "runtime" based on current conditions. MODE processing checks for the following conditions, and changes the block's MODE as appropriate. – External request for MODE switching. – Safety interlock request. • Mode Attribute (MODEATTR) – Lets you set the block's mode attribute. The selections are NONE, OPERATOR, PROGRAM, and NORMAL. The default selection is OPERATOR. MODEATTR identifies who may store values to the output (OP), when the block's MODE is MANual. • Permit Operator Mode Changes (MODEPERM) – Lets you specify if operators are permitted to make MODE changes or not. The default is Enabled (checked). A store to MODE does not change the NORMMODE. • Permit External Mode Switching (ESWPERM) – Lets you specify if external MODE switching through user configured interlocks is permitted or not, if you have at least an Engineering access level. The default is Disabled (unchecked). • Enable External Mode Switching (ESWENB) – Lets you specify if external MODE switching through user configured interlocks is enabled or not, if ESWPERM is checked (Permitted). The default is Disabled (unchecked). • Enable Secondary Initialization Option (SECINITOPT) – Lets you specify if the block is to ignore initialization and override requests from the secondary or not. The default selection is Enabled (checked, do not ignore). • Timeout Time (TMOUTTIME) – Lets you specify a time in seconds that must expire before the block assumes that its input update has timed out. The block must be in CAScade mode for it to monitor its primary input for timeout. The default setting is 0, which means the timeout function is disabled.

Configuration Tab	Description
	If the input is from a connection in another controller in a peer-to-peer architecture, the actual timeout time equals the configured TMOUTTIME plus the CDA timeout time. The CDA timeout time equals four times the configured CEE subscription rate. For example, if the CEE subscription rate is 100 milliseconds and the TMOUTTIME is 5 seconds, the actual timeout time for the block is 4 times 100ms plus 5s or 5.4 seconds. • Safety Interlock Option (SIOPT) – Lets you specify MODE and OP block is to assume upon a safety interlock alarm. The selections are NO_SHED, SHEDHOLD, SHEDLOW, SHEDHIGH, and SHEDSAFE. The default selection is SHEDHOLD. • Bad Control Option (BADCTLOPT) – Lets you specify MODE and OP block is to assume if CV goes BAD. The selections are NO_SHED, SHEDHOLD, SHEDLOW, SHEDHIGH, and SHEDSAFE. The default selection is NOSHED.
Output	• High Limit (%) (OPHILM) – Lets you specify the output high limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you enter a High Limit of 90%, the high limit in engineering units is 90% times 450 or 405 + 50 (CVEULO) equals 455. This check is not applied for a function block that is in the MANual mode. The default value is 105%. • Low Limit (%) (OPLOLM) – Lets you specify the output low limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you enter a Low Limit of 10%, the low limit in engineering units is 10% times 450 or 45 + 50 (CVEULO) equals 95. This check is not applied for a function block that is in the MANual mode. The default value is -5%. • Extended High Limit (%) (OPEXHILM) – Lets you specify the output extended high limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you use the default value of 106.9%, the extended high limit in engineering units is 106.9% times 450 or 481.05 + 50 (CVEULO) equals 531.05. This check is not applied for a function block that is in the MANual mode. The default

Configuration Tab	Description
	value is 106.9%. • Extended Low Limit (%) (OPEXLOLM) – Lets you specify the output extended low limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you use the default value of -6.9%, the extended low limit in engineering units is -6.9% times 450 or -31.05 + 50 (CVEULO) equals 18.95. This check is not applied for a function block that is in the MANual mode. The default value is -6.9%. • Rate of Change Limit (%) (OPROCLM) – Lets you specify a maximum output rate of change limit for both the positive and negative directions of the output in percent per minute. This lets you prevent an excessive rate of change in the output so you can match the slew rate of the control element to the control dynamics. We recommend that you configure this value before you tune the loop, so tuning can accommodate any slow down in response time caused by this rate limiting. This check is not applied for a function block that is in the MANual mode. The default value is Not a Number (NaN), which means no rate limiting is applied. • Minimum Change (%) (OPMINCHG) – Lets you specify an output minimum change limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). This lets you define how much the OP must change before the function block outputs a new value. It filters out changes that are too small for the final control element to respond to. This check is not applied for a function block that is in the MANual mode. The default value is 0, which means no change limiting is applied. • Safe OP (%) (SAFEOP) – Lets you specify the safe output value as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 0 to 500 and you enter a Safe OP of 50%, the safe output value in engineering units is 50% times 500 or 250. The default value is Not a Number (NaN), which means the OP is held at its last good value. • Output Bias (OPBIAS.FIX) – Lets you specify a fixed bias value in engineering units that is added to the Calculated Variable (CV) output value. See the Output Bias section for this function block for details. The default

Configuration Tab	Description
	value is 0, which means no value is added. • Output Bias Rate (OPBIAS.RATE) – Lets you specify an output floating bias ramp rate in engineering units per minute. This bias rate is only applied when the floating bias is non-zero. See the Output Bias section for this function block for details. The default value is Not a Number (NaN), which means no floating bias is calculated. As a result, if the primary does not accept the block's initialization value, a bump in OP occurs. • Gain (K) – Lets you specify a gain (K) value to be factored into the equation for calculating the CV output value. See the equation following this table for details. The default value is 1. • Gain High Limit (GAINHILM) – Lets you specify gain high limit value. Gain (K) is clamped to this value, if the specified gain exceeds it. The default value is 240. • Gain Low Limit (GAINLOLM) – Lets you specify gain low limit value. Gain (K) is clamped to this value, if the specified gain is less than it is. The default value is 0.
Alarms	• Type – Identifies the types of alarm this block supports. Of course, these alarms also interact with other block configuration options such as the Safety Interlock Option (SIOPT) and Bad Control Option (BADCTLOPT). The types are: – Safety Interlock (SIALM.FL) – Bad Control (BADCTLALM.FL) – OP High (OPHIALM.FL) – OP Low (OPLOALM.FL) • Enable Alarm (SIALM.OPT) – Lets you enable or disable Safety Interlock alarm type. A check in the box means the alarm is enabled. The default selection is a checked box or enabled (Yes). You can also configure the SIALM.OPT parameter as a block pin, a configuration and/or a monitoring parameter so it appears on the block in the Project and Monitoring tree view, respectively. • Trip Point – Lets you specify the OP High Alarm (OPHIALM.TP) and OP Low Alarm (OPLOALM.TP) trip

Configuration Tab	Description
	points in percent. The default value is NaN, which disables the trip point. • Priority – Lets you set the desired priority level individually for each alarm type (SIALM.PR, BADCTLALM.PR, OPHIALM.PR, and OPLOALM.PR). The default value is LOW. The levels are: – NONE - Alarm is neither reported nor annunciated. – JOURNAL - Alarm is logged but it does not appear on the Alarm Summary display. – LOW, HIGH, URGENT - Alarm is annunciated and appears on the Alarm Summary display. • Severity – Lets you assign a relative severity individually for each alarm type (SIALM.SV, BADCTLALM.SV, OPHIALM.SV, and OPLOALM.SV) as a number between 0 to 15, with 15 being the most severe. This determines the alarm processing order relative to other alarms. The default value is 0. • Deadband Value (ALMDB) – Lets you specify a deadband value that applies to all analog alarms to prevent nuisance alarms due to noise at values near the trip point. The default value is 1. Note that this value is loaded to the individual alarm parameters (for example, OPHIALM.DB and OPLOALM.DB) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB. • Filter Time (ALMTM) – Lets you specify a time in seconds to define how long an analog alarm must exist before it is set true. The default value is 0, which means the alarm is set true as soon as the value exceeds the deadband value. Note that this value is loaded to the individual alarm parameters (for example, OPHIALM.TM and OPLOALM.TM) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB. • Deadband Units (ALMDBU) – Lets you specify if the

Configuration Tab	Description
	deadband value represents percent or engineering units. The default value is percent. Note that this value is loaded to the individual alarm parameters (for example, OPHIALM.DBU and OPLOALM.DBU) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB.
SCM	• SCM Mode Tracking Option (MODETRACK) – Lets you select the desired Mode Tracking function for the SCM associated with this block's Control Module. It defines how the FB will set the state of the MODEATTR based upon the MODE of the SCM. See the *Sequential Control Module User's Guide* for more information on this function. The default selection is ONESHOT. The selections are: – None – ONESHOT – SEMICONT – CONTRTN – CONT • Option Type – Lets you specify the action the function block is to take when the SCM goes into an abnormal state. The Starting State Option (STARTOPT) applies when the SCM state is Checking, Idle, or Complete. The Stop/Abort State Option (STOPOPT) applies when the SCM state is Stopping or Stopped, Aborting or Aborted. The Hold State Option (HOLDOPT) applies when the SCM state is Holding or Hold. The Restart State Option (RESTARTOPT) applies when the SCM state is Resume or Run. The NONE and LASTREQ are the only selections for the Restart State Option. You can select from one of these types for the other options as applicable for the given regulatory control function block: – NONE - No changes. – MAN - Set MODEREQ = MANUAL. – AUTO - Set MODEREQ = AUTOMATIC (not applicable to this block).

Configuration Tab	Description
	– CAS - Set MODEREQ = CASCADE. – FIXEDOP - Set OPREQ = Configured Value. – HOLDPV - Set SPREQ = PV. – FIXED SP - Set SPREQ = Configured Value and SPRATEREQ = NaN. – RAMPEDSP - Set SPTVREQ = Configured Value and SPRATEREQ = Configured Rate. • Value (STARTVAL, STOPVAL, HOLDVAL) – Depending upon Option Type selection, lets you specify an output or set point value within the respective range. For output, within OPLOLM to OPHILM and within SPLOLM to SPHILM, for set point. The default value is NaN (Not a Number).
Block Pins	Lets you select the available parameters that you want to expose as input/output pins on the function block graphic in Control Builder.
Configuration Parameters	Lets you select the available parameters that you want to appear on the face of the function block in the Project tab in Control Builder.
Monitoring Parameters	Lets you select the available parameters that you want to appear on the face of the function block in the Monitoring tab in Control Builder.
Block Preferences	Lets you change several block-viewing preferences including the color of the block's faceplate.
Insertion Type	Lets you include an insertion type from a CAB instances in the block. See *CAB insertion configuration considerations for regulatory control blocks* for more information

The block calculates the output value (CV) using the following equation:

$CV = K * X1 + OPBIAS.FIX + OPBIAS.FLOAT$

Where:

K	=	gain for CV (user specified)
X1	=	input value
OPBIAS.FIX	=	fixed output bias (user specified)
OPBIAS.FLOAT	=	floating output bias (calculated)

- K and OPBIAS.FIX may either be fixed (that is, stored manually or by the program) or external (that is, brought from another block).
- After an initialization, the block calculates OPBIAS.FLOAT as follows:

  $OPBIAS.FLOAT = CV_{INIT} - [K * X1 + OPBIAS.FIX]$

  where: CV_{INIT} = initialization value from the secondary

Function

The AUTOMAN block is typically used:

- in a cascade control strategy where one of the upstream blocks may not accept an initialization request from its secondary.
- between a FANOUT block and a final control element to provide a "bumpless" output on return to cascade

ATTENTION

The AUTOMAN block:

- has one primary and one secondary.
- requests the primary to initialize when mode changes from CAScade to MANual.

Configuration example

The following figure and its companion callout description table show a sample configuration that uses an AUTOMAN block between a FANOUT block and a downstream PID block for quick reference.

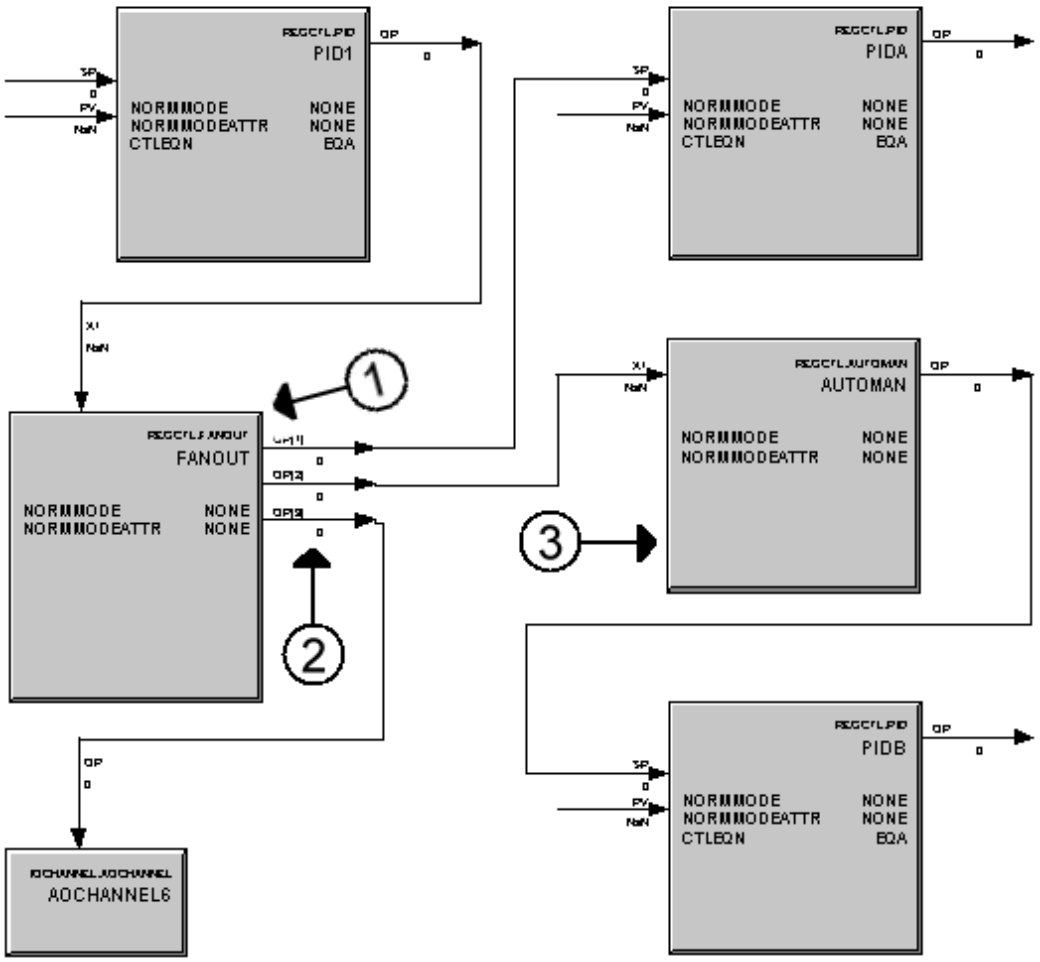


Figure 17 Example of CB configuration using AUTOMAN block.

The following table includes descriptions of the callouts in the figure above.

Callout	Description
1	You can use the FANOUT block to distribute a single primary output to multiple secondaries. (Note that the individual BACKCALCIN/BACKCALCOUT connections for each FANOUT output used are automatically built by Control Builder as implicit/hidden connections.) Since the FANOUT block only initializes when all of its secondaries request it, insert an AUTOMAN block for individual downstream blocks (like PIDB in this example) to ensure bumpless transfer during mode changes.
2	You can specify a gain and bias for each of the FANOUT block outputs.
3	The primary purpose of this AUTOMAN block is to ensure a bumpless output upon return to Cascade mode. The AUTOMAN block is typically used between a FANOUT block and a final control element. Upon a return to Cascade, each secondary provides an initialization request to its primary. In most cases, the primary adjusts its output accordingly. However, if the primary is a FANOUT block, it may ignore the initialization request, since all of its secondaries may not be requesting it. In this case, the AUTOMAN block compensates for this by applying a floating bias to the output. This block applies a user-specified gain and bias to the output. The user-specified values can be fixed or external. A fixed value is stored manually or by a program, and an external value comes from another function block. The AUTOMAN block uses the following equation to calculate its output. • CV = K * X1 + OPBIAS.FIX + OPBIAS.FLOAT • where: – K = gain for CV – X1 = input value – OPBIAS.FIX = fixed output bias (user-specified) – OPBIAS. FLOAT = floating output bias (calculated)

Inputs

The AUTOMAN block requires one input – X1:

- X1 = initializable input which, if used, must be pulled from another block (it cannot be stored to).
- An engineering unit range for X1 (XEUHI and XEULO) must be specified.
- XEUHI and XEULO define the full range of X1:
  - XEUHI represents the 100% of full-scale value.
  - XEULO represents the 0% of full-scale value.

(Note that this block applies no range checks and assumes that X1 is within the XEUHI and XEULO range.)

Output

The AUTOMAN block has the following initializable outputs:

- OP = calculated output, in percent.
- OPEU = calculated output, in engineering units.

ATTENTION

A connection to OP or OPEU may be created, but not to both. Therefore, this block may have only one secondary. If a connection to OP or OPEU is not created, the AUTOMAN block does not have a secondary. Alternately, if OP or OPEU is connected to a non-initializable input, the AUTOMAN block does not have a secondary.

The default OP connection pin is exposed on the blocks and the implicit/hidden connection function automatically makes the appropriate value/status parameter connection when required. For example, if you connect the output from an AUTOMAN block (AUTOMAN1.OP) to the set point of a PID block (PIDB.SP), the implicit/hidden connection is made to AUTOMAN1.OPX to provide value/status data.

Initializable inputs and outputs

"Initializable input" and "initializable output" are variable attributes, similar to data type or access level. A parameter with the "initializable" attribute has an associated BACKCALC parameter. When a connection between an initializable input and initializable output is created, you can also create a BACKCALC connection between them. Control Builder automatically builds the required BACKCALC connections, so you don't have to create them manually. These "implicit" build connections are "hidden" from view and the related parameter pins are not exposed on the control chart.

For example, if you connect OP from an AUTOMAN block to a PID block or an AOCHANNEL block, Control Builder automatically creates the BACKCALCOUT to BACKCALCIN connection.

ATTENTION

The AUTOMAN block provides the X1 input range (XEUHI/XEULO) to the primary through BACKCALC. The primary uses this for its output range (CVEUHI/CVEULO).

Output ranges

CVEUHI and CVEULO define the full range of CV in engineering units.

If the AUTOMAN block has a secondary, it brings the secondary's input range through BACKCALC and sets its CV range to that. If it has no secondary, CVEUHI and CVEULO track the X-input range (XEUHI and XEULO).

- OPHILM and OPLOLM define the normal high and low limits for OP as a percent of the CV range. These are user-specified values.

  OP is clamped to these limits if the algorithm's calculated result (CV) exceeds them or another block or user program attempts to store an OP value that exceeds them. However, the operator may store an OP value that is outside these limits.

- OPEXHILM and OPEXLOLM define the extended high and low limits for OP as a percent of the CV range. These are user-specified values. The operator is prevented from storing an OP that exceeds these limits.

Output bias

The output bias (OPBIAS) is added to the algorithm's Calculated Value (CV) and the result is stored in CV. CV is later checked against OP limits and, if no limits are exceeded, copied to the output.

The OPBIAS is the sum of the user-specified fixed bias (OPBIAS.FIX) and a calculated floating bias (OPBIAS.FLOAT). The purpose of the floating bias is to provide a bumpless transfer when the function block initializes or changes mode.

- OPBIAS is recomputed under the following conditions to avoid a bump in the output. (Note that the function block only applies OPBIAS.FLOAT to the output for the latter two conditions, when it is the first initializable block.)
  - When the function block starts up (that is, goes Active).
  - When the function block initializes (for example, the secondary requests initialization).
  - When the mode changes to Cascade.

ATTENTION

When the mode is Manual, OPBIAS is not used, since the output is not calculated. This means that the OPBIAS is not recomputed, when the mode changes to Manual.

- You may set the OPBIAS value only if the function block is Inactive or Mode equals Manual. This is done to prevent a bump in the output when the bias is changed. The following occurs when you set the OPBIAS value.
  - The total bias (OPBIAS) and fixed bias (OPBIAS.FIX) are both set to the entered value.
  - The floating bias (OPBIAS.FLOAT) is set to zero.

ATTENTION

When the function block goes Active or the Mode changes to Auto or Cascade, OPBIAS and OPBIAS.FLOAT are recomputed.

- There are no limit checks applied when you set an OPBIAS value. However, after the bias is added to CV, the result is compared against the output limits and clamped, if necessary.

- You configure the value for the fixed bias (OPBIAS.FIX) and it is never overwritten by the floating bias (OPBIAS.FLOAT). This means the total bias will eventually equal the OPBIAS.FIX , if you configure OPBIAS.RATE to ramp down OPBIAS.FLOAT.
- The OPBIAS.FLOAT is calculated as follows.

$$OPBIAS.FLOAT = CV_{INIT} - (CV_{UNBIASED} + OPBIAS.FIX)$$

Where:

CV_{INIT} = initialization value received from the secondary

$CV_{UNBIASED}$ = unbiased calculated value (based on input from the primary)

OPBIAS.FIX = fixed bias (user-specified)

- If the primary accepts this block's initialization request, then CV + OPBIAS.FIX should be the same as CVINIT and OPBIAS.FLOAT will be zero. In most cases, OPBIAS.FLOAT will be zero. However, if the primary does not accept this block's initialization request because the primary is a FANOUT block or it was configured to ignore initialization, then OPBIAS.FLOAT value will not be zero.

  If OPBIAS.FLOAT is not zero, you can configure it to ramp down to zero through the OPBIAS.RATE parameter.
- You configure the OPBIAS.RATE to apply a ramprate to the OPBIAS.FLOAT. It is only used when the OPBIAS.FLOAT is not zero. The OPBIAS.RATE is expressed in Engineering Units per minute and may have the following values.
  - Zero:
    If OPBIAS.RATE is zero, a OPBIAS.FLOAT is calculated and bumpless transfer is guaranteed. However, if OPBIAS.FLOAT is not zero, it will never ramp down. As previously mentioned, you can reset the OPBIAS.FLOAT to zero by manually entering a value for OPBIAS.
  - Non-zero:
    If OPBIAS.RATE is not zero, an OPBIAS.FLOAT is calculated and bumpless transfer is guaranteed. If the OPBIAS.FLOAT is not zero, it is ramped to zero at the rate you configured for the OPBIAS.RATE parameter.

    The function block ramps the OPBIAS.FLOAT to zero by applying the following calculation each time it executes.

OPBIAS.FLOAT = OPBIAS.FLOAT – (OPBIAS.RATE / cycles_per_Min)

Where:

cycles_per_min = number of times the function block executes per minute (calculated)

- NaN:
  When the OPBIAS.RATE is Not a Number (NaN), no OPBIAS.FLOAT is calculated. This means a bump in the output will occur, if the primary does not accept this block's initialization value.

Mode Handling

The AUTOMAN block supports both the Cascade and Manual modes:

- If Mode is CAScade: X1 must come from another block.
- If Mode is MANual: an operator or a user program (X1 is ignored) may store OP.

Timeout Monitoring

If mode is CAScade, the AUTOMAN block performs timeout monitoring on X1. If the X1 value is not updated within a predefined time (TMOUTTIME), the AUTOMAN block invokes timeout processing as follows:

- Sets the "input timeout" flag (TMOUTFL).
- Sets the input value to Bad (NaN – Not a Number).
- Requests the X1 primary to initialize.

Note that the AUTOMAN block does not support mode shedding on timeout.

The maximum time between updates is specified by TMOUTTIME (in seconds)

- Enable timeout monitoring by setting TMOUTTIME to a non-zero value.
- Disable timeout monitoring by setting TMOUTTIME to zero.

ATTENTION

If the input is from a connection in another controller in a peer-to-peer architecture, the actual timeout time equals the configured TMOUTTIME plus the CDA timeout time. The CDA timeout time equals four times the configured CEE subscription rate. For example, if the CEE subscription rate is 100 milliseconds and the TMOUTTIME is 5 seconds, the actual timeout time for the block is 4 times 100ms plus 5s or 5.4 seconds.

Control Initialization

The AUTOMAN block brings initialization requests from its secondary through BACKCALC. In addition, the secondary may propagate one-shot initialization requests to the block.

If the secondary is requesting initialization, the AUTOMAN block –

- Initializes its output so CV = initialization value from the secondary
- Builds an initialization request for the X1 primary as follows:

$$\text{INITREQ(X1)} = \text{On}$$

$$\text{INITVAL(X1)} = \frac{\text{CV - OPBIAS.FIX}}{\text{K}}$$

  - Where:

INITREQ(X1) = initialization request flag for the X1 primary

INITVAL(X1) = initialization value for the X1 primary

CV = calculated value

K = Output gain

OPBIAS.FIX = user specified fixed bias

ATTENTION

Following a return to cascade, each secondary provides an initialization request to its primary and in most cases the primary adjusts its output accordingly. However, if the primary is a FANOUT block, it may ignore the initialization request. The AUTOMAN block compensates for this by applying a floating bias to the output.

Secondary initialization option

If a BACKCALC connection is made, the primary always brings initialization data over this connection. However, you can configure the block to ignore this data by not selecting the Enable Secondary Initialization Option on the block's parameter configuration form. This is the same as selecting disable as the setting for the SECINITOPT parameter. The results of the SECINITOPT settings are as follows.

- If SECINITOPT equals Enable, it means the function block should accept initialization and override requests from the secondary.
- If SECINITOPT equals Disable, it means the function block should ignore initialization and override requests from the secondary.

Override feedback processing

If the AUTOMAN block is in a Cascade strategy with a downstream OVRDSEL (Override Selector) block, it receives override feedback data. This data consists of an override status, override feedback value and an override offset flag (for PID block strategies). The status indicates if the block is in the selected or unselected strategy (as determined by the OVRDSEL block).

When the override status changes from selected to unselected, the AUTOMAN block:

- does not initialize its CV, and
- computes a feedback value for its primary as follows:

  feedback value for primary = (override feedback value from secondary – OPBIAS.FIX – OPBIAS.FLOAT) / K

Windup processing

Every regulatory control type block maintains anti-reset windup status for its output (ARWOP) and each of its initializable inputs (ARWNET). The following table lists the posssible values for ARWOP and ARWNET parameters.

If the Value is . . .	Then, the Associated Parameter . . .
Normal	is free to move in either direction.
Hi	is at its high limit and it may only be lowered.
Lo	is at its low limit and it may only be raised.
HiLo	may not move in either direction.

Manual Mode Interaction

When the MODE of a regulatory control block is changed to Manual (Man), the block sets its windup status (ARWNET) to HiLo. This means that every block upstream in a cascade strategy will set its windup status (ARWNET and ARWOP) to HiLo.

ARWOP computation

The ARWOP indicates if the output (OP) can be raised or lowered. The PID-type function blocks use ARWOP to restrict integral control. When ARWOP contains a value other than **Normal**, the PID block stops integral control in the windup direction. Integral control continues in the other direction, as does proportional and derivative control. But, windup status has no impact on proportional and derivative control.

If a function block has a secondary, it fetches the secondary's windup status and recomputes its ARWOP. The conditions within the function block, such as output being at its high limit, also affect the ARWOP. The ARWOP is computed as follows, assuming the block has only one output or that it is not a FANOUT block.

If Any of the Following are True . . .	Then, ARWOP Equals . . .
This block is inactive.	**HiLo**
A secondary exists but this block cannot fetch secondary data from it (communications or configuration error).	

If Any of the Following are True . . .	Then, ARWOP Equals . . .
A secondary exists and its windup state equals HiLo	
This block is in initialization (INITMAN = On).	
A secondary exists and it is requesting this block to initialize.	
A secondary exists and its windup state equals Hi.	**Hi**
This block's output is at its high limit (OPHIFL = On).	
A secondary exists and its windup state equals Lo.	**Lo**
This block's output is at its low limit (OPLOFL = On).	

ARWNET computation

When ARWNET is HiLo, stores to SP are not limited, rather this is the status propagated to the primary. The only *limiting* anti-reset windup status ever does is to stop integral action in one or both directions on PID blocks. For any other regulatory control type block, ARWNET is not used for any kind of limiting. The ARWNET is computed as follows, assuming the block has only one output or that it is not a FANOUT block.

If Any of the Following are True . . .	Then, ARWNET Equals . . .
This block is inactive.	**HiLo**
The ARWOP equals HiLo.	
This block is in Manual mode (MODE = Man)	
The calculated value (CV) range (CVEUHI / CVEULO) is NaN.	
The CV is NaN	
This block is connected to a non-initializable primary	

If Any of the Following are True . . .	Then, ARWNET Equals . . .
The ARWOP equals Hi (Pid function blocks have a configurable Control Action option (CTLACTN). If CTLACTN = Reverse , ARWNET will track ARWOP; but if CTLACTN = Direct , ARWNET will be the opposite of ARWOP.)	**Hi**
The input from the primary is at a high limit. For example, SPHIFL = On.	
This block's output has reached its positive rate-of-change limit (OPROCPOSFL = On)	
The ARWOP equals Lo (Pid function blocks have a configurable Control Action option (CTLACTN). If CTLACTN = Reverse , ARWNET will track ARWOP; but if CTLACTN = Direct , ARWNET will be the opposite of ARWOP.)	**LO**
Input from the primary is at a low limit. For example, SPLO.FL = On.	
This block's output has reached its negative rate-of-change limit (OPROCNEGFL = On)	

Anti-Reset Windup Status

The anti-reset windup network in (ARWNETIN) and anti-reset windup output in (ARWOPIN) parameters are added in the standard anti-reset windup (ARW) computation logic. They are user configurable and allow stores from Sequential Control Modules (SCMs) and control algorithm block (CAB) programs.

The ARWNETIN and ARWOPIN parameters would be *ORed* into the existing standard logic so it is not lost. The following table summarizes the influence the ARWNETIN and ARWOPIN parameters have on the ARWNET and ARWOP parameters, which are not user configurable.

ARWNETIN or ARWOPIN Parameter Is. . .	Standard Computation Logic Is . . .	ARWNET or ARWOP Parameter Is . . .
NORMAL	NORMAL	NORMAL
NORMAL	HI	HI
NORMAL	LO	LO
NORMAL	HILO	HILO
HI	NORMAL	HI
HI	HI	HI
HI	LO	HILO
HI	HILO	HILO
LO	NORMAL	LO
LO	HI	HILO
LO	LO	LO
LO	HILO	HILO
HILO	NORMAL	HILO
HILO	HI	HILO
HILO	LO	HILO
HILO	HILO	HILO

AUTOMAN parameters

REFERENCE - INTERNAL

Refer to the *Control Builder Components Reference* for a complete list of the parameters used with the AUTOMAN block.

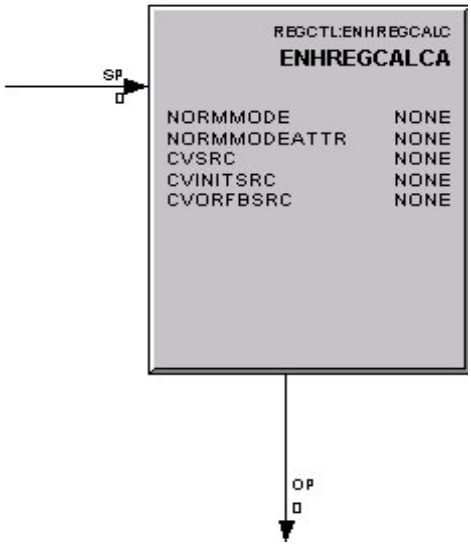
ENHREGCALC (Enhanced Regulatory Control Calculator) Block

Description

The ENHREGCALC block provides the following enhancements over the REGCALC block.

- Expands existing arrayed input parameters XSTS and X from six to ten.
- These arrayed parameters are added to correspond to each of the ten inputs.
  - Input Description
  - Enable/Disable Switch
  - XSUB Substitute Parameter
- An initializable Set Point (SP) input parameter with limit checking and SP ramping is added. Also, the X[1] input is converted to a general purpose input
- Mode can be placed in Automatic so operator or program can supply SP.
- Both the ENHREGCALC and REGCALC blocks are optimized so that expressions use memory based on the number of expressions configured, pcode size of each expression and the number of references in the expression.

It looks like this graphically.



Each ENHREGCALC block supports the following user configurable attributes. The following table lists the given name of the "Tab" in the parameter configuration form and then briefly describes the attributes associated with that Tab. This data is only provided as a quick document reference, since this same information is included in the on-line context sensitive Help.

Configuration Tab	Description
Main	• Name – Block (Tag) name of up to 16 characters long. Must be unique within the CM block containing it. • Description (DESC) – Block descriptor of up to 24 characters long. • Engineering Units (EUDESC) – Lets you specify a text string of up to 16 characters to identify the variable values associated with this block. For example, you could specify DEGF for temperature values in degrees Fahrenheit. This name is used on any associated displays and generated reports. • Safety Interlock Option (SIOPT) – Lets you specify MODE and OP block is to assume upon a safety interlock alarm. The selections are NO_SHED, SHEDHOLD, SHEDLOW, SHEDHIGH, and SHEDSAFE. The default selection is SHEDHOLD.

Configuration Tab	Description
	• Bad Control Option (BADCTLOPT) – Lets you specify MODE and OP block is to assume if CV goes BAD. The selections are NO_SHED, SHEDHOLD, SHEDLOW, SHEDHIGH, and SHEDSAFE. The default selection is NOSHED. • Execution Order in CM (ORDERINCM) – Specifies the execution order of the block in the CM relative to other blocks contained in this CM. Enter as a number between 1 to 32767. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. • Normal Mode (NORMMODE) – Lets you specify the MODE the block is to assume when the Control to Normal function is initiated through the Station display. Selections are MANual, AUTOmatic, CAScade, BackupCAScade and NONE. All selections are not valid for a given block. The default selection is NONE. • Normal Mode Attribute (NORMMODEATTR) – Lets you specify the mode attribute (MODEATTR) the block is to assume when the Control to Normal function is initiated through the Station display. Selections are NONE, OPERATOR, and PROGRAM. The default selection is NONE. • Mode (MODE) – Lets you set the block's current MODE. The selections are MANual, AUTOmatic, CAScade, BackupCAScade, NONE, and NORMAL. All selections are not valid for a given block. The default selection is MANual. MODE identifies who may store values to the block's initializable inputs or output. Blocks strictly enforce the MODE assignment. Some function blocks perform automatic mode switching (or mode shedding), while others require manual intervention. The block's MODE is derived at "runtime" based on current conditions. MODE processing checks for the following conditions, and changes the block's MODE as appropriate. – External request for MODE switching. – Safety interlock request. • Mode Attribute (MODEATTR) – Lets you set the block's mode attribute. The selections are NONE, OPERATOR,

Configuration Tab	Description
	PROGRAM, and NORMAL. The default selection is OPERATOR. MODEATTR identifies who may store values to the output (OP), when the block's MODE is MANual. • Permit Operator Mode Changes (MODEPERM) – Lets you specify if operators are permitted to make MODE changes or not. The default is Enabled (checked). A store to MODE does not change the NORMMODE. • Permit External Mode Switching (ESWPERM) – Lets you specify if external MODE switching through user configured interlocks is permitted or not, if you have at least an Engineering access level. The default is Disabled (unchecked). • Enable External Mode Switching (ESWENB) – Lets you specify if external MODE switching through user configured interlocks is enabled or not, if ESWPERM is checked (Permitted). The default is Disabled (unchecked). • Enable Secondary Initialization Option (SECINITOPT) – Lets you specify if the block is to ignore initialization and override requests from the secondary or not. The default selection is Enabled (checked, do not ignore).
Setpoint	• SP (SP) – Lets you specify an initial set point value. The default value is 0. • High Limit (SPHILM) – Lets you specify a high limit value for the SP. If the SP value exceeds this limit, the block clamps the SP to the limit value and sets the SP high flag (SPHIFL). The default value is 100. • Low Limit SPLOLM) – Lets you specify a low limit value for the SP. If the SP value falls below this limit, the block clamps the SP to the limit value and sets the SP low flag (SPLOFL). The default value is 0. • Setpoint Engineering Units High Range (SPEUHI) – Lets you specify the high SP input range value in Engineering Units that represents 100% full scale input. The default value is 100. • Setpoint Engineering Units Low Limit (SPEULO) – Lets you specify the low SP input range value in Engineering Units that represents the 0 full scale input. The default

Configuration Tab	Description
	value is 0 (zero). • Mode (TMOUTMODE) – Lets you select the desired MODE the block is to assume, if an initializable input times out, which means the input has not been updated within a designated timeout time. The selections are AUTOmatic, BCAScade, CAScade, MANual, NONE, and NORMAL. The default selection is MANual. • Time (TMOUTTIME) – Lets you specify a time in seconds that must expire before the block assumes that its input update has timed out. The block must be in CAScade mode for it to monitor its primary input for timeout. The default setting is 0, which means the timeout function is disabled. If the input is from a connection in another controller in a peer-to-peer architecture, the actual timeout time equals the configured TMOUTTIME plus the CDA timeout time. The CDA timeout time equals four times the configured CEE subscription rate. For example, if the CEE subscription rate is 100 milliseconds and the TMOUTTIME is 5 seconds, the actual timeout time for the block is 4 times 100ms plus 5s or 5.4 seconds. • Enable SP Ramping (SPTVOPT) – Lets you specify if an operator can initiate a set point ramp action or not. It provides a smooth transition from the current set point value to a new one. The default selection is box unchecked (disabled). See the Set point ramping section for this block for more details. • Normal Ramp Rate (SPTVNORMRATE) – Lets you specify a ramp rate in engineering units per minute for the SP ramping function, when it is enabled. This lets an operator start the SP ramping function without specifying a ramp time. The default selection is Not a Number (NaN). See the Set point ramping section for this block for more details. • Enable SP Push: (PUSHSP) – Lets you specify that the ENHREGCALC SP will be pushed from an Inter Cluster Gateway when the ENHREGCALC is the secondary of a cascade that extends over two Experion clusters.
Input	• X Descripiton (XDESC[1..10]) – Lets you enter a description for a given X input. The default value is null or blank.

Configuration Tab	Description
	• X Enable (XENABLE[1..10]) – Lets you select whether a given X input is enabled or disabled. The default value is selected on enabled. • X Substitute (XSUB[1..10]) – Lets you specify a substitute value to used for the given X input. The default setting is NaN. • XK (XK[1..10]) – Lets you specify an individual gain value for each of the ten X inputs. The default value is 1. • XB (XB[1..10]) – Lets you specify an individual bias value for each of the ten X inputs. The default value is 0.00, which means no bias is added. • XKB (XKB[1..10] – Lets you view the individual input values after individual biases and gains have been applied to X[1] through X[10]
Output	• High Limit (%) (OPHILM) – Lets you specify the output high limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you enter a High Limit of 90%, the high limit in engineering units is 90% times 450 or 405 + 50 (CVEULO) equals 455. This check is not applied for a function block that is in the MANual mode. The default value is 105%. • Low Limit (%) (OPLOLM) – Lets you specify the output low limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you enter a Low Limit of 10%, the low limit in engineering units is 10% times 450 or 45 + 50 (CVEULO) equals 95. This check is not applied for a function block that is in the MANual mode. The default value is -5%. • Extended High Limit (%) (OPEXHILM) – Lets you specify the output extended high limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you use the default value of 106.9%, the extended high limit in engineering units is 106.9% times 450 or 481.05 + 50 (CVEULO) equals 531.05. This check is not applied for a function block that is in the MANual mode. The default value is 106.9%.

Configuration Tab	Description
	• Extended Low Limit (%) (OPEXLOLM) – Lets you specify the output extended low limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you use the default value of -6.9%, the extended low limit in engineering units is -6.9% times 450 or -31.05 + 50 (CVEULO) equals 18.95. This check is not applied for a function block that is in the MANual mode. The default value is -6.9%. • Rate of Change Limit (%) (OPROCLM) – Lets you specify a maximum output rate of change limit for both the positive and negative directions of the output in percent per minute. This lets you prevent an excessive rate of change in the output so you can match the slew rate of the control element to the control dynamics. We recommend that you configure this value before you tune the loop, so tuning can accommodate any slow down in response time caused by this rate limiting. This check is not applied for a function block that is in the MANual mode. The default value is Not a Number (NaN), which means no rate limiting is applied. • Minimum Change (%) (OPMINCHG) – Lets you specify an output minimum change limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). This lets you define how much the OP must change before the function block outputs a new value. It filters out changes that are too small for the final control element to respond to. This check is not applied for a function block that is in the MANual mode. The default value is 0, which means no change limiting is applied. • Safe OP (%) (SAFEOP) – Lets you specify the safe output value as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 0 to 500 and you enter a Safe OP of 50%, the safe output value in engineering units is 50% times 500 or 250. The default value is Not a Number (NaN), which means the OP is held at its last good value. ‘ • Output Bias (OPBIAS.FIX) – Lets you specify a fixed bias value in engineering units that is added to the Calculated Variable (CV) output value. See the Output Bias section for this function block for details. The default value is 0, which means no value is added.

Configuration Tab	Description
	• Output Bias Rate (OPBIAS.RATE) – Lets you specify an output floating bias ramp rate in engineering units per minute. This bias rate is only applied when the floating bias in non-zero. See the Output Bias section for this function block for details. The default value is Not a Number (NaN), which means no floating bias is calculated. As a result, if the primary does not accept the block's initialization value, a bump in OP occurs. • Gain (K) – Lets you specify a gain (K) value to be factored into the equation for calculating the CV output value. This value helps guarantee that the output is "bumpless" following initialization or mode changes. The default value is 1. • Gain High Limit (GAINHILM) – Lets you specify gain high limit value. Gain (K) is clamped to this value, if the specified gain exceeds it. The default value is 240. • Gain Low Limit (GAINLOLM) – Lets you specify gain low limit value. Gain (K) is clamped to this value, if the specified gain is less than it. The default value is 0. • CV (CVSRC) – Lets you assign an input or expression result as the source for the CV. The default selection is NONE. • CV Initialization (CVINITSRC) – Lets you assign an input or expression result as the source of the CV initialization. The default selection is NONE. • CV Override (CVORFBSRC) – Lets you assign an input or expression result as the source of the CV during override. The default selection is NONE. • Initialization Request (INITREQSRC) – Lets you assign an input or expression result as the source of the initialization request flag for the primary. If desired, you can leave this parameter unassigned. The default selection is NONE. The ENHREGCALC block uses different values for this parameter depending upon whether or not a source is assigned and a secondary exists as follows. – If a source is assigned, this block uses the assigned source value even if a secondary does exist.

Configuration Tab	Description
	– If no source is assigned and a secondary does exist, this block uses the corresponding value from the secondary. – If no source is assigned and there is no secondary, this block uses default values (NaN for values, OFF for flags). • Initialization Value (INITVALSRC) – Lets you assign an input or expression result as the source of the initialization value for the primary. If desired, you can leave this parameter unassigned. The default selection is NONE. The ENHREGCALC block uses different values for this parameter depending upon whether or not a source is assigned and a secondary exists as follows. – If a source is assigned, this block uses the assigned source value even if a secondary does exist. – If no source is assigned and a secondary does exist, this block uses the corresponding value from the secondary. – If no source is assigned and there is no secondary, this block uses default values (NaN for values, OFF for flags). • Override Feedback Status (ORFBSTSSRC) – Lets you assign an input or expression result as the source of the override feedback status for the primary. If desired, you can leave this parameter unassigned. The default selection is NONE. The ENHREGCALC block uses different values for this parameter depending upon whether or not a source is assigned and a secondary exists as follows. – If a source is assigned, this block uses the assigned source value even if a secondary does exist. – If no source is assigned and a secondary does exist, this block uses the corresponding value from the secondary. – If no source is assigned and there is no secondary, this block uses default values (NaN for values, OFF for flags).

Configuration Tab	Description
	• Override Feedback Value (ORFBVALSRC) – Lets you assign an input or expression result as the source of the override feedback value for the primary. If desired, you can leave this parameter unassigned. The default selection is NONE. The ENHREGCALC block uses different values for this parameter depending upon whether or not a source is assigned and a secondary exists as follows. – If a source is assigned, this block uses the assigned source value even if a secondary exists. – If no source is assigned and a secondary exists, this block uses the corresponding value from the secondary. – If no source is assigned and there is no secondary, this block uses default values (NaN for values, OFF for flags).
Alarms	• Type – Identifies the types of alarms this block supports. Of course, these alarms also interact with other block configuration options such as the Safety Interlock Option (SIOPT) and Bad Control Option (BADCTLOPT). The types are: – Safety Interlock (SIALM.FL) – Bad Control (BADCTLALM.FL) – OP High (OPHIALM.FL) – OP Low (OPLOALM.FL) • Enable Alarm (SIALM.OPT) – Lets you enable or disable Safety Interlock alarm type. A check in the box means the alarm is enabled. The default selection is a checked box or enabled (Yes). You can also configure the SIALM.OPT parameter as a block pin, a configuration and/or a monitoring parameter so it appears on the block in the Project and Monitoring tree view, respectively. • Trip Point – Lets you specify the OP High Alarm (OPHIALM.TP) and OP Low Alarm (OPLOALM.TP) trip points in percent. The default value is NaN, which disables the trip point.

Configuration Tab	Description
	• Priority – Lets you set the desired priority level individually for each alarm type (SIALM.PR, BADCTLALM.PR, OPHIALM.PR, OPLOALM.PR). The default value is LOW. The levels are: – NONE - Alarm is neither reported nor annunciated. – JOURNAL - Alarm is logged but it does not appear on the Alarm Summary display. – LOW, HIGH, URGENT - Alarm is annunciated and appears on the Alarm Summary display. • Severity – Lets you assign a relative severity individually for each alarm type (SIALM.SV, BADCTLALM.SV, OPHIALM.SV, OPLOALM.SV) as a number between 0 to 15, with 15 being the most severe. This determines the alarm processing order relative to other alarms. The default value is 0. • Deadband Value (ALMDB) – Lets you specify a deadband value that applies to all analog alarms to prevent nuisance alarms due to noise at values near the trip point. The default value is 1. Note that this value is loaded to the individual alarm parameters (for example, OPHIALM.DB and OPLOALM.DB) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB. • Filter Time (ALMTM) – Lets you specify a time in seconds to define how long an analog alarm must exist before it is set true. The default value is 0, which means the alarm is set true as soon as the value exceeds the deadband value. Note that this value is loaded to the individual alarm parameters (for example, OPHIALM.TM and OPLOALM.TM) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB. • Deadband Units (ALM.DBU) – Lets you specify if the deadband value represents percent or engineering units. The default value is percent.

Configuration Tab	Description
	Note that this value is loaded to the individual alarm parameters (for example, OPHIALM.DBU and OPLOALM.DBU) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB.
SCM	• SCM Mode Tracking Option (MODETRACK) – Lets you select the desired Mode Tracking function for the SCM associated with this block's Control Module. It defines how the FB will set the state of the MODEATTR based upon the MODE of the SCM. See the SCM Interface and CM Interaction section in this document for selection details. The default selection is ONESHOT. The selections are: – None – ONESHOT – SEMICONT – CONTRTN – CONT • Option Type – Lets you specify the action the function block is to take when the SCM goes into an abnormal state. The Starting State Option (STARTOPT) applies when the SCM state is Checking, Idle, or Complete. The Stop/Abort State Option (STOPOPT) applies when the SCM state is Stopping or Stopped, Aborting or Aborted. The Hold State Option (HOLDOPT) applies when the SCM state is Holding or Hold. The Restart State Option (RESTARTOPT) applies when the SCM state is Resume or Run. The NONE and LASTREQ are the only selections for the Restart State Option. You can select from one of these types for the other options as applicable for the given regulatory control function block: – NONE - No changes. – MAN - Set MODEREQ = MANUAL. – AUTO - Set MODEREQ = AUTOMATIC. – CAS - Set MODEREQ = CASCADE. – FIXEDOP - Set OPREQ = Configured Value.

Configuration Tab	Description
	– HOLDPV - Set SPREQ = PV (not applicable to this block). – FIXED SP - Set SPREQ = Configured Value and SPRATEREQ = NaN (not applicable to this block). – RAMPEDSP - Set SPTVREQ = Configured Value and SPRATEREQ = Configured. • Value (STARTVAL, STOPVAL, HOLDVAL) – Depending upon Option Type selection, lets you specify an output or set point value within the respective range. For output, within OPEXLOLM to OPEXHILM and within SPLOLM to SPHILM, for set point. The default value is NaN (Not a Number). • Rate (STARTRATE, STOPRATE, HOLDRATE) - When the RAMPEDSP option is selected, lets you specify a rate value (STARTRATE, STOPRATE, HOLDRATE) for setting the SPRATEREQ for an SP ramping function.
Expr# 1 to Expr# 8	• Expression (C[1..8]) – Lets you write up to eight desired expressions for custom calculations. See the Guidelines for writing expressions section for this block for more details.
Block Pins	Lets you select the available parameters that you want to expose as input/output pins on the function block graphic in Control Builder.
Configuration Parameters	Lets you select the available parameters that you want to appear on the face of the function block in the Project tab in Control Builder.
Monitoring Parameters	Lets you select the available parameters that you want to appear on the face of the function block in the Monitoring tab in Control Builder.
Block Preferences	Lets you change several block-viewing preferences including the color of the block's faceplate.
Insertion Type	Lets you include an insertion type from a CAB instances in the block. See *CAB insertion configuration considerations for regulatory control blocks* for more information

Function

- Each expression can contain any valid combination of inputs, operators and functions; and may perform arithmetic or logic operations.
- You can write expressions for calculating CV under normal, initialization and override feedback conditions. Or, you can write expressions which produce initialization and override feedback values for this block and its primaries.
- You can assign the result of an expression or an input to any assignable output, which produces the same outputs as every other regulatory control block. You can assign the same input to multiple outputs.

Configuration example

The following figure shows a sample configuration using an ENHREGCALC block in a ratio control loop.

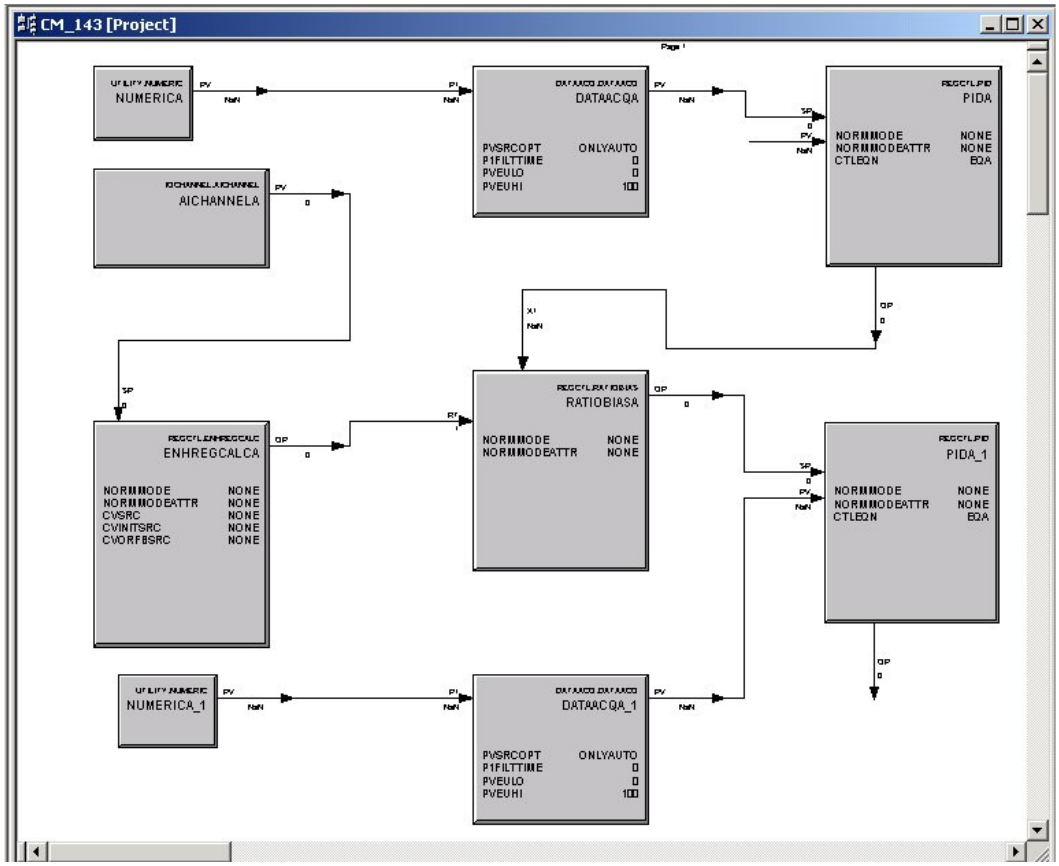


Figure 18 Example of CB configuration using ENHREGCALC block.

Operating modes and mode handling

The ENHREGCALC block supports the Manual, Automatic, and Cascade modes of operation.

If Mode is . . .	Then,
Manual (MAN)	the output can be set by the operator or a user program. The SP input is ignored.
Automatic (AUTO)	the block derives OP; the initializable input (SP) may be stored by the operator or a user program.
Cascade (CAS)	the SP input comes from another function block.

The initialization request occurs when the MODE changes from CAScade to MANual, but not from MANual to CAScade. When the block is put in MANual mode, it initializes itself and requests its primary to initialize.

Inputs

The ENHREGCALC block has the following inputs.

- SP- An initializable input. If Mode is CAScade, SP is pulled from another function block. If Mode is AUTO, it may be stored by the operator or a user program.
- X[1] through X[10] general purpose inputs.
- XB[1..10] individually configurable bias value for each X input.
- XDESC[1..10] individually configurable description for each X input.
- XENABLE[1..10] individually configurable enable/disable switch for each X input.
- XK[1..10] individually configurable gain value for each X input.
- XKB[1..10] individual inputs with gain and bias values applied to them.
- XSTS[1..10] individual status for each X input.
- XSUB[1..10] individulally configurable substitute value for each X/PX input, when corresponding X input is disabled.
- XWHIFL – An external windup high flag.
- XWLOFL – An external windup low flag.

Since SP is an initializable input, the block can have one primary. There is one primary for each initializable input.

Initializable input

Since SP is an initializable input, the block can have one primary. There is one primary for each initializable input. The ENHREGCALC block performs the following processing functions.

- SP limit checking
- SP Target Value Processing

This function provides a smooth transition (ramp) from the current SP to a target SP. It is invoked through the configuration option, SPTVOPT.

SP limit checking

SP Limit Processing makes sure the SP value does not exceed configured limits. These limits are configured with the following parameters:

- SPHILM SP high limit
- SPLOLM SP low limit

If the input SP value is outside the range specified by SPHILM and SPLOLM, the function block clamps SP to the appropriate limit and sets the appropriate **limit exceeded** flag (SPHIFL or SPLOFL).

SPHILM and SPLOLM are configured in the same engineering units, as SP. Crossover of set point limits is not allowed. SPHILM and SPLOLM can be changed even when the Control Execution Environmnet (CEE) is in RUN and the Control Module (CM) is Active.

When the SP limits are violated or the SP returns to normal, the network Anti-Reset Windup status (ARWNET) is recomputed and the primary's windup status is set.

SP target value processing

The SP target value processing provides a smooth transition from the current set point value to a new value. It ramps the SP from the existing value to a desired value over a period of time. It is similar to that of the PID block except that the ENHREGCALC block does not have Max. Ramp Deviation due to absence of PV parameter.

The SP target value processing uses the following parameters:

SPTVOPT	SP Target Value Option (indicates if SP target value processing is requested)
SPTV	target SP value (user-specified)
SPTVNORMRATE	normal ramp rate (user-specified)
SPTVRATE	calculated ramp rate
SPTVTIME	ramp time (calculated or user-specified)
SPTVSTATE	current target value processing state (possible states are Off, Preset, Run) SP target value processing works as follows:

SP target value processing works as follows:

- At configuration time, you should:
  - Enable SP target value processing by setting SPTVOPT to Enable. If SPTVOPT setting is Disable, you cannot store to any other SPTV parameter.
  - Specify a normal SP ramp rate (SPTVNORMRATE). This parameter lets you start SP ramping without having to specify a ramp time. The SPTVNORMRATE may be NaN or greater than zero. The NaN means **step change** or ramp time equals 0).
- To invoke SP ramping, you must do the following:
  - Store a target SP value (SPTV). To store to SPTV, target value processing must be enabled but not running, so SPTVOPT equals Enable, and SPTVSTATE equals Off or Preset. When SPTV is stored, and the function block is active, the following will occur:

    The system calculates a ramp time (SPTVTIME), based on the normal ramp rate

    SPTVSTATE goes to Preset.

  - Set SPTVSTATE equals Run. To store to SPTVSTATE, the function block must be active and to start ramping, MODE must be Auto. When SPTVSTATE goes to Run, SP begins to ramp toward SPTV, and SPTVTIME decreases over time. When SPTVTIME reaches zero, SP will equal SPTV, and SPTVSTATE goes to Off. You may set the ramp time by storing to SPTVNORMRATE or SPTVTIME. The ramp rate is derived from these two parameters and cannot be changed by the user.

- You may start ramping without specifying a ramp time. If you do, the function block calculates a ramp time and rate (SPTVTIME and SPTVRATE) as follows:

  If SPTVNORMRATE is not NaN:

  SPTVRATE = SPTVNORMRATE

  SPTVTIME = |(SPTV - SP)| / SPTVRATE

  Otherwise,

  SPTVRATE = NaN

  SPTVTIME = 0, or do a step change

- If you specify a ramp time (SPTVTIME), the function block calculates a ramp rate (SPTVRATE) as follows:

  If SPTVTIME is non-zero:

  SPTVRATE = |(SPTV - SP)| / SPTVTIME

  Otherwise,

  SPTVRATE = NaN

- If the user changes the normal ramp rate (SPTVNORMRATE), the function block recalculates the ramp time and rate (SPTVTIME and SPTVRATE) as follows:

  If SPTVTIME is non-zero:

  SPTVRATE = SPTVNORMRATE

  SPTVTIME = |(SPTV - SP)| / SPTVRATE

  Otherwise,

  SPTVRATE = NaN

  SPTVTIME = 0 or do a step change

Notes on SP target value processing

- To start ramping to set SPTVSTATE equals Run:
  - MODE must be AUTO
  - SPTVSTATE must be Preset
- SPTVSTATE automatically goes to Preset when:
  - SPTV or SPTVTIME are stored
  - MODE changes to Manual while SPTVSTATE is in Run
  - The function block goes to INIT while SPTVSTATE is in Run

TIP

A oneshot initialization does not cause a change in SPTVSTATE.

- SPTVSTATE automatically goes to Off when:
  - SP is stored by the operator, a user program or another function block
  - MODE changes to Cascade or Backup Cascade
  - The function block goes Inactive
- SPTV and SPTVTIME can only be stored if:
  - SPTVSTATE is Off or Preset, and
  - MODE is Auto or Manual
- The SP limits (SPHILM and SPLOLM) cannot be changed unless SPTVSTATE is Off. This is because SPTV is limit-checked (against the SP limits), and changing the limits could affect SP ramping.
- If the anti-reset windup status (ARWNET) indicates that SP is woundup (Hi, Lo or HiLo), then SP ramping will stop. When ARWNET indicates that SP has returned to normal, SP ramping will continue from where it stopped.

TIP

When SP is ramping, ARWNET is not shown on the Group or Detail displays.

Initializable outputs

"Initializable output" and "initializable input" are variable attributes, similar to data type or access level. A parameter with the "initializable" attribute has an associated BACKCALC variable, and when a connection is created between an initializable input and initializable output, you can also create a BACKCALC connection. Control Builder automatically builds the required BACKCALC connections, so you don't have to create them manually. These "implicit" build connections are "hidden" from view and the related parameter pins are not exposed on the control chart.

For example, if you connect OP from a ENHREGCALC block to SP on a PID block, Control Builder automatically creates the BACKCALCOUT to BACKCALCIN connection.

- The ENHREGCALC block has the following initializable outputs:
  - OP = calculated output in percent.
  - OPEU = calculated output in engineering units.

You may create a connection to OP or OPEU but not both. Therefore, this block may have only one secondary. If you do not create a connection to OP or OPEU, then the block does not have a secondary. Alternately, if you connect OP or OPEU to a non-initializable input, then this block does not have a secondary. (Note that the default OP connection pin is exposed on the blocks and the implicit/hidden connection function automatically makes the appropriate value/status parameter (OPX/OPEUX) connection when required.

For example, if you connect the output from a ENHREGCALC block (ENHREGCALC.OP) to the set point of a PID block (PIDA.SP), the implicit/hidden connection is made to ENHREGCALC.OPX to provide value/status data.)

ATTENTION

Be sure you use a FANOUT block to make multiple output connections. We recommend that you do not make multiple connections from a single ENHREGCALC output.

Output ranges and limits

CVEUHI and CVEULO define the full range of CV in engineering units.

If this block has a secondary, it gets the secondary's input range through BACKCALC and sets its CV range to that. If it has no secondary, you must specify the values for CVEUHI and CVEULO.

ATTENTION

This block gets the secondary's input range regardless of SECINITOPT. This means regardless of whether the secondary's initialization and override data will be used.

OPHILM and OPLOLM define the normal high and low limits for OP, as a percent of the CV range. You must specify these values.

OP will be clamped to these limits if the algorithm's calculated result (CV) exceeds them, or another function block or user program attempts to store an OP value that exceeds them. However, the operator may store an OP value that is outside these limits.

OPEXHILM and OPEXLOLM define the extended high and low limits for OP, as a percent of the CV range. You must specify these values.

The operator is prevented from storing an OP value that exceeds these limits.

Assignable outputs

You can assign expression results and/or inputs to the following parameters.

- CVSRC – CV output source selector.
- CVINITSRC – CVINIT source selector.
- CVORFBSRC – CVORFB source selector.
- INITREQSRC – INITREQ (initialization request flag) source selector.
- INITVALSRC – INITVAL (initialization value) source selector.
- ORFBVALSRC – ORFBVAL (override feedback value) source selector.
- ORFBSTSSRC – ORFBSTS (override feedback status) source selector.

For example, you can assign the result of the second expression to CVSRC and the result of the fourth expression to CVINITSRC and CVORFBSRC. You may assign the same input to multiple outputs. You may also assign inputs directly to outputs, such as assigning X[1] and X[2] to INITVALSRC and INITREQSRC, respectively.

The assignable expression and input parameters are as follows:

C[1..8] – Expressions

CSTS[1..8] – Expression Status

X[1..10] – Inputs

XSTS[1..10] – Input Status

Output assignment rules

ATTENTION

The ENHREGCALC block does perform data conversions, if the source and target parameters are of different types. For example, if you assign the INITREQSRC to X[2], the block converts the real type data from X[2] into Boolean type data for INITREQ[1] that it sends to its primary. You must be careful when making assignments that the resulting data conversions do not make sense. For example, if you assign XSTS[1] to ORFBSTSSRC, the two statuses are entirely different and they cause the block to produce unexpected results.

- The following parameters should be assigned to an input or an expression result
  - CVSRC – Since this parameter controls CV under normal conditions, when the block is not initializing and its mode is CAScade, always assign this parameter. If this parameter is left blank or unassigned, the Control Module containing the block is allowed to go **Active**, but CV is NaN and OP has a value of zero.
  - CVINITSRC – Since this parameter controls CV when the block is in its initialization state, CV will get initialized with the initialization value from the secondary, like the other regulatory control blocks, if this parameter is not assigned. You should only need to assign CVINITSRC when CV needs to be initialized with a customized value.
    If the CV value based on CVINITSRC assignment computes to NaN, it will be replaced by the INITVAL received from the secondary
    If the CV value based on CVINITSRC assignment is used as the INITVAL for the primary and you have assigned INITVALSRC to compute a customized INITVAL, the INITVAL for the primary will be based on INITVALSRC.
  - CVORFBSRC – Since this parameter controls CV when the block's override status is "unselected", you should only need to assign CVORFBSRC when CV needs to be set based on the block's override status. The PID block is the only one that sets its CV to override the feedback value received from its secondary when the block's override status is "unselected". For other regulatory control blocks, CV is not affected by the block's override status.

- The following parameters are provided to the primary through this block's BACKCALC data.
  - INITREQSRC
  - INITVALSRC
  - ORFBVALSRC
  - ORFBSTSSRC
  - You can assign these parameters to an input or an expression result, or leave them unassigned. The following table summarizes possible outcomes for specified parameter assignments. You may need to assign an INITVALSRC to compute a customized initialization value for the primary based on the CVSRC assignment.

If a Parameter is . . .	And, a Secondary. . .	Then, This Block. . .
assigned	does or does not exist	uses the assigned value.
unassigned	exists	uses the corresponding value from the secondary.
unassigned	does not exist	uses default values, such as NaN for values and Off for flags.

Control initialization

The ENHREGCALC block brings initialization requests from its secondary through BACKCALC. In addition, the secondary may propagate oneshot initialization requests to this block. (Note that SECINITOPT may be used to ignore initialization requests from the secondary.)

If the secondary is requesting initialization, the ENHREGCALC block:

- initializes its output:

  CV =CVINIT (an assignable output)

- builds an initialization request for the designated primaries using the assignable output parameters INITREQSRC and INITVALSRC. If you configure no assignments for these parameters, the block behaves like other regulatory control blocks, using the corresponding values brought from its secondary.

Be careful when making INITREQSRC and INITVALSRC assignments to avoid producing the wrong results. For example, you assign the INITREQSRC parameter to C[2], which produces a result of TRUE, and the ENHREGCALC block's mode is CAScade and its INITMAN parameter is OFF. Also, you have assigned CVSRC to C[1], which is configured as "X[1] +10.0", and INITVALSRC to C[3], which is configured as this block's CV. Assume at some moment that X[1] is 15.0 and it produces a C[1] of 25.0, resulting in CV = INITVAL[1] = 25.0. The primary will initialize itself with the value 25.0. This means that the next time the ENHREGCALC block runs it receives an X[1] value of 25.0 from the primary, resulting in C[1] = CV = 35.0. Thus, each cycle that ENHREGCALC runs, its CV increments by 10.0, producing seemingly wrong results.

Output bias

The output bias (OPBIAS) is added to the algorithm's Calculated Value (CV) and the result is stored in CV. CV is later checked against OP limits and, if no limits are exceeded, copied to the output.

The OPBIAS is the sum of the user-specified fixed bias (OPBIAS.FIX) and a calculated floating bias (OPBIAS.FLOAT). The purpose of the floating bias is to provide a bumpless transfer when the function block initializes or changes mode.

- OPBIAS is recomputed under the following conditions to avoid a bump in the output. (Note that the ENHREGCALC block only applies OPBIAS.FLOAT to the output for the latter two conditions, when it is the first initializable block.)
  - When the function block starts up (that is, goes Active).
  - When the function block initializes (for example, the secondary requests initialization).
  - When the mode changes to Auto or Cascade.

ATTENTION

When the mode is Manual, OPBIAS is not used, since the output is not calculated. This means that the OPBIAS is not recomputed, when the mode changes to Manual.

- You may set the OPBIAS value only if the function block is Inactive or mode equals Manual. This is done to prevent a bump in the output when the bias is changed. The following occurs when you set the OPBIAS value.

- The total bias (OPBIAS) and fixed bias (OPBIAS.FIX) are both set to the entered value.
- The floating bias (OPBIAS.FLOAT) is set to zero.

ATTENTION

When the function block goes Active or the Mode changes to Auto or Cascade, OPBIAS and OPBIAS.FLOAT are recomputed.

- There are no limit checks applied when you set an OPBIAS value. However, after the bias is added to CV, the result is compared against the output limits and clamped, if necessary.
- You configure the value for the fixed bias (OPBIAS.FIX) and it is never overwritten by the floating bias (OPBIAS.FLOAT). This means the total bias will eventually equal the OPBIAS.FIX , if you configure OPBIAS.RATE to ramp down OPBIAS.FLOAT.
- The OPBIAS.FLOAT is calculated as follows.

$$OPBIAS.FLOAT = CV_{INIT} - (CV_{UNBIASED} + OPBIAS.FIX)$$

Where:

CV_{INIT} = initialization value received from the secondary

$CV_{UNBIASED}$ = unbiased calculated value (based on input from the primary)

OPBIAS.FIX = fixed bias (user-specified)

- If the primary accepts this block's initialization request, then CV + OPBIAS.FIX should be the same as CVININT and OPBIAS.FLOAT will be zero. In most cases, OPBIAS.FLOAT will be zero. However, if the primary does not accept this block's initialization request because the primary is a FANOUT block or it was configured to ignore initialization, then OPBIAS.FLOAT value will not be zero.

  If OPBIAS.FLOAT is not zero, you can configure it to ramp down to zero through the OPBIAS.RATE parameter.

- You configure the OPBIAS.RATE to apply a ramprate to the OPBIAS.FLOAT. It is only used when the OPBIAS.FLOAT is not zero. The OPBIAS.RATE is expressed in Engineering Units per minute and may have the following values.

  – Zero:
    If OPBIAS.RATE is zero, an OPBIAS.FLOAT is calculated and bumpless transfer is guaranteed. However, if OPBIAS.FLOAT is not zero, it will never ramp down.

  – Non-zero:
    If OPBIAS.RATE is not zero, an OPBIAS.FLOAT is calculated and bumpless transfer is guaranteed. If the OPBIAS.FLOAT is not zero, it is ramped to zero at the rate you configured for the OPBIAS.RATE parameter.

    The function block ramps the OPBIAS.FLOAT to zero by applying the following calculation each time it executes.

    OPBIAS.FLOAT = OPBIAS.FLOAT – (OPBIAS.RATE / cycles_per_Min)

  Where:

    cycles_per_min = number of times the function block executes per minute (calculated)

  – NaN:
    When the OPBIAS.RATE is Not a Number (NaN), no OPBIAS.FLOAT is calculated. This means a bump in the output will occur, if the primary does not accept this block's initialization value.

- After initialization, the ENHREGCALC block calculates the floating bias using the following equation.

$$OPBIAS.FLOAT = CVINIT - (Cv_{unbiased} + OPBIAS.FIX)$$

Where:

$CV_{unbiased}$ = unbiased CV (It equals K * X[1], if X[1] is assigned to CV.)

OPBIAS.FIX = fixed output bias (user specified)

Timeout monitoring

If mode is CAScade, the ENHREGCALC block performs timeout monitoring of the initializable input, SP If the SP value is not updated within a predefined time (TMOUTTIME), the block invokes timeout processing as noted in the following section.

The maximum time between updates is specified by TMOUTTIME (in seconds)

- Enable timeout monitoring by setting TMOUTTIME to a non-zero value.
- Disable timeout monitoring by setting TMOUTTIME to zero.

Timeout processing

If SP times out, the ENHREGCALC block does the following:

- Sets the input timeout flag (TMOUTFL)
- Holds SP at its last good value.
- Sheds to a user-specified timeout mode (MODE = TMOUTMODE).
- Requests the SP primary to initialize (via BACKCALCOUT)

ATTENTION

If the input is from a connection in another controller in a peer-to-peer architecture, the actual timeout time equals the configured TMOUTTIME plus the CDA timeout time. The CDA timeout time equals four times the configured CEE subscription rate. For example, if the CEE subscription rate is 100 milliseconds and the TMOUTTIME is 5 seconds, the actual timeout time for the block is 4 times 100ms plus 5s or 5.4 seconds.

Mode shedding on timeout

- The ENHREGCALC block sets its cascade request flag (CASREQFL), if SP times out and sheds to AUTOmatic mode. This indicates that the block is waiting to return to the CAScade mode, and it will as soon as it fetches a good SP value. When it receives a good SP value, the block does the following:
  - Changes the mode back to CAScade.
  - Updates the SP.
- You **cannot** set the CASREQFL. However, it can be cleared by setting the block's MODE to MANUAL.
- If you want to disable the automatic return to CAScade, you may do one of the following:
  - Store to MODE, which causes the CASREQFL to be cleared.
  - Store to CASREQFL.
- If a block sheds to MANual mode, it will **not** set CASREQFL. Consequently, it will not return to CAScade when it fetches a good value from the primary.

Override feedback processing

If the ENHREGCALC block is in a cascade strategy with a downstream OVRDSEL (Override Selector) block, it receives override feedback data. The data consists of an override status, override feedback value and an override offset flag. The status indicates if this block is in the selected or unselected strategy (as determined by the OVRDSEL block). The offset flag only applies to PID-type blocks.

When the override status changes from selected to unselected, the ENHREGCALC block does the following:

- initializes its output:

  CV = CVORFB (an assignable output)

- Computes a feedback value for its primary:

  feedback value for primary = ORFBVAL (an assignable output)

  feedback status for primary = ORFBSTS (an assignable output)

If the ORFBVAL and ORFBSTS are not assigned and this block has a secondary, the ORFBVAL and ORFBSTS received from the secondary are used to compute ORFBVAL for the primary. When the override status from the secondary changes from selected to unselected, this block does the following:

feedback value for primary = feedback value received from secondary.

ATTENTION

You can use SECINITOPT to ignore override requests from the secondary.

You can customize the override feedback computation and propagation using the following block parameters.

ORFBSTSSRC – If you make an ORFBSTSSRC parameter assignment, the ENHREGCALC block computes the override feedback status from the assignment and uses it for override processing and propagation to the primary. If you do not make an assignment, the ENHREGCALC block uses the override status received from the secondary for override processing, just like other regulatory control blocks do.

ORFBVALSRC – Like ORFBSTSSRC, if you make an ORFBVALSRC parameter assignment, the ENHREGCALC block computes the override feedback value for the primary based on the assignment. Otherwise, the block uses the override status received from the secondary for override processing , just like other regulatory blocks do.

CVORFBSRC – If you make a CVORFBSRC parameter assignment, the ENHREGCALC block computes the CV override feedback value based on the assignment and it sets its CV equal to the CVORFB, when the override status for the block is “unselected”. The override status could be based on the default status received from the secondary, when the ORFBSTSSRC parameter is unassigned, or a computed customized status based on the CVFBSTSSRC parameter assignment.

You can write incremental (like PID block) or non-incremental (like AUTOMAN block) expressions for CV, but certain configuration combinations may cause misleading block behavior – especially, when the expression for CV is non-incremental. For example, if you assign CVSRC to X[1] and CVORFBSRC to C[1] with C[1] configured as X[2], assume that at some moment X[1] is 10.0 and X[2] is 50.0, and the override status for the block is “unselected”. This configuration produces different values for the block’s CV and OP parameters. Based on X[1], the first CV value is computed as 10.0 and the resulting OP value is 10.0. But, based on X[2], the CVORFB value is computed as 50.0 and the block overwrites the previous CV value of 10.0 with 50.0, resulting in different CV and OP values. In this case, assigning CVSRC to X[1] was the wrong configuration

to use. You can eliminate this type of discrepancy by assigning the CVSRC to an expression that calculates a CV incrementally, such as CV + Delta (CV) so that Delta (CV) is the incremental value added to its previous value of CV.

Windup handling

The ENHREGCALC block derives the ARWOP from a combination of the following parameters and the secondary's windup status.

- CV
- XWHIFL
- XWLOFL

The following table summarizes how the block derives ARWOP for some given conditions.

If XWLOFL and/or XWHIFL are. . .	And a Secondary. . .	Then, the Block Derives ARWOP from . . .
True	does or does not exist	CV, XWHIFL, and XWLOFL.
False	exists	CV and secondary's windup status.
False	does not exist	CV only.

When the ENHREGCALC block computes its ARWOP windup status for its primary (ARWNET[1]), which is computed based on ARWOP, it will be propagated to the primary just like other regulatory control blocks.

ATTENTION

The ARWNET[1] computation is independent of whether gain (K) is positive or negative.

Windup processing

Every regulatory control type block maintains anti-reset windup status for its output (ARWOP) and each of its initializable inputs (ARWNET). The following table lists the posssible values for ARWOP and ARWNET parameters.

If the Value is . . .	Then, the Associated Parameter . . .
Normal	is free to move in either direction.
Hi	is at its high limit and it may only be lowered.
Lo	is at its low limit and it may only be raised.
HiLo	may not move in either direction.

Manual Mode Interaction

When the MODE of a regulatory control block is changed to Manual (Man), the block sets its windup status (ARWNET) to HiLo. This means that every block upstream in a cascade strategy will set its windup status (ARWNET and ARWOP) to HiLo.

ARWOP computation

The ARWOP indicates if the output (OP) can be raised or lowered. The PID-type function blocks use ARWOP to restrict integral control. When ARWOP contains a value other than **Normal**, the PID block stops integral control in the windup direction. Integral control continues in the other direction, as does proportional and derivative control. But, windup status has no impact on proportional and derivative control.

If a function block has a secondary, it fetches the secondary's windup status and recomputes its ARWOP. The conditions within the function block, such as output being at its high limit, also affect the ARWOP. The ARWOP is computed as follows, assuming the block has only one output or that it is not a FANOUT block.

If Any of the Following are True . . .	Then, ARWOP Equals . . .
This block is inactive.	**HiLo**
A secondary exists but this block cannot fetch secondary data from it (communications or configuration error).	

If Any of the Following are True . . .	Then, ARWOP Equals . . .
A secondary exists and its windup state equals HiLo	
This block is in initialization (INITMAN = On).	
A secondary exists and it is requesting this block to initialize.	
A secondary exists and its windup state equals Hi.	**Hi**
This block's output is at its high limit (OPHIFL = On).	
A secondary exists and its windup state equals Lo.	**Lo**
This block's output is at its low limit (OPLOFL = On).	

ARWNET computation

When ARWNET is HiLo, stores to SP are not limited, rather this is the status propagated to the primary. The only *limiting* anti-reset windup status ever does is to stop integral action in one or both directions on PID blocks. For any other regulatory control type block, ARWNET is not used for any kind of limiting. The ARWNET is computed as follows, assuming the block has only one output or that it is not a FANOUT block.

If Any of the Following are True . . .	Then, ARWNET Equals . . .
This block is inactive.	**HiLo**
The ARWOP equals HiLo.	
This block is in Manual mode (MODE = Man)	
The calculated value (CV) range (CVEUHI / CVEULO) is NaN.	
The CV is NaN	
This block is connected to a non-initializable primary	

If Any of the Following are True . . .	Then, ARWNET Equals . . .
The ARWOP equals Hi (Pid function blocks have a configurable Control Action option (CTLACTN). If CTLACTN = Reverse , ARWNET will track ARWOP; but if CTLACTN = Direct , ARWNET will be the opposite of ARWOP.)	**Hi**
The input from the primary is at a high limit. For example, SPHIFL = On.	
This block's output has reached its positive rate-of-change limit (OPROCPOSFL = On)	
The ARWOP equals Lo (Pid function blocks have a configurable Control Action option (CTLACTN). If CTLACTN = Reverse , ARWNET will track ARWOP; but if CTLACTN = Direct , ARWNET will be the opposite of ARWOP.)	**LO**
Input from the primary is at a low limit. For example, SPLO.FL = On.	
This block's output has reached its negative rate-of-change limit (OPROCNEGFL = On)	

Anti-Reset Windup Status

The anti-reset windup network in (ARWNETIN) and anti-reset windup output in (ARWOPIN) parameters are added in the standard anti-reset windup (ARW) computation logic. They are user configurable and allow stores from Sequential Control Modules (SCMs) and control algorithm block (CAB) programs.

The ARWNETIN and ARWOPIN parameters would be *ORed* into the existing standard logic so it is not lost. The following table summarizes the influence the ARWNETIN and ARWOPIN parameters have on the ARWNET and ARWOP parameters, which are not user configurable.

ARWNETIN or ARWOPIN Parameter Is. . .	Standard Computation Logic Is . . .	ARWNET or ARWOP Parameter Is . . .
NORMAL	NORMAL	NORMAL
NORMAL	HI	HI
NORMAL	LO	LO
NORMAL	HILO	HILO
HI	NORMAL	HI
HI	HI	HI
HI	LO	HILO
HI	HILO	HILO
LO	NORMAL	LO
LO	HI	HILO
LO	LO	LO
LO	HILO	HILO
HILO	NORMAL	HILO
HILO	HI	HILO
HILO	LO	HILO
HILO	HILO	HILO

Expressions

You can write up to eight expressions, each expression can contain any valid combination of inputs, operators, and functions. Table 1 lists the expression operators and functions supported by this block for reference as well as some case sensitive strings that can be used for special value constants in expressions.

Table 1 Expression operators, functions, and strings reference

Operators	Description		
Unary	+ –		
Binary Arithmetic	+ – * / MOD (x MOD y) ^ (x^y)		
Logical	AND OR NOT		
Relational	= <> <= >= < >		
Conditional	? : (For example, X ?Y : Z; similar to IF, THEN, ELSE)		
Parenthesis	()		
Array Syntax	[ ]		
Unary Functions			
ABS	absolute value	LOG	Base 10 logarithm of a number
ATN	arc tangent	RND	round value
COS[1]	cosine	SGN	sign of value (returns -1,0 or +1)
EXP	e to the power of x	SIN[1]	sine
INT	convert to integer	SQR	square of a number
ISFIN	is finite	SQRT	square root
ISNAN	is Not a Number	TAN[1]	tangent
LN	Natural logarithm of a number (log to the base of e)		
Multiple Argument Functions			

Operators		Description	
MIN	minimum of n arguments (ignore bad values)	MID	medium value of n arguments (average of middle values for even n)
MAX	maximum of n arguments (ignore bad values)	MUL	product of n arguments
AVG	average of n arguments	SUM	sum of n arguments
String Support Functions			
LEN	Returns an integer length of the string	NUMSTR	Takes the input parameter, casts it to a Float64 and converts it to a string
MID	Takes a string, an integer starting position and an integer length. The function returns the specified portion of the original string.	STRNUM	Takes the string input parameter and converts it to a Float64
Time Support Functions			
ABSTOD	Takes an absolute time data type and strips off the year and date and returns a 64-bit float representing the time of day in milliseconds.	DTIMNUM	Takes a delta TIME data type and returns a 64-bit float representing the number of milliseconds.
NOW	Returns the current local date and time as an absolute time data type	NUMDTIM	Takes a 64-bit float representing some number of milliseconds and converts it to a delta TIME data type.
NUMTIM	Takes a 64-bit float representing the number of milliseconds since Jan 1, 1972 and converts it to absolute TIME data type.	STRTIM	Takes a string input parameter and converts it to an Absolute time. The string must be in the same format as an Absolute time constant.

Operators		Description	
TOD	Returns the current local time of day as Time of Day data type	TIMNUM	Takes an Absolute TIME data type and returns a 64-bit float representing the total number of milliseconds since Jan 1, 1972.
UTCTOD	Returns the current UTC time of day as Time of Day data type	UTCNOW	Returns the current UTC date and time of day as an absolute time data type

[1]Be sure you specify the trigonometric functions cosine, sine, and tangent in radians and **not** degrees.

Case Sensitive Strings for Special Value Constants	
NAN	IEEE NaN value
+INF	IEEE + Infinity value
-INF	IEEE – Infinity value
PI	PI (3.14159. . .)
E	e (2.718. . .)

Parameters in Expressions

You must specify a parameter by its full tag name (for example, “CM25.PumpASelect.PVFL”, or “CM57.PID100.MODE”). In effect, tag names allow expressions to have an unlimited number of inputs, and work with any data type.

The expression syntax has been expanded. Delimiters (‘) can be used in an expression containing an external reference component. The format for the delimiter usage is as follows:

- TagName.’text’

TagName is the name of the external reference component (i.e. an OPC Server). Text can contain any characters, space, and special characters except for the delimiter character.

When entering this format, only the syntax and TagName are checked for accuracy. The correct syntax of TagName-dot-delimiter-text-delimiter is verified and the TagName is verified to be an external reference component. If either of these stipulations is incorrect, an error is issued. The text between the delimiters is not checked. It is the users responsibility to ensure that the text is something that the external reference component will understand. If this text is incorrect, runtime errors will occur.

ATTENTION

When the expression is sent to the external reference component, the delimiters are removed: TagName.’text’ becomes TagName.text.

Guidelines for Writing Expressions

- Must include full tag.parameter name for X inputs in the expression and enclose identification number in brackets instead of parentheses. For example, CM151.REGCALC_1.X[1] * CM151.REGCALC_2.X[2] is valid.
- Expressions cannot contain an assignment operation (a colon followed by an equal sign with the current syntax) For example, “PID1.MODE:=X[1]” is invalid.

  Each expression produces a single value (arithmetic or logical which is automatically stored in a “C” parameter. For example, if you write four expressions, the result of the first expression is stored in C[1], the result of the second is stored in C[2], etc. You can use these results, by name, in succeeding expressions. In this example, you could use C[1] as an input to expressions 2, 3, and 4.
- You can mix and nest all operators and functions (including conditional assignments) in any order as long as types match or can be converted.

- You can use blanks between operators and parameter names, but they are not required.
- You can use all data types in expressions, including enumerations. They are all treated as numeric types.

TIP

You can use the integer parameters YEAR, MONTH, DAY HOUR, MINUTE, and SECOND that provide local date and time for the controller in all expressions, just like other integer parameters.

- You must configure calculator expressions contiguously (without breaks) in the arrays. For example, a sample expression for calculating the average between minimum and maximum values would be as follows:
  - AVG (MIN(CM1.REGCALC.X[1], CM1.REGCALC.X[2], CM1.REGCALC.X[3]), MAX(CM1.REGCALCX[1], CM1.REGCALC.X[2], CM1.REGCALC.X[3]))

Enable/Disable switch example expression

The Enable/Disable switch is used as a flag in the following user expressions.

Example 1

MIN((CM.ENHREGCALCA.XENABLE[10] = 1) ? CM.ENHREGCALCA.X [10]:<user-entered default>, value2, value3).

Example 2

CM.ENHREGCALCA.X[1]<CM.ENHREGCALCA.X[2] and say X[1] is disabled then the CEE will evaluate the expression as CM.ENHREGCALCA.XSUB[1]<CM.ENHREGCALCA.X[2].

In this case, if an input is disabled, the corresponding substitute value is used in the expressions.

String data support in expressions

The following operators can have string constants and/or string references as operands.

Operator	Description
:=	Assignment - used only in the SCM Step Output blocks to assign the results of an expression to a reference. *Example:* *CM.block.mystringparam := "This is a string constant"* *CM.block.mystringparam := CM.desc*
+	Concatenation *Example:* *CM.block.mystringparam + CM.desc*
=	Equal to *Example:* *CM.block.mystringparam = CM.desc*
<>	Not equal to *Example:* *CM.block.mystringparam <> "This is a string"*

Time support in expressions

Time data types

The following time data types are supported in expressions.

- Absolute Time - Is stored as a 64-bit integer representing the number of tenths of milliseconds since 1/1/1972.
- Delta Time - Is also stored as a 64-bit integer and it represents an Absolute time difference in tenths of milliseconds.
- Time of Day - is an unsigned 32 –bit integer that represents a time of day in tenths of milliseconds.

Time constants

You can ues the following valid time constants in expressions.

- An Absolute Time constant is entered MM/DD/YYY hh:mm:ss:uuuu, where uuuu is milliseconds
- A Delta Time constant is entered as hh:mm:ss:uuuu, where uuuu is milliseconds
- Time of Day constant is also entered as hh:mm:ss:uuuu.

Time related operators

The following operators can have time constants and/or time references as operands:

Operator	Description
:=	Assignment - used only in the SCM Step Output blocks to assign the results of an expression to a reference. The data type in the expression result must agree with the data type of the reference.
+	If both operands are of the same time data type the result is the same data type. Delta time or Time of Day can be added to an absolute time, which results in absolute time. Time of day can be added to delta time, which results in a delta time. See the next section *Adding time data types*. .
*	One operand can be a delta time or time of day data type and the second operand must be a number. The result is a delta time data type.
-	Absolute Time can be subtracted from Absolute time, which results in a Delta Time. Delta time or Time of Day can be subtracted from an absolute time, which results in absolute time. Time of Day can be subtracted from Delta Time, which results in a Delta Time. See the following section Subtracting time data types.
=, <>, <=, >=, <, >	Compares two operands of type time. Both operands must be of the same time data type.
DAYS[1]	Takes operand and returns equivalent delta time value.
HOURS[1]	Takes operand and returns equivalent delta time value.
MINS[1]	Takes operand and returns equivalent delta time value.
SECS[1]	Takes operand and returns equivalent delta time value

[1]The DAY, HOURS, MINS, SECS Operators are not case specific.

Adding time data types

The following table shows results of adding the various time data types

Operand 2 Data Type	Operand 1 Data Type		
	Absolute Time	***Delta Time***	***Time of Day***
Absolute Time	Absolute Time	Absolute Time	Absolute Time
Delta Time	Absolute Time	Delta Time	Delta Time
Time of Day	Absolute Time	Delta Time	Time of Day

Subtracting time data types

The following table shows results of subtracting the various time data types

Subtrahend Data Type	Minuend 1 Data Type		
	Absolute Time	***Delta Time***	***Time of Day***
Absolute Time	Delta Time	N/A	N/A
Delta Time	Absolute Time	Delta Time	N/A
Time of Day	Absolute Time	Delta Time	Time of Day

Time expression examples

The following are examples of some valid time expressions.

- MYCM.block.elapsedtime > 5 MINS
- CEE01.CURRTIME + 2 DAYS
- CEE01.CURRTIME > 10/30/2002
- CEE01.CURRTIME + CM.TIMER.SP SECS
- (CEE01.CURRTIME – 1/01/2002 10:15:01) * 2.
- STRTIM("12/01/2002") > CEE01.CURRTIME

- TIMNUM(CEE01.CURRTIME)
- NUMTIM(1000.0)
- NOW – MyCM.myblock.todparam
- ABSTOD(CEE01.CURRTIME)

The following are examples of invalid expressions.

- CEE01.CURRTIME + 2
- CEE01.CURRTIME > 5.0

REFERENCE - INTERNAL

Please refer to *Time Support in Experion System* for more informaiton about time support in the system

Restart or Point Activation

On a warm restart or when an ENHREGCALC block is activated or inactivated, initialization takes place. Initialization also takes place when the ACE controller/node is repowered.

ENHREGCALC parameters

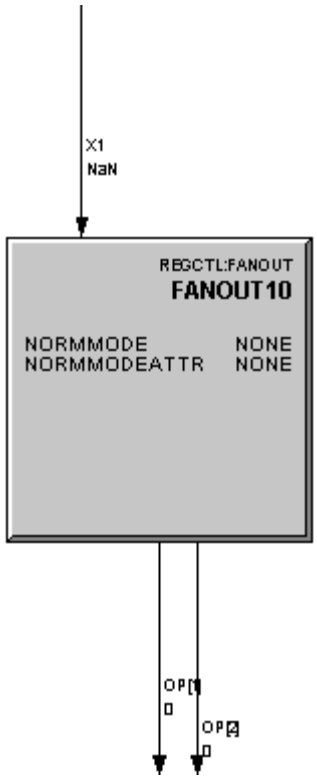
REFERENCE - INTERNAL

Refer to the *Control Builder Components Reference* for a complete list of the parameters used with the ENHREGCALC block.

FANOUT Block

Description

The FANOUT block has one input and up to eight initializable outputs. It may also have up to eight secondaries, since there is one secondary per initializable output. You may specify a separate gain, bias, and rate for each output. Each specified value can be fixed or external. A fixed value is stored manually or by a program, and an external value comes from another function block. This block calculates a separate floating bias for each output following an initialization or mode change. This provides a "bumpless" transition for each output. It looks like this graphically:



Each FANOUT block supports the following user configurable attributes. The following table lists the given name of the "Tab" in the parameter configuration form and then briefly describes the attributes associated with that Tab. This data is only provided as a quick document reference, since this same information is included in the on-line context sensitive Help.

Configuration Tab	Description
Main	• Name – Block (Tag) name of up to 16 characters long. Must be unique within the CM block containing it. • Description (DESC) – Block descriptor of up to 24 characters long. • Engineering Units (EUDESC) – Lets you specify a text string of up to 16 characters to identify the variable values associated with this block. For example, you could specify DEGF for temperature values in degrees Fahrenheit. This name is used on any associated displays and generated reports. • Execution Order in CM (ORDERINCM) – Specifies the execution order of the block in the CM relative to other blocks contained in this CM. Enter as a number between 1 to 32767. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. This is the block's parameter. • High Limit (XEUHI) – Lets you specify the high input range limit that represents 100% full-scale input for the block. The default value is 100. • Low Limit (XEULO) – Lets you specify the low input range limit that represents the 0 full-scale input for the block. The default value is 0 (zero). • Normal Mode (NORMMODE) – Lets you specify the MODE the block is to assume when the Control to Normal function is initiated through the Station display. Selections are MANual, AUTOmatic, CAScade, BackupCAScade and NONE. All selections are not valid for a given block. The default selection is NONE. • Normal Mode Attribute (NORMMODEATTR) – Lets you specify the mode attribute (MODEATTR) the block is to assume when the Control to Normal function is

Configuration Tab	Description
	initiated through the Station display. Selections are NONE, OPERATOR and PROGRAM. The default selection is NONE. • Mode (MODE) – Lets you set the block's current MODE. The selections are MANual, AUTOmatic, CAScade, BackupCAScade, NONE, and NORMAL. All selections are not valid for a given block. The default selection is MANual. MODE identifies who may store values to the block's initializable inputs or output. Blocks strictly enforce the MODE assignment. Some function blocks perform automatic mode switching (or mode shedding), while others require manual intervention. The block's MODE is derived at "runtime" based on current conditions. MODE processing checks for the following conditions, and changes the block's MODE as appropriate. – External request for MODE switching. – Safety interlock request. • Mode Attribute (MODEATTR) – Lets you set the block's mode attribute. The selections are NONE, OPERATOR, PROGRAM, and NORMAL. The default selection is OPERATOR. MODEATTR identifies who may store values to the output (OP), when the block's MODE is MANual. • Permit Operator Mode Changes (MODEPERM) – Lets you specify if operators are permitted to make MODE changes or not. The default is Enabled (checked). A store to MODE does not change the NORMMODE. • Permit External Mode Switching (ESWPERM) – Lets you specify if external MODE switching through user configured interlocks is permitted or not, if you have at least an Engineering access level. The default is Disabled (unchecked). • Enable External Mode Switching (ESWENB) – Lets you specify if external MODE switching through user configured interlocks is enabled or not, if ESWPERM is checked (Permitted). The default is Disabled (unchecked). • Safety Interlock Option (SIOPT) – Lets you specify

Configuration Tab	Description
	MODE and OP block is to assume upon a safety interlock alarm. The selections are NO_SHED, SHEDHOLD, SHEDLOW, SHEDHIGH, and SHEDSAFE. The default selection is SHEDHOLD. • Bad Control Option (BADCTLOPT) – Lets you specify MODE and OP block is to assume if CV goes BAD. The selections are NO_SHED, SHEDHOLD, SHEDLOW, SHEDHIGH, and SHEDSAFE. The default selection is NOSHED. • Timeout Time (TMOUTTIME) – Lets you specify a time in seconds that must expire before the block assumes that its input update has timed out. The block must be in CAScade mode for it to monitor its primary input for timeout. The default setting is 0, which means the timeout function is disabled. If the input is from a connection in another controller in a peer-to-peer architecture, the actual timeout time equals the configured TMOUTTIME plus the CDA timeout time. The CDA timeout time equals four times the configured CEE subscription rate. For example, if the CEE subscription rate is 100 milliseconds and the TMOUTTIME is 5 seconds, the actual timeout time for the block is 4 times 100ms plus 5s or 5.4 seconds.
Common Output	• High Limit (%) (OPHILM) – Lets you specify the output high limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you enter a High Limit of 90%, the high limit in engineering units is 90% times 450 or 405 + 50 (CVEULO) equals 455. This check is not applied for a function block that is in the MANual mode. The default value is 105%. • Low Limit (%) (OPLOLM) – Lets you specify the output low limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you enter a Low Limit of 10%, the low limit in engineering units is 10% times 450 or 45 + 50 (CVEULO) equals 95. This check is not applied for a function block that is in the MANual mode. The default value is -5%. • Extended High Limit (%) (OPEXHILM) – Lets you specify the output extended high limit as a percent of

Configuration Tab	Description
	the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you use the default value of 106.9%, the extended high limit in engineering units is 106.9% times 450 or 481.05 + 50 (CVEULO) equals 531.05. This check is not applied for a function block that is in the MANual mode. The default value is 106.9%. • Extended Low Limit (%) (OPEXLOLM) – Lets you specify the output extended low limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you use the default value of -6.9%, the extended low limit in engineering units is -6.9% times 450 or -31.05 + 50 (CVEULO) equals 18.95. This check is not applied for a function block that is in the MANual mode. The default value is -6.9%. • Rate of Change Limit (%) (OPROCLM) – Lets you specify a maximum output rate of change limit for both the positive and negative directions of the output in percent per minute. This lets you prevent an excessive rate of change in the output so you can match the slew rate of the control element to the control dynamics. We recommend that you configure this value before you tune the loop, so tuning can accommodate any slow down in response time caused by this rate limiting. This check is not applied for a function block that is in the MANual mode. The default value is Not a Number (NaN), which means no rate limiting is applied. • Minimum Change (%) (OPMINCHG) – Lets you specify an output minimum change limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). This lets you define how much the OP must change before the function block outputs a new value. It filters out changes that are too small for the final control element to respond to. This check is not applied for a function block that is in the MANual mode. The default value is 0, which means no change limiting is applied. • Safe OP (%) (SAFEOP) – Lets you specify the safe output value as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 0 to 500 and you enter a Safe OP of 50%, the safe output value in engineering units is 50% times 500

Configuration Tab	Description
	or 250. The default value is Not a Number (NaN), which means the OP is held at its last good value. • Gain High Limit (GAINHILM) – Lets you specify gain high limit value. Gain (K) is clamped to this value, if the specified gain exceeds it. The default value is 240. • Gain Low Limit (GAINLOLM) – Lets you specify gain low limit value. Gain (K) is clamped to this value, if the specified gain is less than it. The default value is 0.
Individual Output	• Gain (K[1..8]) – Lets you specify a gain (K) value to be factored into the equation for calculating the CV output value for each individual output. See the equation following this table for details. The default value is 1. • Output Bias (OPBIAS[1..8].FIX) – Lets you specify a fixed bias value in engineering units that is added to the Calculated Variable (CV) output value for each individual output. See the Output Bias section for this function block for details. The default value is 0, which means no value is added. • Output Bias Rate (OPBIAS[1..8].RATE) – Lets you specify an output floating bias ramp rate in engineering units per minute for each individual output. This bias rate is only applied when the floating bias is non-zero. See the Output Bias section for this function block for details. The default value is Not a Number (NaN), which means no floating bias is calculated. As a result, if the primary does not accept the block's initialization value, a bump in OP occurs. • Enable Secondary Initialization Option (SECINITOPT[1..8]) – Lets you specify if the block is to ignore initialization and override requests from the secondary or not for each individual output. The default selection is Enabled (checked, do not ignore).

Configuration Tab	Description
Alarms	• Type – Identifies the types of alarms this block supports. Of course, these alarms also interact with other block configuration options such as the Safety Interlock Option (SIOPT) and Bad Control Option (BADCTLOPT). The types are: – Safety Interlock (SIALM.FL) – Bad Control (BADCTLALM.FL) • Enable Alarm (SIALM.OPT) – Lets you enable or disable Safety Interlock alarm type. A check in the box means the alarm is enabled. The default selection is a checked box or enabled (Yes). You can also configure the SIALM.OPT parameter as a block pin, a configuration and/or a monitoring parameter so it appears on the block in the Project and Monitoring tree view, respectively. • Priority – Lets you set the desired priority level individually for each alarm type (SIALM.PR, BADCTLALM.PR). The default value is LOW. The levels are: – NONE - Alarm is neither reported nor annunciated. – JOURNAL - Alarm is logged but it does not appear on the Alarm Summary display. – LOW, HIGH, URGENT - Alarm is annunciated and appears on the Alarm Summary display. • Severity – Lets you assign a relative severity individually for each alarm type (SIALM.SV, BADCTLALM.SV) as a number between 0 to 15, with 15 being the most severe. This determines the alarm processing order relative to other alarms. The default value is 0.
SCM	• SCM Mode Tracking Option (MODETRACK) – Lets you select the desired Mode Tracking function for the SCM associated with this block's Control Module. It defines how the FB will set the state of the MODEATTR based upon the MODE of the SCM. See the *Sequential Control Module User's Guide* for more information on this function. The default selection is

Configuration Tab	Description
	ONESHOT. The selections are: • – None – ONESHOT – SEMICONT – CONTRTN – CONT • Option Type – Lets you specify the action the function block is to take when the SCM goes into an abnormal state. The Starting State Option (STARTOPT) applies when the SCM state is Checking, Idle, or Complete. The Stop/Abort State Option (STOPOPT) applies when the SCM state is Stopping or Stopped, Aborting or Aborted. The Hold State Option (HOLDOPT) applies when the SCM state is Holding or Hold. The Restart State Option (RESTARTOPT) applies when the SCM state is Resume or Run. The NONE and LASTREQ are the only selections for the Restart State Option. You can select from one of these types for the other options as applicable for the given regulatory control function block: – NONE - No changes. – MAN - Set MODEREQ = MANUAL. – AUTO - Set MODEREQ = AUTOMATIC (Not applicable for this block). – CAS - Set MODEREQ = CASCADE. – FIXEDOP - Set OPREQ = Configured Value. – HOLDPV - Set SPREQ = PV (Not applicable for this block). – FIXED SP - Set SPREQ = Configured Value and SPRATEREQ = NaN (Not applicable for this block). – RAMPEDSP - Set SPREQ = Configured Value and SPRATEREQ = Configured Rate (Not applicable for this block).

Configuration Tab	Description
	• Value (STARTVAL, STOPVAL, HOLDVAL) – Depending upon Option Type selection, lets you specify an output or set point value within the respective range. For output, within OPLOLM to OPHILM and within SPLOLM to SPHILM, for set point. The default value is NaN (Not a Number).
Block Pins	Lets you select the available parameters that you want to expose as input/output pins on the function block graphic in Control Builder.
Configuration Parameters	Lets you select the available parameters that you want to appear on the face of the function block in the Project tab in Control Builder.
Monitoring Parameters	Lets you select the available parameters that you want to appear on the face of the function block in the Monitoring tab in Control Builder.
Block Preferences	Lets you change several block-viewing preferences including the color of the block's faceplate.

Each output value (CV [1..8]) is calculated using the following equation:

$$CV(n) = X1 * K(n) + [OPBIAS(n).FIX + OPBIAS(n).FLOAT]$$

where:

X1	=	input value
K(n)	=	gain for output CV(n) (user-specified)
(n)	=	output channel (number1 to 8)
OPBIAS(n).FIX	=	fixed bias for output CV(n) (user-specified)
OPBIAS(n).FLOAT	=	floating bias for output CV(n) (calculated)

- A separate gain [K(n)] and bias [OPBIAS(n).FIX] may be specified for each output.
- K(n) and OPBIAS(n).FIX may either be fixed (that is, stored manually or by the program) or external (that is, brought from another block). You can specify a different gain and fixed bias value for each output.
- The FANOUT block applies a separate floating bias to each output.

- The OP% is the CV expressed as a percentage of the CV range for that secondary.

  The CV may be used to calculate the OP which is given by:

  OP = (CV - CVEULO) /CVEUSPANBY100
  where:
  CVEUSPANBY100 = (CVEUHI-CVEULO)/ 100.

  The values for CVEUHI and CVEULO are set to be the same as the values for PVEUHI and PVEULO for the secondary. The PVEUHI and PVEULO values are in turn input by the user.

After an initialization, the block calculates OPBIAS(n).FLOAT for each output as:

$OPBIAS(n).FLOAT = CV_{INIT}(n) - [K(n) * X1 + OPBIAS(n).FIX]$
where:

(n) = output channel (number 1 to 8)

$CV_{INIT}(n)$ = CV(n) during initialization

ATTENTION

The FANOUT block is the only Regulatory Control Block that can have multiple secondaries.

Function

The FANOUT block provides a "bumpless" output for each of up to eight outputs following initialization or mode changes.

Configuration example

See the previous *Example of CB configuration using AUTOMAN block.* figure for an example of a FANOUT block being used to provide multiple outputs from a single PID block.

Inputs

The FANOUT block requires one input – X1:

- X1 = initializable input which must come from another block (it cannot be set by an operator or a program).
- You must specify an engineering unit range (XEUHI and XEULO) for X1. The block applies no range check. It assumes that X1 is within the specified range.
- XEUHI and XEULO define the full range of X1:
  - XEUHI represents the 100% of full scale value.
  - XEULO represents the 0% of full scale value.

ATTENTION

The FANOUT block:

- has 1 input and as many as 8 initializable outputs, and
- has 1 primary and up to 8 secondaries.

It requests the primary to initialize when mode changes from CAScade to MANual.

Outputs

The FANOUT block may have up to 8 initializable outputs as follows:

- OP[1..8] – calculated output, in percent.
- OPEU[1..8] – calculated output, in engineering units.

Initializable inputs and outputs

"Initializable input" and "initializable output" are variable attributes, similar to data type or access level. A parameter with the "initializable" attribute has an associated BACKCALC parameter and, when a connection between an initializable input and initializable output is created, you can also create a BACKCALC connection between them. Control Builder automatically builds the required BACKCALC connections, so you don’t have to create them manually. These “implicit” build connections are “hidden” from view and the related parameter pins are not exposed on the control chart.

For example, if you connect OP from a FANOUT block to an AUTOMAN block or an AOCHANNEL block, Control Builder automatically creates the BACKCALCOUT to BACKCALCIN connection.

- For a given secondary, a connection to OP or OPEU may be created, but not to both. (The default OP connection is exposed, but the implicit/hidden connection function automatically makes a connection to a value/status parameter (OPX/OPEUX) when it is required.)
- A separate gain and bias may be specified for each output.
- The FANOUT block applies a separate floating bias to each output.
- Gain limits may be configured with negative values, thereby making it possible to reverse outputs by using negative gains.
- The FANOUT block provides the X1 input range (XEUHI/XEULO) to the primary through BACKCALC. The primary uses this for its output range (CVEUHI/CVEULO).

Output ranges

- CVEUHI[1..8] and CVEULO[1..8] define the full range of CV in engineering units for each given output.
  - The FANOUT block does separate ranging for each output by maintaining a separate CV range for each output which tracks the input range of the corresponding secondary.
  - The CV range for each output must be the same as the input range of each secondary. The FANOUT block brings the input range from each secondary (through BACKCALC) and stores it as the corresponding CV range. As a result, each output may have a different CV range. For example, a FANOUT block has its outputs OP[1] and OP[2] connected to blocks PID1 and PID2, respectively. It brings the input ranges of PID1 and PID2 and sets its CV ranges of OPX[1] and OPX[2] to these input ranges, respectively.
  - The FANOUT block brings the secondary's input range regardless of SECINITOPT (that is, regardless of whether the secondary's initialization and override data will be used).
- OPHILM and OPLOLM define the normal high and low limits for OP as a percent of the CV range. These are user-specified values. The same limits apply to all outputs.

  - OP is clamped to these limits if the algorithm's calculated result (CV) exceeds them or another function block or the user program attempts to store an OP value that exceeds them. However, the operator may store an OP value that is outside these limits.

- OPEXHILM and OPEXLOLM define the extended high and low limits for OP as a percent of the CV range. These are user-specified values. The same limits apply to all outputs. The operator is prevented from storing an OP that exceeds these limits.

Output bias

The output bias (OPBIAS) is added to the algorithm's Calculated Value (CV) and the result is stored in CV. CV is later checked against OP limits and, if no limits are exceeded, copied to the output. Since the FANOUT block can have up to eight outputs, a separate output bias is determined for each output. This means that the parameters referenced in this discussion are actually indexed to the given output. For example, OPBIAS[1] and CV[1] are indexed to OP[1], and so on for the other seven outputs numbered 2 to 8.

The OPBIAS is the sum of the user-specified fixed bias (OPBIAS.FIX) and a calculated floating bias (OPBIAS.FLOAT). The purpose of the floating bias is to provide a bumpless transfer when the function block initializes or changes mode as long as the FANOUT block is the first initializable block.

- OPBIAS is recomputed under the following conditions to avoid a bump in the output. (Note that the function block only applies OPBIAS.FLOAT to the output for the latter two conditions, when it is the first initializable block.)
  - When the function block starts up (that is, goes Active).
  - When the function block initializes (for example, the secondary requests initialization).
  - When the mode changes to Cascade (as applicable for the given block).

ATTENTION

When the mode is Manual, OPBIAS is not used, since the output is not calculated. This means that the OPBIAS is not recomputed, when the mode changes to Manual.

- You may set the OPBIAS value only if the function block is Inactive or Mode equals Manual. This is done to prevent a bump in the output when the bias is changed. The following occurs when you set the OPBIAS value.

- The total bias (OPBIAS) and fixed bias (OPBIAS.FIX) are both set to the entered value.
- The floating bias (OPBIAS.FLOAT) is set to zero.

ATTENTION

When the function block goes Active or the Mode changes to Cascade (as applicable for the given block), OPBIAS and OPBIAS.FLOAT are recomputed.

- There are no limit checks applied when you set an OPBIAS value. However, after the bias is added to CV, the result is compared against the output limits and clamped, if necessary.
- You configure the value for the fixed bias (OPBIAS.FIX) and it is never overwritten by the floating bias (OPBIAS.FLOAT). This means the total bias will eventually equal the OPBIAS.FIX , if you configure OPBIAS.RATE to ramp down OPBIAS.FLOAT.

The OPBIAS.FLOAT is calculated as follows.

OPBIAS.FLOAT	=	$CV_{INIT} – (CV_{UNBIASED} + OPBIAS.FIX)$
Where:		
CV_{INIT}	=	initialization value received from the secondary
$CV_{UNBIASED}$	=	unbiased calculated value (based on input from the primary)
OPBIAS.FIX	=	fixed bias (user-specified)

- If the primary accepts this block's initialization request, then CV + OPBIAS.FIX should be the same as CVINIT and OPBIAS.FLOAT will be zero. In most cases, OPBIAS.FLOAT will be zero. However, if the primary does not accept this block's initialization request because the primary is a FANOUT block or it was configured to ignore initialization, then OPBIAS.FLOAT value will not be zero.

  If OPBIAS.FLOAT is not zero, you can configure it to ramp down to zero through the OPBIAS.RATE parameter.
- You configure the OPBIAS.RATE to apply a ramprate to the OPBIAS.FLOAT. It is only used when the OPBIAS.FLOAT is not zero. The OPBIAS.RATE is expressed in Engineering Units per minute and may have the following values.

- Zero:
  If OPBIAS.RATE is zero, a OPBIAS.FLOAT is calculated and bumpless transfer is guaranteed. However, if OPBIAS.FLOAT is not zero, it will never ramp down.

- Non-zero:
  If OPBIAS.RATE is not zero, an OPBIAS.FLOAT is calculated and bumpless transfer is guaranteed. If the OPBIAS.FLOAT is not zero, it is ramped to zero at the rate you configured for the OPBIAS.RATE parameter.

- The function block ramps the OPBIAS.FLOAT to zero by applying the following calculation each time it executes.

OPBIAS.FLOAT = OPBIAS.FLOAT – (OPBIAS.RATE / cycles_per_Min)

Where:

cycles_per_min = number of times the function block executes per minute (calculated)

- NaN:
  When the OPBIAS.RATE is Not a Number (NaN), no OPBIAS.FLOAT is calculated. This means a bump in the output will occur, if the primary does not accept this block's initialization value.

Mode handling

The FANOUT block supports both the Cascade and Manual modes:

- If mode is CAScade, then: X1 must be pulled from another block.
- If mode is MANual, then: OP may be stored by the operator or a user-program (X1 is ignored).

Timeout monitoring

If mode is CAScade, the FANOUT block performs timeout monitoring on X1. If the X1 value is not updated within a predefined time (TMOUTTIME), the FANOUT block invokes timeout processing as follows:

- Sets the "input timeout" flag (TMOUTFL).
- Sets the input value to Bad (NaN – Not a Number).
- Requests the X1 primary to initialize (through X1BACKCALCOUT).

The FANOUT block does not support mode shedding on timeout.

The maximum time between updates is specified by TMOUTTIME (in seconds)

- Enable timeout monitoring by setting TMOUTTIME to a non-zero value.
- Disable timeout monitoring by setting TMOUTTIME to zero.

ATTENTION

If the input is from a connection in another controller in a peer-to-peer architecture, the actual timeout time equals the configured TMOUTTIME plus the CDA timeout time. The CDA timeout time equals four times the configured CEE subscription rate. For example, if the CEE subscription rate is 100 milliseconds and the TMOUTTIME is 5 seconds, the actual timeout time for the block is 4 times 100ms plus 5s or 5.4 seconds.

Control initialization

The FANOUT block brings initialization requests from its secondary through BACKCALC. In addition, the secondary may propagate one-shot initialization requests to the FANOUT block.

If all secondaries are requesting initialization and the SECINITOPT for corresponding outputs is enabled, the FANOUT block –

- Initializes its output so CV[1..8] = INITVAL[1..8] from corresponding secondary

- Builds an initialization request for the primary as follows:

INITREQ(X1) = On

$$INITVAL(X1) = \frac{CV - OPBIAS(last).FIX}{K(last)}$$

- Where:

(last) = last or most recent secondary to request initialization. When more than one secondary requests initialization simultaneously and the FANOUT initializes, “last” will be the secondary with the lowest index number from the secondaries that requested initialization.

CV		calculated value
OPBIAS(last).FIX	=	fixed output bias for the last secondary
K(last)	=	gain for the last secondary
INITREQ(X1)	=	initialization request flag for the X1 primary
INITVAL(X1)	=	initialization value for the X1 primary

ATTENTION

- SECINITOPT may be used to ignore initialization requests from selected secondaries.
- When more than one secondary initializes simultaneously and the FANOUT block initializes, the “LAST” in the previous equation represents the secondary corresponding to the lowest numbered output that requested initialization.
- The FANOUT block only performs initialization if all of the secondaries are requesting it. As long as one secondary is not requesting initialization, the FANOUT block ignores all requests. For this reason, we recommend that you use an AUTOMAN block between a FANOUT block output and a given AOCHANNEL block to provide a "bumpless" output after any mode change. However, To prevent a bump in the output, you must configure the AUTOMAN block OPBIAS.RATE parameter for a value (in Engineering Units per minute) other than 0.0 (zero) or NaN (Not a Number) to enable the ramping function for the floating bias.

- When FANOUT block comes out of initialization, it calculates INITVAL(X1) as follows:

$$\text{INTVAL (X1)} = \frac{\text{CV - OPBIAS (index).FIX}}{\text{K (index)}}$$

  - Where:

(Index)	=	output channel number (1 to 8) corresponding to the secondary which is not requesting initialization any more. If more than one secondary stops requesting initialization simultaneously, “Index” represents the secondary corresponding to the lowest numbered output that stopped requesting initialization.

Secondary initialization option

If a BACKCALC connection is made, the primary always brings initialization data over this connection. However, you can configure the block to ignore this data by not selecting the Enable Secondary Initialization Option on the block's parameter configuration form. This is the same as selecting disable as the setting for the SECINITOPT parameter. The results of the SECINITOPT settings are as follows.

- If SECINITOPT equals Enable, it means the function block should accept initialization and override requests from the secondary.
- If SECINITOP equals Disable, it means the function block should ignore initialization and override requests from the secondary.

Since the FANOUT block can have up to eight secondaries, you can selectively enable/disable the SECINITOPT for each output.

Override feedback processing

The FANOUT block does not propagate override data to its primaries.

BACKCALC processing

BACKCALC contains initialization, windup, and range data from each secondary. The FANOUT block always uses the secondary's windup status and range data, and you may specify whether to ignore initialization through the SECINITOPT parameter. There is 1 SECINITOPT per secondary.

Since initialization and windup data may be received from multiple secondaries, the FANOUT block applies the following rules to decide what it should propagate from its secondaries:

e) Initialization is propagated only if all secondaries are requesting it. The FANOUT block uses the initialization value from the last secondary to request it. SECINITOPT may be used to ignore initialization requests from selected secondaries.

f) Refer to Windup Processing below.

Windup processing in FANOUT block

Windup is propagated only if all secondaries agree. The FANOUT block uses the windup status from all secondaries regardless of SECINITOPT. The FANOUT block only propagates a high or low windup status to its primary under the following conditions:

- If all secondaries are in high windup, the FANOUT block propagates a high windup status to its primary (ARWNET = Hi).
- If all secondaries are in low windup, the FANOUT block propagates a low windup status to its primary (ARWAY = Lo).

Note that if the gain is reversed for one of the outputs, then high windup on that output will be the same as low windup on the others.

The FANOUT block propagates a normal windup status to its primary under the following conditions:

- If at least one secondary has a normal windup status.
- If at least one secondary is in Hi windup and another is in Lo.

Note that the FANOUT block checks the windup status from all secondaries, regardless of SECINITOPT selection.

Windup processing in Regulatory blocks

Every regulatory control type block maintains anti-reset windup status for its output (ARWOP) and each of its initializable inputs (ARWNET). The following table lists the posssible values for ARWOP and ARWNET parameters.

If the Value is . . .	Then, the Associated Parameter . . .
Normal	is free to move in either direction.
Hi	is at its high limit and it may only be lowered.
Lo	is at its low limit and it may only be raised.
HiLo	may not move in either direction.

Manual Mode Interaction

When the MODE of a regulatory control block is changed to Manual (Man), the block sets its windup status (ARWNET) to HiLo. This means that every block upstream in a cascade strategy will set its windup status (ARWNET and ARWOP) to HiLo.

ARWOP computation

The ARWOP indicates if the output (OP) can be raised or lowered. The PID-type function blocks use ARWOP to restrict integral control. When ARWOP contains a value other than **Normal**, the PID block stops integral control in the windup direction. Integral control continues in the other direction, as does proportional and derivative control. But, windup status has no impact on proportional and derivative control.

If a function block has a secondary, it fetches the secondary's windup status and recomputes its ARWOP. The conditions within the function block, such as output being at its high limit, also affect the ARWOP. The ARWOP is computed as follows, assuming the block has only one output or that it is not a FANOUT block.

If Any of the Following are True . . .	Then, ARWOP Equals . . .
This block is inactive.	**HiLo**
A secondary exists but this block cannot fetch secondary data from it (communications or configuration error).	
A secondary exists and its windup state equals HiLo	
This block is in initialization (INITMAN = On).	
A secondary exists and it is requesting this block to initialize.	
A secondary exists and its windup state equals Hi.	**Hi**
This block's output is at its high limit (OPHIFL = On).	
A secondary exists and its windup state equals Lo.	**Lo**
This block's output is at its low limit (OPLOFL = On).	

ARWNET computation

When ARWNET is HiLo, stores to SP are not limited, rather this is the status propagated to the primary. The only *limiting* anti-reset windup status ever does is to stop integral action in one or both directions on PID blocks. For any other regulatory control type block, ARWNET is not used for any kind of limiting. The ARWNET is computed as follows, assuming the block has only one output or that it is not a FANOUT block.

If Any of the Following are True . . .	Then, ARWNET Equals . . .
This block is inactive.	**HiLo**
The ARWOP equals HiLo.	
This block is in Manual mode (MODE = Man)	
The calculated value (CV) range (CVEUHI / CVEULO) is NaN.	
The CV is NaN	
This block is connected to a non-initializable primary	
The ARWOP equals Hi (Pid function blocks have a configurable Control Action option (CTLACTN). If CTLACTN = Reverse , ARWNET will track ARWOP; but if CTLACTN = Direct , ARWNET will be the opposite of ARWOP.)	**Hi**
The input from the primary is at a high limit. For example, SPHIFL = On.	
This block's output has reached its positive rate-of-change limit (OPROCPOSFL = On)	

If Any of the Following are True . . .	Then, ARWNET Equals . . .
The ARWOP equals Lo (Pid function blocks have a configurable Control Action option (CTLACTN). If CTLACTN = Reverse , ARWNET will track ARWOP; but if CTLACTN = Direct , ARWNET will be the opposite of ARWOP.)	**LO**
Input from the primary is at a low limit. For example, SPLO.FL = On.	
This block's output has reached its negative rate-of-change limit (OPROCNEGFL = On)	

Anti-Reset Windup Status

The anti-reset windup network in (ARWNETIN) and anti-reset windup output in (ARWOPIN) parameters are added in the standard anti-reset windup (ARW) computation logic. They are user configurable and allow stores from Sequential Control Modules (SCMs) and control algorithm block (CAB) programs.

The ARWNETIN and ARWOPIN parameters would be *ORed* into the existing standard logic so it is not lost. The following table summarizes the influence the ARWNETIN and ARWOPIN parameters have on the ARWNET and ARWOP parameters, which are not user configurable.

ARWNETIN or ARWOPIN Parameter Is. . .	Standard Computation Logic Is . . .	ARWNET or ARWOP Parameter Is . . .
NORMAL	NORMAL	NORMAL
NORMAL	HI	HI
NORMAL	LO	LO
NORMAL	HILO	HILO
HI	NORMAL	HI
HI	HI	HI
HI	LO	HILO
HI	HILO	HILO

ARWNETIN or ARWOPIN Parameter Is. . .	Standard Computation Logic Is . . .	ARWNET or ARWOP Parameter Is . . .
LO	NORMAL	LO
LO	HI	HILO
LO	LO	LO
LO	HILO	HILO
HILO	NORMAL	HILO
HILO	HI	HILO
HILO	LO	HILO
HILO	HILO	HILO

FANOUT parameters

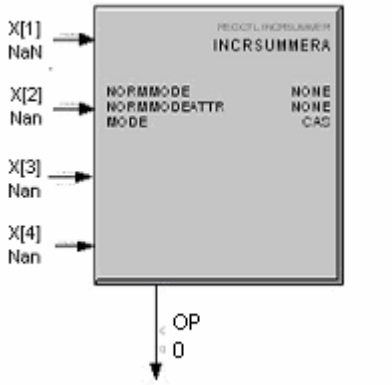
REFERENCE - INTERNAL

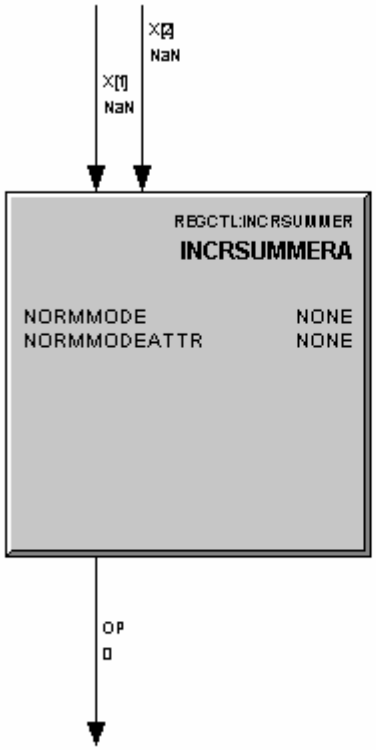
Refer to the *Control Builder Components Reference* for a complete list of the parameters used with the FANOUT block.

INCRSUMMER (Incremental Summer) Block

Description

The IncrSummer block calculates the sum of the incremental changes in up to four input values X1 to X4. The output OP is obtained by adding the sum of the changes in all the inputs after each input is multiplied by its respective scale factor K(1) to K(4). It looks like this graphically:





Each INCRSUMMER block supports the following user configurable attributes. The following table lists the given name of the “Tab” in the parameter configuration form and then briefly describes the attributes associated with that Tab. This data is only provided as a quick document reference, since this same information is included in the on-line context sensitive Help.

Configuration Tab	Description
Main	• Name – Block (Tag) name of up to 16 characters long. Must be unique within the CM block containing it. • Description (DESC) – Block descriptor of up to 24 characters long. • Engineering Units (EUDESC) – Lets you specify a text string of up to 16 characters to identify the variable values associated with this block. For example, you could specify DEGF for temperature values in degrees Fahrenheit. This name is used on any associated displays and generated reports. • Execution Order in CM (ORDERINCM) – Specifies the execution order of the block in the CM relative to other blocks contained in this CM. Enter as a number between 1 and 32767. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. • Normal Mode (NORMMODE) – Lets you specify the MODE the block is to assume when the Control to Normal function is initiated through the Station display. Selections are MANual, AUTOmatic, CAScade, BackupCAScade and NONE. All selections are not valid for a given block. The default selection is NONE. • Normal Mode Attribute (NORMMODEATTR) – Lets you specify the mode attribute (MODEATTR) the block is to assume when the Control to Normal function is initiated through the Station display. Selections are NONE, OPERATOR and PROGRAM. The default selection is NONE. • Mode (MODE) – Lets you set the block's current MODE. The selections are MANual, AUTOmatic, CAScade, BackupCAScade, NONE, and NORMAL. All selections are not valid for a given block. The default selection is MANual. MODE identifies who may store values to the block's initializable inputs or output. Blocks strictly enforce the MODE assignment. Some function blocks perform automatic mode switching (or mode shedding), while others require manual intervention. The block's MODE is derived at "runtime"

Configuration Tab	Description
	based on current conditions. MODE processing checks for the following conditions, and changes the block's MODE as appropriate. – External request for MODE switching. – Safety interlock request. • Mode Attribute (MODEATTR) – Lets you set the block's mode attribute. The selections are NONE, OPERATOR, PROGRAM, and NORMAL. The default selection is OPERATOR. MODEATTR identifies who may store values to the output (OP), when the block's MODE is MANual. • Permit Operator Mode Changes (MODEPERM) – Lets you specify if operators are permitted to make MODE changes or not. The default is Enabled (checked). A store to MODE does not change the NORMMODE. • Permit External Mode Switching (ESWPERM) – Lets you specify if external MODE switching through user configured interlocks is permitted or not, if you have at least an Engineering access level. The default is Disabled (unchecked). • Enable External Mode Switching (ESWENB) – Lets you specify if external MODE switching through user configured interlocks is enabled or not, if ESWPERM is checked (Permitted). The default is Disabled (unchecked). • Enable Secondary Initialization Option (SECINITOPT) – Lets you specify if the block is to ignore initialization and override requests from the secondary or not. The default selection is Enabled (checked, do not ignore). • Safety Interlock Option (SIOPT) – Lets you specify MODE and OP block is to assume upon a safety interlock alarm. The selections are NO_SHED, SHEDHOLD, SHEDLOW, SHEDHIGH, and SHEDSAFE. The default selection is SHEDHOLD. • Bad Control Option (BADCTLOPT) – Lets you specify MODE and OP block is to assume if CV goes BAD. The selections are NO_SHED, SHEDHOLD, SHEDLOW, SHEDHIGH, and SHEDSAFE. The default

Configuration Tab	Description
	selection is NOSHED.
Input	• High Limit (XEUHI) – Lets you specify the high input range limit that represents 100% full-scale input for the block. The default value is 100. • Low Limit (XEULO) – Lets you specify the low input range limit that represents the 0 full-scale input for the block. The default value is 0 (zero). • XDESC[1..4] - Lets you specify a text string of up to 23 characters to identify inputs 1..4. • K(1)..K(4) – Lets you specify the scaling factors for inputs 1..4 – Timeout Time (TMOUTTIME) – Lets you specify a time in seconds that must expire before the block assumes that its input update has timed out. The block must be in CAScade mode for it to monitor its primary input for timeout. The default setting is 0, which means the timeout function is disabled. If the input is from a connection in another controller in a peer-to-peer architecture, the actual timeout time equals the configured TMOUTTIME plus the CDA timeout time. The CDA timeout time equals four times the configured CEE subscription rate. For example, if the CEE subscription rate is 100 milliseconds and the TMOUTTIME is 5 seconds, the actual timeout time for the block is 4 times 100ms plus 5s or 5.4 seconds.
Output	• High Limit (%) (OPHILM) – Lets you specify the output high limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you enter a High Limit of 90%, the high limit in engineering units is 90% times 450 or 405 + 50 (CVEULO) equals 455. This check is not applied for a function block that is in the MANual mode. The default value is 105%. • Low Limit (%) (OPLOLM) – Lets you specify the output low limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you enter a Low Limit of 10%, the low limit in engineering units is 10% times 450 or 45 + 50 (CVEULO) equals 95. This check is not applied for a function block that is in the MANual

Configuration Tab	Description
	mode. The default value is -5%. • Extended High Limit (%) (OPEXHILM) – Lets you specify the output extended high limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you use the default value of 106.9%, the extended high limit in engineering units is 106.9% times 450 or 481.05 + 50 (CVEULO) equals 531.05. This check is not applied for a function block that is in the MANual mode. The default value is 106.9%. • Extended Low Limit (%) (OPEXLOLM) – Lets you specify the output extended low limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you use the default value of -6.9%, the extended low limit in engineering units is -6.9% times 450 or -31.05 + 50 (CVEULO) equals 18.95. This check is not applied for a function block that is in the MANual mode. The default value is -6.9%. • Rate of Change Limit (%) (OPROCLM) – Lets you specify a maximum output rate of change limit for both the positive and negative directions of the output in percent per minute. This lets you prevent an excessive rate of change in the output so you can match the slew rate of the control element to the control dynamics. We recommend that you configure this value before you tune the loop, so tuning can accommodate any slow down in response time caused by this rate limiting. This check is not applied for a function block that is in the MANual mode. The default value is Not a Number (NaN), which means no rate limiting is applied. • Minimum Change (%) (OPMINCHG) – Lets you specify an output minimum change limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). This lets you define how much the OP must change before the function block outputs a new value. It filters out changes that are too small for the final control element to respond to. This check is not applied for a function block that is in the MANual mode. The default value is 0, which means no change limiting is applied. • Safe OP (%) (SAFEOP) – Lets you specify the safe

Configuration Tab	Description
	output value as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, if the CV range is 0 to 500 and you enter a Safe OP of 50%, the safe output value in engineering units is 50% times 500 or 250. The default value is Not a Number (NaN), which means the OP is held at its last good value. • Output Bias (OPBIAS.FIX) – Lets you specify a fixed bias value in engineering units that is added to the Calculated Variable (CV) output value. See the Output Bias section for this function block for details. The default value is 0, which means no value is added. • Output Tolerance (OPTOL) – Lets you specify a limit for operator entered output changes in %. Operator-entered values greater than OPTOL will result in a warning to the operator which must be overridden to permit the value to be entered.
Alarms	• Type – Identifies the types of alarm this block supports. Of course, these alarms also interact with other block configuration options such as the Safety Interlock Option (SIOPT) and Bad Control Option (BADCTLOPT). The types are: – Safety Interlock (SIALM.FL) – Bad Control (BADCTLALM.FL) – OP High (OPHIALM.FL) – OP Low (OPLOALM.FL) • Enable Alarm (SIALM.OPT) – Lets you enable or disable Safety Interlock alarm type. A check in the box means the alarm is enabled. The default selection is a checked box or enabled (Yes). You can also configure the SIALM.OPT parameter as a block pin, a configuration and/or a monitoring parameter so it appears on the block in the Project and Monitoring tree view, respectively. • Trip Point – Lets you specify the OP High Alarm (OPHIALM.TP) and OP Low Alarm (OPLOALM.TP) trip points in percent. The default value is NaN, which disables the trip point.

Configuration Tab	Description
	• Priority – Lets you set the desired priority level individually for each alarm type (SIALM.PR, BADCTLALM.PR, OPHIALM.PR, and OPLOALM.PR). The default value is LOW. The levels are: – NONE - Alarm is neither reported nor annunciated. – JOURNAL - Alarm is logged but it does not appear on the Alarm Summary display. – LOW, HIGH, URGENT - Alarm is annunciated and appears on the Alarm Summary display. • Severity – Lets you assign a relative severity individually for each alarm type (SIALM.SV, BADCTLALM.SV, OPHIALM.SV, and OPLOALM.SV) as a number between 0 and 15, with 15 being the most severe. This determines the alarm processing order relative to other alarms. The default value is 0. • Deadband Value (ALMDB) – Lets you specify a deadband value that applies to all analog alarms to prevent nuisance alarms due to noise at values near the trip point. The default value is 1. Note that this value is loaded to the individual alarm parameters (for example, OPHIALM.DB and OPLOALM.DB) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB. • Filter Time (ALMTM) – Lets you specify a time in seconds to define how long an analog alarm must exist before it is set true. The default value is 0, which means the alarm is set true as soon as the value exceeds the deadband value. Note that this value is loaded to the individual alarm parameters (for example, OPHIALM.TM and OPLOALM.TM) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB. • Deadband Units (ALMDBU) – Lets you specify if the

Configuration Tab	Description
	deadband value represents percent or engineering units. The default value is percent. Note that this value is loaded to the individual alarm parameters (for example, OPHIALM.DBU and OPLOALM.DBU) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB.
SCM	– SCM Mode Tracking Option (MODETRACK) – Lets you select the desired Mode Tracking function for the SCM associated with this block's Control Module. It defines how the FB will set the state of the MODEATTR based upon the MODE of the SCM. See the *Sequential Control Module User's Guide* for more information on this function. The default selection is ONESHOT. The selections are: – None – ONESHOT – SEMICONT – CONTRTN – CONT • Option Type – Lets you specify the action the function block is to take when the SCM goes into an abnormal state. The Starting State Option (STARTOPT) applies when the SCM state is Checking, Idle, or Complete. The Stop/Abort State Option (STOPOPT) applies when the SCM state is Stopping or Stopped, Aborting or Aborted. The Hold State Option (HOLDOPT) applies when the SCM state is Holding or Hold. The Restart State Option (RESTARTOPT) applies when the SCM state is Resume or Run. The NONE and LASTREQ are the only selections for the Restart State Option. You can select from one of these types for the other options as applicable for the given regulatory control function block: – NONE - No changes. – MAN - Set MODEREQ = MANUAL.

Configuration Tab	Description
	– AUTO - Set MODEREQ = AUTOMATIC (not applicable to this block). – CAS - Set MODEREQ = CASCADE. – FIXEDOP - Set OPREQ = Configured Value. – HOLDPV - Set SPREQ = PV. – FIXED SP - Set SPREQ = Configured Value and SPRATEREQ = NaN. – RAMPEDSP - Set SPTVREQ = Configured Value and SPRATEREQ = Configured Rate. • Value (STARTVAL, STOPVAL, HOLDVAL) – Depending upon Option Type selection, lets you specify an output or set point value within the respective range. For output, within OPLOLM to OPHILM and within SPLOLM to SPHILM, for set point. The default value is NaN (Not a Number).
Block Pins	Lets you select the available parameters that you want to expose as input/output pins on the function block graphic in Control Builder.
Configuration Parameters	Lets you select the available parameters that you want to appear on the face of the function block in the Project tab in Control Builder.
Monitoring Parameters	Lets you select the available parameters that you want to appear on the face of the function block in the Monitoring tab in Control Builder.
Block Preferences	Lets you change several block-viewing preferences including the color of the block's faceplate.
Insertion Type	Lets you include an insertion type from a CAB instances in the block. See ***CAB insertion configuration considerations for*** regulatory control blocks for more information
Identification	Lets you view information pertinent to the qualfication and version control system and enter block comments, if desired.
Dependencies	Lets you view block hierarchical information.
Template Defining	Lets you view and define parameters for associcated templates.

Equation

There is only one equation provided for this block.

CV is calculated as follows:

$$CV(n) = CV(n-1) + K(1)*[X1(n) - X1(n-1)] + K(2)*[X2(n) - X2(n-1)] + K(3)*[X3(n) - X3(n-1)] + K(4)*[X4(n) - X4(n-1)] + OPBIAS$$

where:

CV (n) = Current CV value of this algorithm in engineering units.

CV (n-1) = Past full value of the output of this algorithm in engineering units

K(1) through K(4) = User-specified scale factors (gains). K(1) through K(4) default 1.0.

X1 (n) through X4 (n) = Current values of each X input in use.

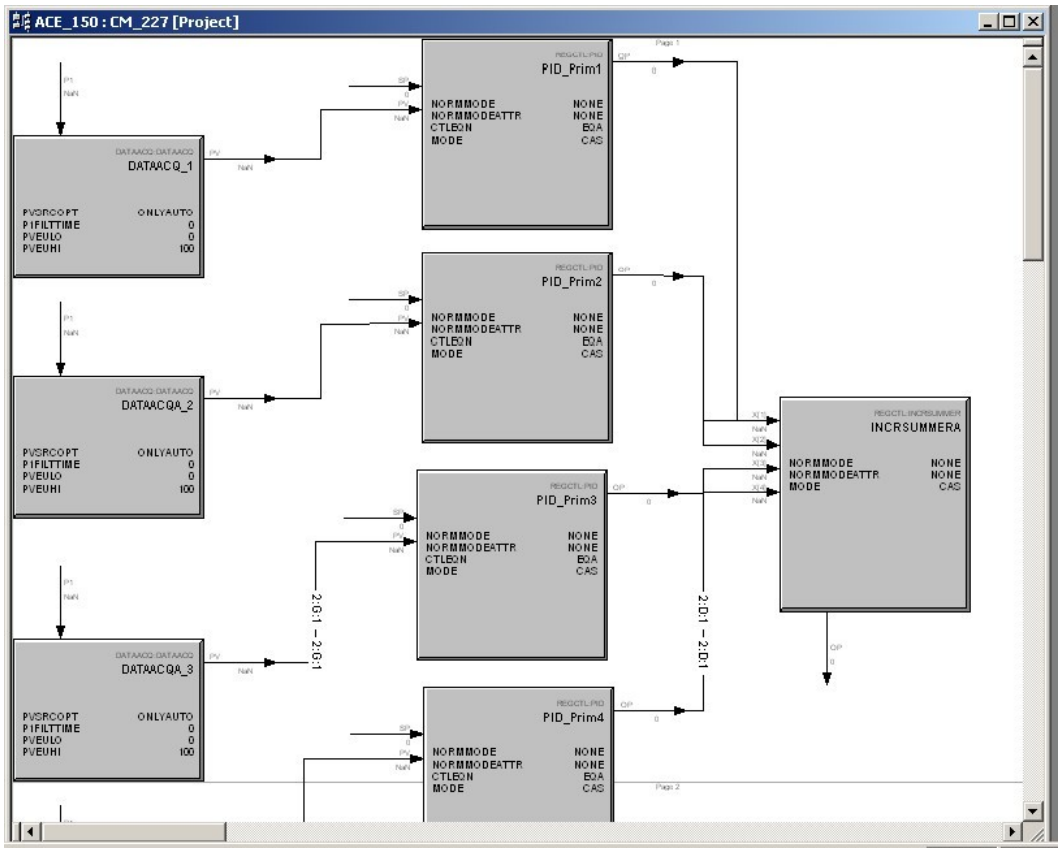
X1 (n-1) through X4 (n-1) = Past value of each X input (value from the last time the data point was processed).

OPBIAS = Total bias (OPBIAS.FIX)

In the above equation if an input goes bad, that input is ignored in the calculation of CV. When the bad input returns to Normal, its use in the output computation is resumed in the CV calculation without bumping the value.

Configuration example

The following screen shot depicts the scenario wherein the INCRSUMMER block is used:



Function

This block is typically used where more than one primary data point is used to manipulate the setpoint of the same secondary data point. The primaries usually use PID algorithms, and are connected to an IncrSummer data popint. The output from the IncrSummer block is connected to a secondary. This is sometimes referred to as a "fan-in" connection.

Inputs

- The block takes up to four inputs X[1..4] with a scale factor for each input.
- X(1) through X(4) are initializable inputs. So this block can have up to four primaries.
- The block can also be loaded with no inputs connected but it cannot be activated unless at least two inputs are connected to the IncrSummer block.
- Inputs can be connected in any order. For example, X[1] and X[3] can be connected without X[2] connected in this block.
- Typically the primaries of IncrSummer are normal PID or a PID Feedforward.
- The X-input ranges XEUHI and XEULO must be configured for this block.
- Primaries to IncrSummer block have to be in the same Execution Environment (EE). An error is thrown and the block will be prevented from activation in case any primary of IncrSummer is in another peer, that is, in a different EE than IncrSummer block.

Outputs

The INCRSUMMER block has the following initializable outputs:

- OP - Calculated output, in percent
- OPEU - Calculated output, in engineering units

ATTENTION

The user may create a connection to OP or OPEU, but not both i.e. only one connection to the RegCtl block output should be made Therefore, this block may have only one secondary. If the user does not create a connection to OP or OPEU, then the block does not have a secondary. Alternately, if the user connects OP or OPEU to a non-initializable input, then this block does not have a secondary. If the block has a secondary, then the OPX or OPEUX is the proper parameter to connect to RegCtl secondary. The "X" parameter is a structure containing both the OP value and the OP status; it is critical to use these parameters so that Initialization handshaking works properly. The BACKCALCOUT or X1BACKCALOUT of secondary must be connected to the BACKCALCIN of primary.

Initializable inputs and outputs

"Initializable output" and "Initializable input" are variable attributes, similar to data type or access level. A parameter with the "initializable" attribute has an associated BACKCALC parameter, and when the user creates a connection between an initializable input and initializable output, the system will also create a BACKCALC connection. For example, X1 is an initializable input with an associated X1BACKCALOUT, and OP is an initializable output with BACKCALCIN. If the user creates a connection between X1 and OP, the system will also create a connection between X1BACKCALOUT and BACKCALCIN.

Output Ranges

- CVEUHI and CVEULO define the full range of CV, in engineering units.
- If this block has a secondary, it fetches the secondary's input range via BACKCALC and sets its CV range to that. If it has no secondary, CVEUHI and CVEULO track the X-input range (XEUHI and XEULO).
- **Note:** This block fetches the secondary's input range regardless of SECINITOPT (i.e., regardless of whether the secondary's initialization and override data will be used).
- The user has to set the CVEUHI and CVEULO such that it should be the same as that of its secondaries XEUHI and XEULO respectively, when the secondary is connected to IncrSummer, else if there's no secondary, the CV ranges of IncrSummer block should follow its own XEU ranges. In case, the CVEU ranges differ from the XEU ranges, then the results can be *unexpected*.

- OPHILM and OPLOLM define the normal high and low limits for OP, as a percent of the CV range. These are user-specified values.
- OP will be clamped to these limits if the algorithm's calculated result (CV) exceeds them, or another function block or user program attempts to store an OP value that exceeds them. However, the operator may store an OP value that is outside these limits.
-
- OPEXHILM and OPEXLOLM define the extended high and low limits for OP, as a percent of the CV range. These are user-specified values.
- The operator is prevented from storing an OP value that exceeds these limits.
- OPTOL allows the user to configure a tolerance limit for the manually entered OP. If the difference between new OP value and current OP value is greater the OPTOL then confirmation is required from the user to store this new value.

Output bias

The user may specify a fixed bias (OPBIAS.FIX) to be added to the output.

Mode handling

This function block supports the Cascade and Manual modes:

- If MODE = Cascade, all inputs must be pulled from another function block.
- CV value can be changed using CAB program or SCM step output. This causes a shift in the CV value but the dynamic changes to the CV value continue. This is because the IncrSummer dynamically increments or decrements CV each time the function block is processed.
- If MODE = Manual, OP may stored by the operator or a user program. (All inputs are ignored.).

The table below depicts the expected values of CV and X inputs when the Primary and Secondary are operated in different modes:

SNO	Primary (PID)	Secondary (IncrSummer)	Primary CV	Xn	Xn-1	Secondary CV
1	Cascade	Cascade	Every time the IncrSummer block executes it sets the primary PID's CV value to the IncrSummer's CV value	Fetched	CV	Calculated or Stored by CAB/SCM
2	Cascade	Manual	The CV of primary will be initialized to IncrSummer's InitVal.	Not fetched but initialized to Incremental Summer's CV value	NaN	While OP is supplied by operator, CV is back calculated from this OP
3	Manual	Manual	CV of primary will be initialized to Incremental Summer's InitVal.	Not fetched but initialized to Incremental Summer's CV value	NaN	While OP is supplied by operator, CV is back calculated from this OP
4	Manual	Cascade	CV of Primary is set to its OP value which is supplied by the operator.	Fetched	Xn	Calculated or Stored by CAB/SCM

Timeout monitoring

If MODE is Cascade, this block performs timeout monitoring on all inputs (X(1) - X(4)). If an input value is not updated within a predefined time, the block invokes timeout processing as follows:

If MODE is Cascade and an input times-out, this block does the following:

- Sets the "input timeout" flag (TMOUTFL)
- Sets the input value to Bad (NaN).
- Requests the input's primary to initialize

The INCRSUMMER block does *not* support mode shedding on time-out.

Control initialization

Inputs X1 to X4 are initializable. This block fetches initialization requests from its secondary via BACKCALC. If this block goes into initialization all the primary function blocks go into initialization. In addition, the secondary may propagate ones hot initialization requests to this block.

Note: SECINITOPT may be used to ignore initialization requests from the secondary.

If the secondary is requesting initialization, this block:

- Initializes its output:
  - If the initialization value received from the secondary is within the output limits of this function block:
  - CV = initialization value from the secondary
  - Otherwise, CV is clamped to the appropriate limit.
- Builds an initialization request for each of the X primaries as follows:

X(n).INITREQ = On

X(n).INITVAL = CV – OPBIAS.FIX

where

X(n).INITREQ = initialization request flag for the X(n) primary

X(n).INITVAL = initialization value for the X(n) primary

OPBIAS.FIX = User specified bias value.

n = 1 to 4

When the cascade is broken, inputs X1 to X4 go into initialization and get initialized to this Incremental Summer's CV value. The INITVAL sent to the primary is same as that mentioned above.

Override feedback processing

If the IncrSummer block's immediate secondary is an Override Selector block and its override status is not selected, the CV of the IncrSummer is set equal to override feedback value and it propagates the feedback value received to all its primaries. This happens as long as IncrSummer is non-selected primary of the Override Selector block. During the next execution of IncrSummer block, it computes the CV and this new CV value may allow it to become a selected input of the Override Selector and hence the CV value is changed from the override feedback value stored in the previous execution. Even if the immediate secondary of IncrSummer block is a PID and if the PID is in the non-selected path of an Override Selector block, then the CV value of IncrSummer is set equal to that of the override feedback value it receives from its secondary, that is, the PID block.

Windup processing

Every regulatory control type block maintains anti-reset windup status for its output (ARWOP) and each of its initializable inputs (ARWNET). The following table lists the posssible values for ARWOP and ARWNET parameters.

If the Value is . . .	Then, the Associated Parameter . . .
Normal	is free to move in either direction.
Hi	is at its high limit and it may only be lowered.
Lo	is at its low limit and it may only be raised.
HiLo	may not move in either direction.

Manual Mode Interaction

When the MODE of a regulatory control block is changed to Manual (Man), the block sets its windup status (ARWNET) to HiLo. This means that every block upstream in a cascade strategy will set its windup status (ARWNET and ARWOP) to HiLo.

ARWOP computation

The ARWOP indicates if the output (OP) can be raised or lowered. The PID-type function blocks use ARWOP to restrict integral control. When ARWOP contains a value other than **Normal**, the PID block stops integral control in the windup direction. Integral control continues in the other direction, as does proportional and derivative control. But, windup status has no impact on proportional and derivative control.

If a function block has a secondary, it fetches the secondary's windup status and recomputes its ARWOP. The conditions within the function block, such as output being at its high limit, also affect the ARWOP. The ARWOP is computed as follows, assuming the block has only one output or that it is not a FANOUT block.

If Any of the Following are True . . .	Then, ARWOP Equals . . .
This block is inactive.	**HiLo**
A secondary exists but this block cannot fetch secondary data from it (communications or configuration error).	
A secondary exists and its windup state equals HiLo	
This block is in initialization (INITMAN = On).	
A secondary exists and it is requesting this block to initialize.	
A secondary exists and its windup state equals Hi.	**Hi**
This block's output is at its high limit (OPHIFL = On).	
A secondary exists and its windup state equals Lo.	**Lo**
This block's output is at its low limit (OPLOFL = On).	

ARWNET computation

When ARWNET is HiLo, stores to SP are not limited, rather this is the status propagated to the primary. The only *limiting* anti-reset windup status ever does is to stop integral action in one or both directions on PID blocks. For any other regulatory control type

block, ARWNET is not used for any kind of limiting. The ARWNET is computed as follows, assuming the block has only one output or that it is not a FANOUT block.

If Any of the Following are True . . .	Then, ARWNET Equals . . .
This block is inactive.	**HiLo**
The ARWOP equals HiLo.	
This block is in Manual mode (MODE = Man)	
The calculated value (CV) range (CVEUHI / CVEULO) is NaN.	
The CV is NaN	
This block is connected to a non-initializable primary	
The ARWOP equals Hi (Pid function blocks have a configurable Control Action option (CTLACTN). If CTLACTN = Reverse , ARWNET will track ARWOP; but if CTLACTN = Direct , ARWNET will be the opposite of ARWOP.)	**Hi**
The input from the primary is at a high limit. For example, SPHIFL = On.	
This block's output has reached its positive rate-of-change limit (OPROCPOSFL = On)	
The ARWOP equals Lo (Pid function blocks have a configurable Control Action option (CTLACTN). If CTLACTN = Reverse , ARWNET will track ARWOP; but if CTLACTN = Direct , ARWNET will be the opposite of ARWOP.)	**LO**
Input from the primary is at a low limit. For example, SPLO.FL = On.	

If Any of the Following are True . . .	Then, ARWNET Equals . . .
This block's output has reached its negative rate-of-change limit (OPROCNEGFL = On)	

INCRSUMMER parameters

REFERENCE - INTERNAL

Refer to *Control Builder Components Reference* for a complete list of the parameters used with the INCRSUMMER block.

OVRDSEL (Override Selector) Block

Description

The OVRDSEL block accepts up to four inputs (primaries) and selects the one with the highest or lowest value. It looks like this graphically:

Each OVRDSEL block supports the following user configurable attributes. The following table lists the given name of the "Tab" in the parameter configuration form and then briefly describes the attributes associated with that Tab. This data is only provided as a quick document reference, since this same information is included in the on-line context sensitive Help.

Configuration Tab	Description
Main	• Name – Block (Tag) name of up to 16 characters long. Must be unique within the CM block containing it. • Description (DESC) – Block descriptor of up to 24 characters long. • Engineering Units (EUDESC) – Lets you specify a text string of up to 16 characters to identify the variable values associated with this block. For example, you could specify DEGF for temperature values in degrees Fahrenheit. This name is used on any associated displays and generated reports. • Execution Order in CM (ORDERINCM) – Specifies the execution order of the block in the CM relative to other blocks contained in this CM. Enter as a number between 1 to 32767. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. • Control Equation (CTLEQN) – Lets you select Equation A or B to define if the block is to select the highest or the lowest non-bypassed inputs. The default selection is EQA, which means the block selects the highest non-bypassed inputs. • Enable Override Option (OROPT) – Lets you specify if the override option is to be enabled or not. This determines if the block propagates override feedback data to the non-selected inputs or not. The default selection is disabled (unchecked or OFF), which means the feedback data is not propagated. • Enable Override Offset (OROFFSET) – Lets you specify if an upstream PID block should apply a calculated offset to the propagated feedback value or not. This only applies when the OROPT is enabled (checked or ON). The default selection is disabled (unchecked or OFF), which means the PID block does not apply an offset to the feedback value. • Enable Secondary Initialization Option (SECINITOPT) – Lets you specify if the block is to ignore initialization and override requests from the secondary or not. The default selection is enabled (checked, do not ignore).

Configuration Tab	Description
	• Normal Mode (NORMMODE) – Lets you specify the MODE the block is to assume when the Control to Normal function is initiated through the Station display. Selections are MANual, AUTOmatic, CAScade, BackupCAScade and NONE. All selections are not valid for a given block. The default selection is NONE. • Normal Mode Attribute (NORMMODEATTR) – Lets you specify the mode attribute (MODEATTR) the block is to assume when the Control to Normal function is initiated through the Station display. Selections are NONE, OPERATOR and PROGRAM. The default selection is NONE. • Mode (MODE) – Lets you set the block's current MODE. The selections are MANual, AUTOmatic, CAScade, BackupCAScade, NONE, and NORMAL. All selections are not valid for a given block. The default selection is MANual. MODE identifies who may store values to the block's initializable inputs or output. Blocks strictly enforce the MODE assignment. Some function blocks perform automatic mode switching (or mode shedding), while others require manual intervention. The block's MODE is derived at "runtime" based on current conditions. MODE processing checks for the following conditions, and changes the block's MODE as appropriate. – External request for MODE switching. – Safety interlock request. • Mode Attribute (MODEATTR) – Lets you set the block's mode attribute. The selections are NONE, OPERATOR, PROGRAM, and NORMAL. The default selection is OPERATOR. MODEATTR identifies who may store values to the output (OP), when the block's MODE is MANual. • Permit Operator Mode Changes (MODEPERM) – Lets you specify if operators are permitted to make MODE changes or not. The default is Enabled (checked). A store to MODE does not change the NORMMODE. • Permit External Mode Switching (ESWPERM) – Lets you specify if external MODE switching through user

Configuration Tab	Description
	configured interlocks is permitted or not, if you have at least an Engineering access level. The default is Disabled (unchecked). • Enable External Mode Switching (ESWENB) – Lets you specify if external MODE switching through user configured interlocks is enabled or not, if ESWPERM is checked (Permitted). The default is Disabled (unchecked). • Safety Interlock Option (SIOPT) – Lets you specify MODE and OP block is to assume upon a safety interlock alarm. The selections are NO_SHED, SHEDHOLD, SHEDLOW, SHEDHIGH, and SHEDSAFE. The default selection is SHEDHOLD. • Bad Control Option (BADCTLOPT) – Lets you specify MODE and OP block is to assume if CV goes BAD. The selections are NO_SHED, SHEDHOLD, SHEDLOW, SHEDHIGH, and SHEDSAFE. The default selection is NOSHED.
Input	• High Limit (XEUHI) – Lets you specify the high input range limit that represents 100% full scale input for all the block inputs (X[1..4]). The default value is 100. • Low Limit (XEULO) – Lets you specify the low input range limit that represents the 0 full scale input for all the block inputs (X[1..4]). The default value is 0 (zero). • Enable Input Bypassing (ORBYPPERM) – Lets you specify whether or not an operator can explicitly bypass (ignore) any input to the block. The default selection is disabled (unchecked or OFF), which means an operator can not bypass any input. • Timeout Time (TMOUTTIME) – Lets you specify a time in seconds that must expire before the block assumes that its input update has timed out. The block must be in CAScade mode for it to monitor its primary input for timeout. The default setting is 0, which means the timeout function is disabled. If the input is from a connection in another controller in a peer-to-peer architecture, the actual timeout time equals the configured TMOUTTIME plus the CDA timeout time. The CDA timeout time equals four times the configured CEE subscription rate. For example, if

Configuration Tab	Description
	the CEE subscription rate is 100 milliseconds and the TMOUTTIME is 5 seconds, the actual timeout time for the block is 4 times 100ms plus 5s or 5.4 seconds. • Description – Lets you enter up to a 15-character description for each input (X[1..4]). The description is stored in the XDESC[1..4] parameter and is copied to the SELXDESC parameter when the corresponding input is selected. This means SELXDESC is automatically updated whenever SELXINP is updated. • Bad Input Option (BADINPTOPT[1..4]) – Lets you specify whether the block is to include or ignore an input with bad values in its selection process. The default selection is INCLUDEBAD, which means the block's CV value is set to NaN (Not a Number). • Bypass (ORBYPASSFL[1..4]) – Lets you specify whether a given input is to be bypassed or not. If a given input flag is ON (checked), this input is not used in the block's selection process. The default selection is OFF (unchecked), which means the input is not bypassed. If all inputs are bypassed, the block holds its CV at its last value.
Output	• High Limit (%) (OPHILM) – Lets you specify the output high limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you enter a High Limit of 90%, the high limit in engineering units is 90% times 450 or 405 + 50 (CVEULO) equals 455. This check is not applied for a function block that is in the MANual mode. The default value is 105%. • Low Limit (%) (OPLOLM) – Lets you specify the output low limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you enter a Low Limit of 10%, the low limit in engineering units is 10% times 450 or 45 + 50 (CVEULO) equals 95. This check is not applied for a function block that is in the MANual mode. The default value is -5%. • Extended High Limit (%) (OPEXHILM) – Lets you specify the output extended high limit as a percent of

Configuration Tab	Description
	the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you use the default value of 106.9%, the extended high limit in engineering units is 106.9% times 450 or 481.05 + 50 (CVEULO) equals 531.05. The default value is 106.9%. • Extended Low Limit (%) (OPEXLOLM) – Lets you specify the output extended low limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you use the default value of -6.9%, the extended low limit in engineering units is -6.9% times 450 or -31.05 + 50 (CVEULO) equals 18.95. The default value is -6.9%. • Rate of Change Limit (%) (OPROCLM) – Lets you specify a maximum output rate of change limit for both the positive and negative directions of the output in percent per minute. This lets you prevent an excessive rate of change in the output so you can match the slew rate of the control element to the control dynamics. We recommend that you configure this value before you tune the loop, so tuning can accommodate any slow down in response time caused by this rate limiting. This check is not applied for a function block that is in the MANual mode. The default value is Not a Number (NaN), which means no rate limiting is applied. • Minimum Change (%) (OPMINCHG) – Lets you specify an output minimum change limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). This lets you define how much the OP must change before the function block outputs a new value. It filters out changes that are too small for the final control element to respond to. This check is not applied for a function block that is in the MANual mode. The default value is 0, which means no change limiting is applied. • Safe OP (%) (SAFEOP) – Lets you specify the safe output value as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 0 to 500 and you enter a Safe OP of 50%, the safe output value in engineering units is 50% times 500 or 250. The default value is Not a Number (NaN), which means the OP is held at its last good value.

Configuration Tab	Description
	• Output Bias (OPBIAS.FIX) – Lets you specify a fixed bias value in engineering units that is added to the Calculated Variable (CV) output value. See the Output Bias section for this function block for details. The default value is 0, which means no value is added. • Output Bias Rate (OPBIAS.RATE) – Lets you specify an output floating bias ramp rate in engineering units per minute. This bias rate is only applied when the floating bias in non-zero. See the Output Bias section for this function block for details. The default value is Not a Number (NaN), which means no floating bias is calculated. As a result, if the primary does not accept the block's initialization value, a bump in OP occurs.
Alarms	• Type – Identifies the types of alarms this block supports. Of course, these alarms also interact with other block configuration options such as the Safety Interlock Option (SIOPT) and Bad Control Option (BADCTLOPT). The types are: – Safety Interlock (SIALM.FL) – Bad Control (BADCTLALM.FL) – OP High (OPHIALM.FL) – OP Low (OPLOALM.FL) • Enable Alarm (SIALM.OPT) – Lets you enable or disable Safety Interlock alarm type. A check in the box means the alarm is enabled. The default selection is a checked box or enabled (Yes). You can also configure the SIALM.OPT parameter as a block pin, a configuration and/or a monitoring parameter so it appears on the block in the Project and Monitoring tree view, respectively. • Trip Point – Lets you specify the OP High Alarm (OPHIALM.TP) and OP Low Alarm (OPLOALM.TP) trip points in percent. The default value is NaN, which disables the trip point. • Priority – Lets you set the desired priority level individually for each alarm type (SIALM.PR, BADCTLALM.PR, OPHIALM.PR, OPLOALM.PR). The default value is LOW. The levels are:

Configuration Tab	Description
	– NONE - Alarm is neither reported nor annunciated. – JOURNAL - Alarm is logged but it does not appear on the Alarm Summary display. – LOW, HIGH, URGENT - Alarm is annunciated and appears on the Alarm Summary display. • Severity – Lets you assign a relative severity individually for each alarm type (SIALM.SV, BADCTLALM.SV, OPHIALM.SV, OPLOALM.SV) as a number between 0 to 15, with 15 being the most severe. This determines the alarm processing order relative to other alarms. The default value is 0. • Deadband Value (ALMDB) – Lets you specify a deadband value that applies to all analog alarms to prevent nuisance alarms due to noise at values near the trip point. The default value is 1. Note that this value is loaded to the individual alarm parameters (for example, OPHIALM.DB and OPLOALM.DB) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB. • Filter Time (ALMTM) – Lets you specify a time in seconds to define how long an analog alarm must exist before it is set true. The default value is 0, which means the alarm is set true as soon as the value exceeds the deadband value. Note that this value is loaded to the individual alarm parameters (for example, OPHIALM.TM and OPLOALM.TM) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB. • Deadband Units (ALMDBU) – Lets you specify if the deadband value represents percent or engineering units. The default value is percent. Note that this value is loaded to the individual alarm parameters (for example, OPHIALM.DBU and OPLOALM.DBU) when the CM is loaded. If you

Configuration Tab	Description
	configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB.
SCM	– SCM Mode Tracking Option (MODETRACK) – Lets you select the desired Mode Tracking function for the SCM associated with this block's Control Module. It defines how the FB will set the state of the MODEATTR based upon the MODE of the SCM. See the *Sequential Control Module User's Guide* for more information on this function. The default selection is ONESHOT. The selections are: – None – ONESHOT – SEMICONT – CONTRTN – CONT • Option Type – Lets you specify the action the function block is to take when the SCM goes into an abnormal state. The Starting State Option (STARTOPT) applies when the SCM state is Checking, Idle, or Complete. The Stop/Abort State Option (STOPOPT) applies when the SCM state is Stopping or Stopped, Aborting or Aborted. The Hold State Option (HOLDOPT) applies when the SCM state is Holding or Hold. The Restart State Option (RESTARTOPT) applies when the SCM state is Resume or Run. The NONE and LASTREQ are the only selections for the Restart State Option. You can select from one of these types for the other options as applicable for the given regulatory control function block: – NONE - No changes. – MAN - Set MODEREQ = MANUAL. – AUTO - Set MODEREQ = AUTOMATIC (Not applicable for this block). – CAS - Set MODEREQ = CASCADE.

Configuration Tab	Description
	– FIXEDOP - Set OPREQ = Configured Value. – HOLDPV - Set SPREQ = PV (Not applicable for this block). – FIXED SP - Set SPREQ = Configured Value and SPRATEREQ = NaN (Not applicable for this block). – RAMPEDSP - Set SPTVREQ = Configured Value and SPRATEREQ = Configured Rate (Not applicable for this block). • Value (STARTVAL, STOPVAL, HOLDVAL) – Depending upon Option Type selection, lets you specify an output or set point value within the respective range. For output, within OPLOLM to OPHILM and within SPLOLM to SPHILM, for set point. The default value is NaN (Not a Number).
Block Pins	Lets you select the available parameters that you want to expose as input/output pins on the function block graphic in Control Builder.
Configuration Parameters	Lets you select the available parameters that you want to appear on the face of the function block in the Project tab in Control Builder.
Monitoring Parameters	Lets you select the available parameters that you want to appear on the face of the function block in the Monitoring tab in Control Builder.
Block Preferences	Lets you change several block-viewing preferences including the color of the block's faceplate.
Insertion Type	Lets you include an insertion type from a CAB instances in the block. See *CAB insertion configuration considerations for regulatory control blocks* for more information

Function

This block always forces the unselected inputs to track the selected input by enabling the override feedback option. You select the override option by setting the parameter OROPT to ON or by selecting the Enable Override Option check box on the block's parameter configuration form.

If OROPT is . . .	Then, . . .
ON	This block propagates override feedback data to the unselected inputs. And, if the inputs come from cascade strategies, this block provides override feedback data to every upstream block in every unselected path.
OFF	The feedback value is not propagated but behaves as a simple high-low selector. However, the unselected primaries are kept from winding up by propagating a windup status opposite to the override equation. This windup propagation prevents the unselected primaries of the selector from winding up in the direction opposite to the selector equation.

The windup propagation is done whether OROPT is ON or OFF.

The override feedback method of tracking is different from tracking by initialization in the following way:

- With initialization, the upstream blocks set their output based solely on the initialization value from their secondary. They do not fetch inputs or perform their normal algorithm calculations.
- Override feedback is a two-step process.

Step	Action
1	The override feedback data is propagated to all of the blocks in an upstream cascade and they set their outputs accordingly. The data consists of an override status (FBORSTS, which indicates if the primary is selected or not), an override feedback value (internal, which is calculated to prevent "wind-up" in unselected primaries), and an override offset flag (internal), which indicates how the feedback value should be calculated.
2	The cascade executes as normal, where each block fetches its input and performs its normal algorithm calculation.

As previously stated, this block "provides" override feedback data to every block in an upstream cascade. It doesn't matter how many blocks are upstream, or whether they are on-node or off. However, the keyword here is "provides" because it may take several execution cycles for the data to reach the furthest block. The OVRDSEL block will propagate the data to a limited number of on-node blocks. (See limitations below.) When it reaches that limit, it will interrupt the propagation and pass the data to the next upstream block through BACKCALC. When the upstream block fetches BACKCALC, it detects that override propagation was interrupted, and resumes propagating (subject to the same limitations).

Limitations:

- For a given input path, propagation stops at a block that is inactive.
- For a given path, propagation is interrupted at a block with an off-node primary. The primary resumes propagation on its next execution cycle.
- For a given path, propagation is interrupted after five upstream blocks. The sixth block resumes propagation on its next execution cycle.

Example: Assume an OVRDSEL block has four inputs, where one input is a cascade of nine upstream blocks, and each of the others is a cascade of four upstream blocks. Also, assume that all of the blocks are on-node. Then, the OVRDSEL block will propagate to the first five blocks in the first cascade, and to every block in the other cascades. The next time the sixth block runs, it will bring BACKCALC from the fifth, determine that propagation was interrupted, and resume propagation to the remaining blocks in that cascade.

ATTENTION

The system's ability to interrupt and resume override propagation has advantages and disadvantages.

- The advantages are, there are no limitations on the number of blocks in an override strategy or where the blocks reside.
- The disadvantage is, if propagation is interrupted, the blocks above the interrupt point will be using override data that is older than the blocks below it. Override data above this point will typically lag by one or two cycles.

If you have an override strategy where all blocks must have their override data in sync, then that strategy must be on the same node, and have no more than five blocks in each input cascade.

This block provides a bypass flag for each input, which allows the operator, another function block, or a user program to exclude any input from being selected. Inputs may be bypassed regardless of whether OROPT is On or Off.

This block provides bumpless switching by applying a floating bias to the output, regardless of whether OROPT is On or Off.

Configuration example

The following figure and its companion callout description table show a sample configuration that uses an OVRDSEL block to provide override feedback data to upstream PID blocks for quick reference.

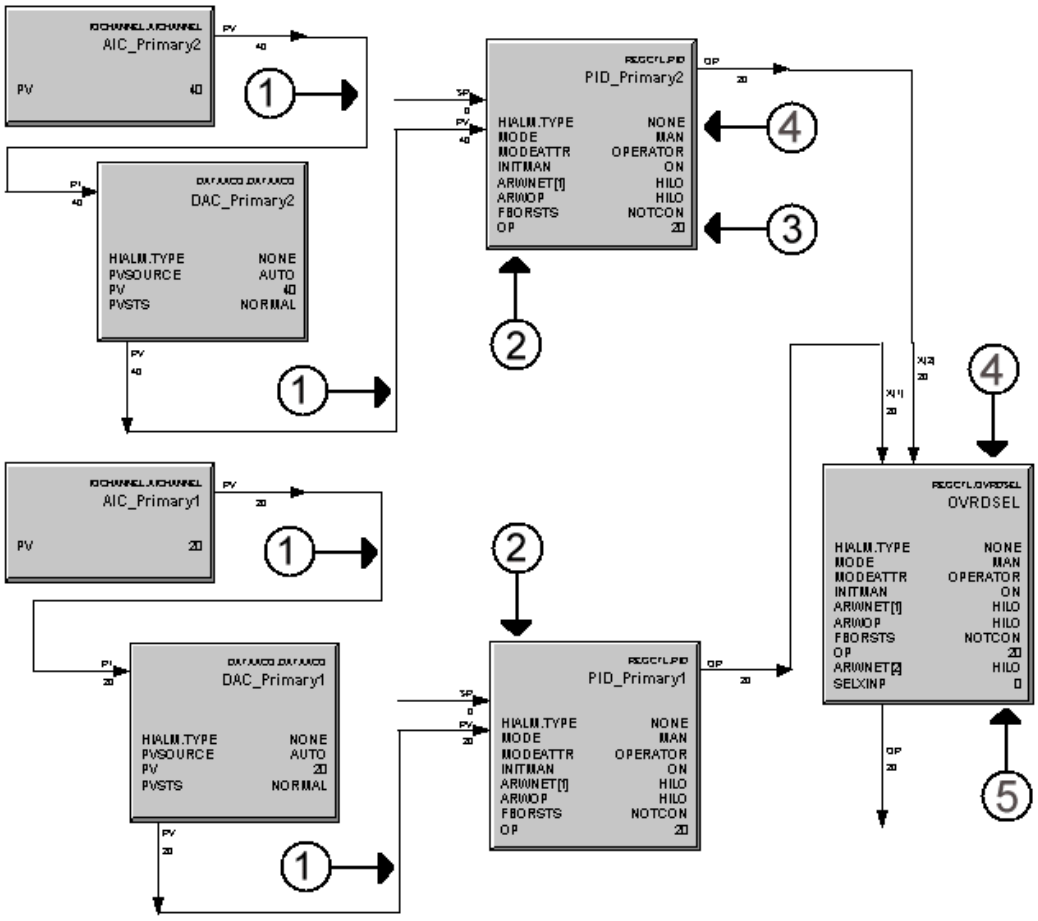


Figure 19 Example of CB configuration using OVRDSEL block.

The following table includes descriptions of the callouts in figure above.

Callout	Description
1	Use the PV parameter connection to carry data from the analog input to the PID block. The default PV connection is exposed, but the implicit/hidden connection function automatically makes a connection to a value/status parameter (PVVALSTS) when it is required.
2	When monitoring Control Module, the FBORSTS parameter shows whether the PID block is selected or not. You must configure the FBORSTS parameter to appear on the faceplate of the block through the Monitoring Parameters tab in the block configuration form.
3	The Enable Override Option (OROPT) is selected for the OVRDSEL block. This means that the Not Selected primary PID's output is initialized to the same value as the Selected PID's output. You must configure the OP parameter to appear on the faceplate of the block through the Monitoring Parameters tab in the block configuration form.

Callout	Description
4	Use the BACKCALCIN/BACKCALCOUT connection to carry secondary data from the OVRDSEL block to the primary PID block. The individual BACKCALCIN/BACKCALCOUT connections for each output used are automatically built by Control Builder as implicit/hidden connections. The secondary data includes this information. • Anti-Reset Windup Status: Indicates if the secondary's initializable input (which is this block's output) is at its high or low limit. • Initialization Request Flag: Used to request continuous initialization. If the flag is set (and this block is configured to accept secondary initialization), this block initializes itself for one cycle, and resumes normal processing on the next. • Oneshot Initialization Flag: Used to request oneshot initialization. If the flag is set (and this block is configured to accept secondary initialization), this block initializes itself for one cycle, and resumes normal processing on the next. • Initialization Value: Used for continuous and oneshot initialization. • Override Status: If a block is in an override strategy, this flag indicates whether it is the selected strategy or not. If the block is in an unselected strategy (and configured to accept secondary initialization), it invokes its override feedback processing. • Override Feedback Value: Similar to initialization value; this is calculated to prevent "wind-up" in unselected primaries. • Override Offset: Only applies to PID type function blocks. If a PID is in an unselected override strategy, this flag indicates how it should calculate its output. • Engineering Units: The engineering units (EU) of the secondary's initializable input. For example, If the secondary's input is SP, it sends SPEUHI and SPEULO to the primary. The primary then sets its CV range (CVEUHI and CVEULO) to this.
5	You can configure the OVRDSEL block to select the lower of the two primary inputs by selecting Equation B or the higher of two inputs by selecting Equation A.

Configuration considerations

Keep the following considerations in mind when configuring control strategies using OVRDSEL blocks.

- When possible, load control strategies using OVRDSEL blocks in the same CEE. Only the most downstream OVRDSEL block in the cascade propagates the override feedback value to its primaries. When this strategy is in the same CEE, the propagation of override feedback value to the unselected primaries of an OVRDSEL block takes place in one execution cycle of the block. The means the override feedback value and other feedback data are the most recent values.

- In any control strategy that includes OVRDSEL blocks, the sequence of execution of all blocks is very important. All the primaries should run before the OVRDSEL block that propagates the feedback gets a chance to execute. This another reason for loading control strategies that include OVRDSEL blocks in the same CEE. The following configuration scenarios outline some typical execution settings for reference.

  - If all the blocks are contained in the same Control Module, all the primaries should execute before the OVRDSEL block does. This means the ORDERINCM parameter of the OVRDSEL block must be larger than the corresponding number for all its primaries. For example, if Control Module CM01 has blocks PID01, PID02, PID03, PID04, and OVRDSEL05, the suggested settings for the ORDERINCM parameter are PID01.ORDERINCM < PID02.ORDERINCM < PID03.ORDERINCM < PID04.ORDERINCM < OVRDSEL05.ORDERINCM.

  - If primaries are residing in different Control Modules within the same CEE, the previous scenario still applies for the Control Module containing the OVRDSEL block. Plus, the ORDERINCEE parameter setting for the Control Modules that contain other primaries should be smaller than the ORDERINCEE parameter for the Control Module that contains the OVRDSEL block. For example, if Control Module CM01 contains a PID cascade loop with an OVRDSEL block and Control Modules CM02 and CM03 contain other primaries of the OVRDSEL block, the suggested settings for the ORDERINCEE parameter are CM01.ORDERINCEE > CM02.ORDERINCEE > CM03.ORDERINCEE.

  - The strategy includes a cascade loop with an OVRDSEL block that propagates only 5 on-node regulatory control blocks in its one execution cycle. The propagation then continues through the BACKCALC connection , when the primary runs the next time. The override feedback value could be old for any primaries that are off-node or beyond the limit of 5.

Inputs

The OVRDSEL block accepts one to four inputs – X[1] through X[4]. It requires at least **two** inputs, but they can be any of the four.

- X[1] through X[4] are initializable inputs.
- The inputs must be pulled from other function blocks; you cannot store to them.
- This block may have two to four primaries, depending on the number of inputs that are configured. (There is one primary per initializable input.)

Input ranges

XEUHI and XEULO define the full range of inputs.

- XEUHI represents the 100% of full scale value.
- XEULO represents the 0% of full scale value.

This block assumes that all X-inputs are within XEUHI and XEULO. It applies no range checks.

Input descriptors

You can define descriptor (name) of up to 15-characters for each input. The descriptors reside in the XDESC parameter, and when an input is selected, the corresponding descriptor is copied to SELXDESC.

When SELXINP is updated, then SELXDESC is automatically updated.

Initializable outputs

"Initializable output" and "initializable input" are variable attributes, similar to data type or access level. A parameter with the "initializable" attribute has an associated BACKCALC parameter, and when you create a connection between an initializable input and initializable output, you can also create a BACKCALC connection. Control Builder automatically builds the required BACKCALC connections, so you don't have to create them manually. These "implicit" build connections are "hidden" from view and the related parameter pins are not exposed on the control chart.

For example, if you connect OP from a PID block to an OVRDSEL block or an AOCHANNEL block, Control Builder automatically creates the BACKCALCOUT to BACKCALCIN connection.

The OVRDSEL block has the following initializable outputs:

- OP = Calculated output, in percent.

- OPEU = Calculated output, in engineering units.

You may create a connection to OP or OPEU but not to both. Therefore, this block may have only one secondary. If you do not create a connection to OP or OPEU, then the block does not have a secondary. Alternately, if you connect OP or OPEU to a non-initializable input, then this block does not have a secondary. (Note that the default OP connection pin is exposed on the blocks and the implicit/hidden connection function automatically makes the appropriate value/status parameter connection when required. For example, if you connect the output from a PID block (PIDA.OP) to the input of an OVRDSEL block (OVRDSEL1.X1), the implicit/hidden connection is made to PIDA.OPX to provide value/status data.)

Output ranges and limits

- CVEUHI and CVEULO define the full range of CV in engineering units.
- This block doesn't have CVEUHI and CVEULO as configurable parameters for the reasons given below:
  - If this block has a secondary, it brings the secondary's input range through BACKCALCIN and sets its CV range to that.

ATTENTION

This block fetches the secondary's input range regardless of SECINITOPT (i.e.,regardless of whether the secondary's initialization and override data will be used).

- IncrSummer's CVEU range follows its secondary's XEU ranges if a secondary is connected, else it follows its own XEU ranges. So it is the user's responsibility to set the XEU ranges of IncrSummer and that of its secondary the same. If the ranges are different, the results can be *unexpected*.
- OPHILM and OPLOLM define the normal high and low limits for OP, as a percent of the CV range. These are user-specified values.
  - OP will be clamped to these limits if the algorithm's calculated result (CV) exceeds them, or another function block or user program attempts to store an OP value that exceeds them. However, the operator may store an OP value that is outside these limits.
- OPEXHILM and OPEXLOLM define the extended high and low limits for OP, as a percent of the CV range. These are user-specified values.

  - The operator is prevented from storing an OP value that exceeds these limits.

- OPTOL allows the user to configure a tolerance limit for the manually entered OP. If the difference between the new OP value and the current OP value is greater than OPTOL, then confirmation is required from the user to store this new value.

Mode handling

This function block supports the Cascade and Manual modes.

If MODE is . . .	Then, . . .
Cascade	All inputs must be pulled from another function block.
Manual	OP may be stored by the operator or a user program. (All inputs are ignored.)

The initialization request occurs when the MODE changes from CAScade to MANual, but not from MANual to CAScade.

Timeout monitoring

If MODE is CAScade, this block performs timeout monitoring on all inputs (X[1..4]) that are not bypassed. (See Bypass Processing paragraph below.) If an input value is not updated within a predefined time (TMOUTTIME), the block invokes timeout processing.

The maximum time between updates is specified by TMOUTTIME (in seconds)

  - Enable timeout monitoring by setting TMOUTTIME to a non-zero value.
  - Disable timeout monitoring by setting TMOUTTIME to zero.

Timeout processing

This function block only performs timeout monitoring on inputs that are not bypassed. (See Bypass Processing paragraph below.)

If MODE is CAScade and an input times out, this block does the following :

- Sets the “input timeout” flag (TMOUTFL)
- Sets the input value to Bad (NaN).
- Requests the input’s primary to initialize

This block does not support mode shedding on timeout.

ATTENTION

If the input is from a connection in another controller in a peer-to-peer architecture, the actual timeout time equals the configured TMOUTTIME plus the CDA timeout time. The CDA timeout time equals four times the configured CEE subscription rate. For example, if the CEE subscription rate is 100 milliseconds and the TMOUTTIME is 5 seconds, the actual timeout time for the block is 4 times 100ms plus 5s or 5.4 seconds.

Bypass processing

You may explicitly bypass (ignore) any input. The primary will initialize if it is bypassed. The following parameters support this:

- ORBYPASSFL[1..4] - Override Bypass Flags. A flag for each input; used to specify which inputs should be bypassed. If a flag is set, the corresponding input is not used in the selection process. If all bypass flags are set, this block holds CV at its last value. This block uses the bypass flags regardless of whether OROPT is ON or OFF.
- ORBYPPERM- Override Bypass Enable. Indicates if the operator is allowed to bypass inputs.

Bad input option

The block can include or ignore inputs with bad values (NaN) per input, by setting the parameter BADINPTOPT[1..4].

- BADINPTOPT- Bad Input Option enable. Indicates if the function block should include bad inputs (NaN) in the selection process.
- BADINPTOPT has the following options:
  - IgnoreBad (Ignore bad inputs)
  - IncludeBad (Include bad inputs)

When an input goes bad and its BADINPTOPT(i) is set to IncludeBad, then the OVRDSEL's CV is set to NaN.

When an input goes bad and its BADINPTOPT(i) is set to IgnoreBad, then the OVRDSEL ignores that input in its processing and will select one of the other inputs based on its configured equation (High or Low selector).

Equations

The OVRDSEL block selects one of the inputs according to the following user-selected equations:

- Equation A - select the highest of the non-bypassed inputs:

CV = the highest input + OPBIAS.FIX + OPBIAS.FLOAT

- Equation B - select the lowest of the non-bypassed inputs:

CV = the lowest input + OPBIAS.FIX + OPBIAS.FLOAT

This block stores the number of the selected input in parameter SELXINP, and sets or resets the input selection flags SELXFL(1..4). There is one selection flag per input; ON means the input was selected, and Off means it was not.

If multiple inputs have the same value, and that value is selected, this block selects the input with the lowest index. For example, assume X[2] and X[3] have the same value, and that value is selected; then the selected input will be X[2].

Input switching

This block provides bumpless switching by applying a floating bias to the output, regardless of whether OROPT is On or Off.

Output bias

– The function block ramps the OPBIAS.FLOAT to zero by applying the following calculation each time it executes.

OPBIAS.FLOAT = OPBIAS.FLOAT – (OPBIAS.RATE / cycles_per_Min)

Where:

cycles_per_min = number of times the function block executes per minute (calculated)

– NaN:
When the OPBIAS.RATE is Not a Number (NaN), no OPBIAS.FLOAT is calculated. This means a bump in the output will occur, if the primary does not accept this block's initialization value.

Bad CV processing

If the selected input is bad and MODE is Cascade, this block does the following:

- sets CV to Bad (NaN)
- sets the Bad Control flag (BADCTLFL)

When the selected input returns to normal, this block does the following:

- resets the Bad Control flag (BADCTLFL)
- requests the bypassed primaries to initialize (i.e., sets INTREQ(n) to On when ORBYPASSFL(n) is On)

Control initialization

This block brings initialization requests from its secondary through BACKCALC. In addition, the secondary may propagate one-shot initialization requests to this block.

You can use SECINITOPT to ignore initialization requests from the secondary.

If the secondary is requesting initialization, this block:

- initializes its output:

```
CV = initialization value from the secondary
```

- sets SELXINP = None
- builds initialization requests for the primaries:

If this block is in the Manual mode, it requests all primaries to initialize. Otherwise, it only requests the bypassed primaries to initialize. It builds initialization requests as follows:

- INITREQ(n) = On
- INITVAL(n) = CV

where:

(n)	=	identifies the primary to be initialized
INITREQ	=	initialization request flag for primary
INITVAL	=	initialization value for primary

Restart or function block activation

When this function block is activated, or on a warm restart, it does the following:

- Sets CV = initialization value from the secondary, and
- Requests all primaries to initialize (That is set all INTREQ flags On, and set INITVAL = CV).

Override feedback propagation

This block propagates override feedback data to the upstream blocks if it is,

- in the Cascade mode,
- not initializing, and
- OROPT is ON.

If these conditions are true, this block provides the following data to every upstream block:

- Override status — This status indicates if the upstream block is on the selected input path or not. Possible values are:

Value	Sent to
SEL (Selected)	all blocks on the selected input path.
NotCon (Not Connected)	all blocks on bypassed input paths.
NotSel (Not Selected)	blocks on unselected, non-bypassed paths.

- Override feedback value: The OVRDSEL block sends its current CV to each of its primaries.
  - The CV is clamped to OPHILM if it is greater than OPHILM and to OPLOLM if it is less than OPLOLM.

- Override offset flag: This flag only applies to upstream PIDs; it indicates if the PID should apply a calculated offset to the override feedback value.
  - If the offset flag is Off, the PID doesn't apply an offset; it initializes its CV as follows:
    CV = override feedback value
  - If the offset flag is On, the PID applies an offset; it initializes its CV as follows:
    CV = (override feedback value) + Gain * (PVP - SPP) for direct control action.
    CV = (override feedback value) - Gain * (PVP - SPP) for reverse control action.
  - Additionally, the Gain *(PVP – SPP) term is set to 0.0; If:
  - à Gain * (PVP – SPP) > 0.0 and the downstream OVRDSEL block is a High selector. Or,
  - è Gain * (PVP – SPP) < 0.0 and the downstream OVRDSEL block is a Low selector.

Recommendations on configuring override strategies

- While PIDs in an override strategy can be configured with proportional and derivative action, use of these actions should be carefully considered because undesired results may occur, such as momentary oscillations caused by "kicks" in the error.
- We do not recommend using a PID with External Reset Feedback in an override strategy.
- If override propagation is interrupted, the blocks above the interrupt point are using override data that is older than the blocks below it. Override data above this point typically lags by one or two function block cycles.
  If you have an override strategy where all blocks must have their override data in sync, then that strategy must be on the same node, and have no more than seven blocks in each input cascade.

Windup processing

Every regulatory control type block maintains anti-reset windup status for its output (ARWOP) and each of its initializable inputs (ARWNET). The following table lists the posssible values for ARWOP and ARWNET parameters.

If the Value is . . .	Then, the Associated Parameter . . .
Normal	is free to move in either direction.
Hi	is at its high limit and it may only be lowered.
Lo	is at its low limit and it may only be raised.
HiLo	may not move in either direction.

Manual Mode Interaction

When the MODE of a regulatory control block is changed to Manual (Man), the block sets its windup status (ARWNET) to HiLo. This means that every block upstream in a cascade strategy will set its windup status (ARWNET and ARWOP) to HiLo.

ARWOP computation

The ARWOP indicates if the output (OP) can be raised or lowered. The PID-type function blocks use ARWOP to restrict integral control. When ARWOP contains a value other than **Normal**, the PID block stops integral control in the windup direction. Integral control continues in the other direction, as does proportional and derivative control. But, windup status has no impact on proportional and derivative control.

If a function block has a secondary, it fetches the secondary's windup status and recomputes its ARWOP. The conditions within the function block, such as output being at its high limit, also affect the ARWOP. The ARWOP is computed as follows, assuming the block has only one output or that it is not a FANOUT block.

If Any of the Following are True . . .	Then, ARWOP Equals . . .
This block is inactive.	**HiLo**
A secondary exists but this block cannot fetch secondary data from it (communications or configuration error).	
A secondary exists and its windup state equals HiLo	
This block is in initialization (INITMAN = On).	
A secondary exists and it is requesting this block to initialize.	

If Any of the Following are True . . .	Then, ARWOP Equals . . .
A secondary exists and its windup state equals Hi.	**Hi**
This block's output is at its high limit (OPHIFL = On).	
A secondary exists and its windup state equals Lo.	**Lo**
This block's output is at its low limit (OPLOFL = On).	

ARWNET computation

When ARWNET is HiLo, stores to SP are not limited, rather this is the status propagated to the primary. The only *limiting* anti-reset windup status ever does is to stop integral action in one or both directions on PID blocks. For any other regulatory control type block, ARWNET is not used for any kind of limiting. The ARWNET is computed as follows, assuming the block has only one output or that it is not a FANOUT block.

If Any of the Following are True . . .	Then, ARWNET Equals . . .
This block is inactive.	**HiLo**
The ARWOP equals HiLo.	
This block is in Manual mode (MODE = Man)	
The calculated value (CV) range (CVEUHI / CVEULO) is NaN.	
The CV is NaN	
This block is connected to a non-initializable primary	
The ARWOP equals Hi (Pid function blocks have a configurable Control Action option (CTLACTN). If CTLACTN = Reverse , ARWNET will track ARWOP; but if CTLACTN = Direct , ARWNET will be the opposite of ARWOP.)	**Hi**

If Any of the Following are True . . .	Then, ARWNET Equals . . .
The input from the primary is at a high limit. For example, SPHIFL = On.	
This block's output has reached its positive rate-of-change limit (OPROCPOSFL = On)	
The ARWOP equals Lo (Pid function blocks have a configurable Control Action option (CTLACTN). If CTLACTN = Reverse , ARWNET will track ARWOP; but if CTLACTN = Direct , ARWNET will be the opposite of ARWOP.)	**LO**
Input from the primary is at a low limit. For example, SPLO.FL = On.	
This block's output has reached its negative rate-of-change limit (OPROCNEGFL = On)	

Anti-Reset Windup Status

The anti-reset windup network in (ARWNETIN) and anti-reset windup output in (ARWOPIN) parameters are added in the standard anti-reset windup (ARW) computation logic. They are user configurable and allow stores from Sequential Control Modules (SCMs) and control algorithm block (CAB) programs.

The ARWNETIN and ARWOPIN parameters would be *ORed* into the existing standard logic so it is not lost. The following table summarizes the influence the ARWNETIN and ARWOPIN parameters have on the ARWNET and ARWOP parameters, which are not user configurable.

ARWNETIN or ARWOPIN Parameter Is. . .	Standard Computation Logic Is . . .	ARWNET or ARWOP Parameter Is . . .
NORMAL	NORMAL	NORMAL
NORMAL	HI	HI
NORMAL	LO	LO
NORMAL	HILO	HILO

ARWNETIN or ARWOPIN Parameter Is. . .	Standard Computation Logic Is . . .	ARWNET or ARWOP Parameter Is . . .
HI	NORMAL	HI
HI	HI	HI
HI	LO	HILO
HI	HILO	HILO
LO	NORMAL	LO
LO	HI	HILO
LO	LO	LO
LO	HILO	HILO
HILO	NORMAL	HILO
HILO	HI	HILO
HILO	LO	HILO
HILO	HILO	HILO

OVRDSEL parameters

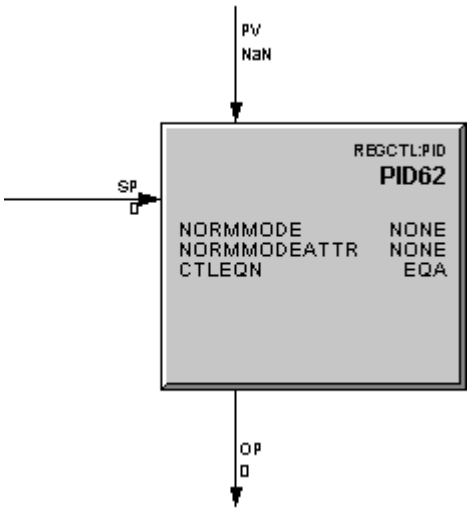
REFERENCE - INTERNAL

Refer to the *Control Builder Components Reference</HH>* for a complete list of the parameters used with the OVRDSEL block.

PID Block

Description

The PID block is a regulatory control block that operates as a proportional-integral-derivative (PID) controller. It supports the Ideal form of calculating the PID terms. The Ideal form is often called the digital-computer version of the PID controller. The PID block looks like this graphically:



The PID block has two analog inputs – a process variable (PV) and a set point (SP). The difference between PV and SP is the error and this block calculates a control output (OP) that should drive the error to zero.

The following equations are supported:

- Proportional, Integral, and Derivative (PID) on the error
- Proportional and Integral (PI) on the error and Derivative (D) on changes in PV
- Integral (I) on the error and Proportional and Derivative (PD) on changes in PV
- Integral (I) only
- Proportional (P) only

The PID block may be used in a single control loop or with multiple PIDs in a cascade strategy. The following figure shows two PID controllers being used for simple cascade control where the output of a temperature controller is used as the set point of a flow controller.

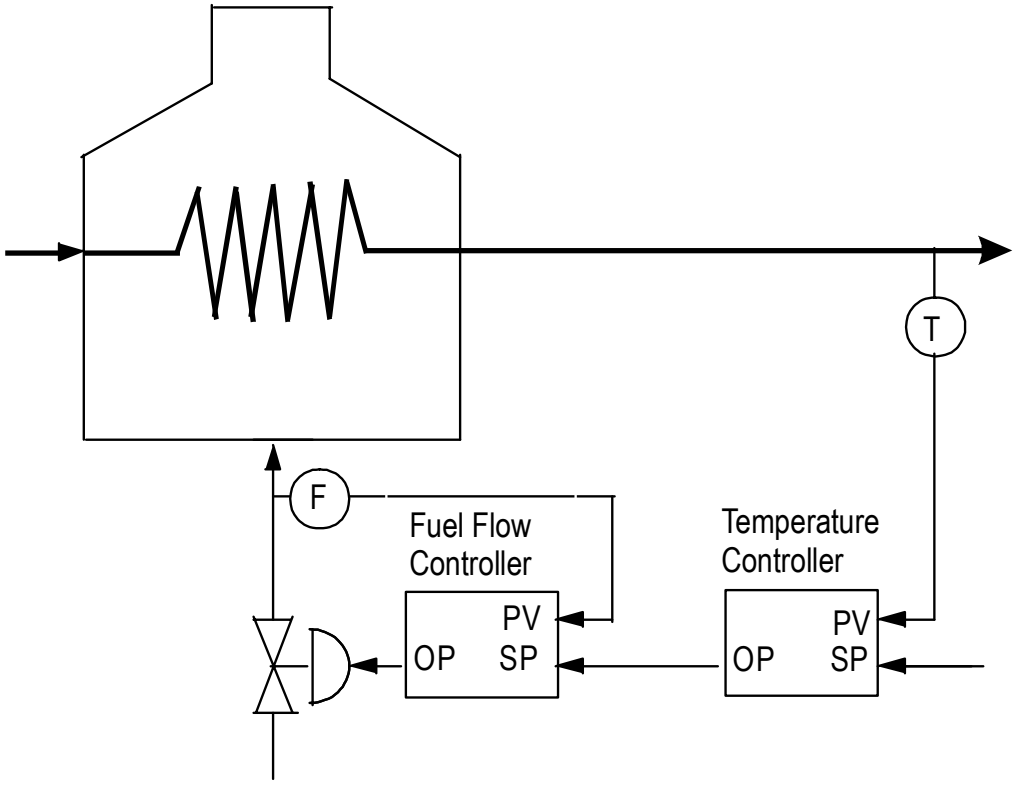


Figure 20 Simple cascade control loop example.

Each PID block supports the following user configurable attributes. The following table lists the given name of the "Tab" in the parameter configuration form and then briefly describes the attributes associated with that Tab. This data is only provided as a quick document reference, since this same information is included in the on-line context sensitive Help.

Configuration Tab	Description
Main	• Name – Block (Tag) name of up to 16 characters long. Must be unique within the CM block containing it. • Description (DESC) – Block descriptor of up to 24 characters long. • Engineering Units (EUDESC) – Lets you specify a text string of up to 16 characters to identify the variable values associated with this block. For example, you could specify DEGF for temperature values in degrees Fahrenheit. This name is used on any associated displays and generated reports. • Execution Order in CM (ORDERINCM) – Specifies the execution order of the block in the CM relative to other blocks contained in this CM. Enter as a number between 1 to 32767. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. • PVEU Range High (PVEUHI) – Lets you specify the high input range value in engineering units that represents 100% full scale PV input for the block. The default value is 100. • PVEU Range Low (PVEULO) – Lets you specify the low input range value in engineering units that represents the 0 full scale PV input for the block. The default value is 0 (zero). • Manual PV Option (PVMANOPT) – Lets you specify the mode and output the block is to assume when PVSTS changes to MANual. The selections are: – NOSHED - No changes. – SHEDHOLD - Sets MODE to MANual and MODEATTR to Operator, disables external mode switching (ESWPERM), and holds output at the last good value.

Configuration Tab	Description
	– SHEDLOW - Sets MODE to MANual and MODEATTR to Operator, disables external mode switching (ESWPERM), and sets output to its extended low limit (OPEXLOLM) value. – SHEDHIGH - Sets MODE to MANual and MODEATTR to Operator, disables external mode switching (ESWPERM), and sets output to its extended high limit (OPEXHILM) value. – SHEDSAFE - Sets MODE to MANual and MODEATTR to Operator, disables external mode switching (ESWPERM), and sets output to the configured safe output (SAFEOP) value. • The default selection is SHEDHOLD. • The block requests its primary to initialize after a mode shed or lets the primary know that it is woundup, if it does shed its mode. An operator can change the block's mode after it is shed, but, the operator must first set PVMANOPT to NOSHED, so the mode doesn't shed again. When PVSTS returns to normal, the block clears its primary initialization request but remains in MANual mode after a mode shed. An operator must return the block to its normal mode. If mode was not shed, the block clears its windup condition and does a oneshot initialization. It also requests the primary to do a oneshot initialization. • Normal Mode (NORMMODE) – Lets you specify the MODE the block is to assume when the Control to Normal function is initiated through the Station display. Selections are MANual, AUTOmatic, CAScade, BackupCAScade and NONE. All selections are not valid for a given block. The default selection is NONE. • Normal Mode Attribute (NORMMODEATTR) – Lets you specify the mode attribute (MODEATTR) the block is to assume when the Control to Normal function is initiated through the Station display. Selections are NONE, OPERATOR and PROGRAM. The default selection is NONE. • Mode (MODE) – Lets you set the block's current MODE. The selections are MANual, AUTOmatic,

Configuration Tab	Description
	CAScade, BackupCAScade, NONE, and NORMAL. All selections are not valid for a given block. The default selection is MANual. MODE identifies who may store values to the block's initializable inputs or output. Blocks strictly enforce the MODE assignment. For example, if the MODE is CAScade and the block is getting its SP from another block, an operator is prevented from overwriting the SP value. However, If there is a breakdown in the cascade loop, the MODE can be changed so an operator can write a value to the SP. Some function blocks perform automatic mode switching (or mode shedding), while others require manual intervention. The block's MODE is derived at "runtime" based on current conditions. MODE processing checks for the following conditions, and changes the block's MODE as appropriate. – External request for MODE switching. – Safety interlock request. • Mode Attribute (MODEATTR) – Lets you set the block's mode attribute. The selections are NONE, OPERATOR, PROGRAM, and NORMAL. The default selection is OPERATOR. MODEATTR identifies who may store values to the output (OP), when the block's MODE is MANual. • Permit Operator Mode Changes (MODEPERM) – Lets you specify if operators are permitted to make MODE changes or not. The default is Enabled (checked). A store to MODE does not change the NORMMODE. • Permit External Mode Switching (ESWPERM) – Lets you specify if external MODE switching through user configured interlocks is permitted or not, if you have at least an Engineering access level. The default is Disabled (unchecked). • Enable External Mode Switching (ESWENB) – Lets you specify if external MODE switching through user configured interlocks is enabled or not, if ESWPERM is checked (Permitted). The default is Disabled (unchecked). • Enable Secondary Initialization Option (SECINITOPT) – Lets you specify if the block is to ignore initialization

Configuration Tab	Description
	and override requests from the secondary or not. The default selection is Enabled (checked, do not ignore). • Safety Interlock Option (SIOPT) – Lets you specify MODE and OP block is to assume upon a safety interlock alarm. The selections are NO_SHED, SHEDHOLD, SHEDLOW, SHEDHIGH, and SHEDSAFE. The default selection is SHEDHOLD. • Bad Control Option (BADCTLOPT) – Lets you specify MODE and OP block is to assume if CV goes BAD. The selections are NO_SHED, SHEDHOLD, SHEDLOW, SHEDHIGH, and SHEDSAFE. The default selection is NOSHED.
Algorithm	• Control Equation Type – Lets you select the control equation the block is to use. The selections are EQA, EQB, EQC, EQD, and EQE. See the PID Equations section for this block for details. The default selection is EQA. • Control Action – Lets you specify if the block is to provide direct or reverse control action. The default selection is REVERSE, which means output decreases as input increases. • T1 (minutes) (T1) – Lets you set integral time to be used for the integral term in the control equation. • T1 High Limit (minutes) (T1HILM) – Lets you define the high limit value in minutes for the integral time setting. The default limit is 1440. • T1 Low Limit (minutes) (T1LOLM) – Lets you define the low limit value in minutes for the integral time setting. The default limit is 0. • T2 (minutes) (T2) – Lets you set the derivative time to be used for the derivative term in the control equation. • T2 High Limit (minutes) (T2HILM) – Lets you define the high limit value in minutes for the derivative time setting. The default limit is 1440. • T2 Low Limit (minutes) (T2LOLM) – Lets you define the low limit value in minutes for the derivative time setting. The default limit is 0.

Configuration Tab	Description
	• High Gain Limit (GAINHILM) – Lets you set a high limit for the gain (K) value. If this value is exceeded, K is clamped to this limit. The default value is 240. • Low Gain Limit (GAINLOLM) – Lets you set a low limit for the gain (K) value. If K is less than this value, it is clamped to this limit. The default value is 0. • Gain Options (GAINOPT) – Lets you select the type of gain term to be used in the PID equation. The default value is LIN. The selections are: – LIN - The LINear gain option provides a proportional control action that is equal to a constant (K) time the error (PV -SP). This applies to equations A, B, and C. – GAP - The GAP gain option reduces the sensitivity of the control action when the PV is in a user-specified band (gap) around the set point. This applies to equations A, B, and C. – NONLIN - The NONLIN gain option provides a proportional control action that is equal to the square of the error, rather than the error itself. This applies to equations A, B, and C. – EXT - The EXTernal gain option modifies the gain (K) by an input value from either the process, another function block, or a user program. You can use this option to compensate for nonlinear process gain - lets you tune the PID gain independent of the normal operating point of the process. For example, If you are controlling the level of tank whose cross-section is not constant, you can use the EXT option to modify the gain to compensate for the nonlinear rate of level change, which is caused by the changing shape of the tank. This applies to equations A, B, and C. • Overall Gain (K) – Lets you set the overall gain value used to calculate the proportional term in the PID equation. The default value is 1. • Gap High Limit (GAPHILM) – Lets you define the high limit value in PV engineering units to be used when calculating GAP gain.

Configuration Tab	Description
	• Gap Low Limit (GAPLOLM) – Lets you define the low limit value in PV engineering units to be used when calculating GAP gain. • Gap Gain Factor (KMODIFGAP)– Lets you specify the value to be used for calculating overall gain (K) when the PV input is within the user specified band (GAPLOLM - GAPHILM) around the SP. The value range is 0.0 to 1.0. • Linear Gain Factor (KLIN) – Lets you specify the value to be used for calculating the overall gain (K) in association with GAP, NONLIN, or EXT gain option. • Non-Linearity Form (NLFORM) –Lets you specify the non-linearity form (0 or 1) to be used for calculating gain (K) based on the formula shown for Non Linear Gain in the Gain Options section. The default value is 1. • Non-Linear Gain Factor (NLGAIN) –Lets you specify the non-linear gain value to be used for calculating gain (K) based on the formula shown for Non Linear Gain in the Gain Options section. The default value is 0. • External Gain Factor (KMODIFEXT) – Lets you specify an input value from either the process, another function block, or a user program to be used to modify the gain (K) calculation per this formula: K = KLIN * KMODIFEXT See External Gain in the Gain Options section for more details. The default value is 1. • Legacy Gap: An option which allows the user to revert back to older GAP and Nonlinear gain calculations so that there is no change in behavior after migration.from TPS to Experion The option when enabled will allow the Experion controllers to have the same behavior as the TPS xPM controller. See LEGACYGAP Option in R300
SetPoint	• SP (SP) – Lets you specify an initial set point value. The default value is 0. • High Limit (SPHILM) – Lets you specify a high limit

Configuration Tab	Description
	value for the SP. If the SP value exceeds this limit, the block clamps the SP to the limit value and sets the SP high flag (SPHIFL). The default value is 100. • Low Limit SPLOLM) – Lets you specify a low limit value for the SP. If the SP value falls below this limit, the block clamps the SP to the limit value and sets the SP low flag (SPLOFL). The default value is 0. • Mode (TMOUTMODE) – Lets you select the desired MODE the block is to assume, if an initializable input times out, which means the input has not been updated within a designated timeout time. The selections are AUTOmatic, BCAScade, CAScade, MANual, NONE, and NORMAL. The default selection is MANual. • Time (TMOUTTIME) – Lets you specify a time in seconds that must expire before the block assumes that its input update has timed out. The block must be in CAScade mode for it to monitor its primary input for timeout. The default setting is 0, which means the timeout function is disabled. If the input is from a connection in another controller in a peer-to-peer architecture, the actual timeout time equals the configured TMOUTTIME plus the CDA timeout time. The CDA timeout time equals four times the configured CEE subscription rate. For example, if the CEE subscription rate is 100 milliseconds and the TMOUTTIME is 5 seconds, the actual timeout time for the block is 4 times 100ms plus 5s or 5.4 seconds. • Enable Advisory SP Processing (ADVDEVOPT) – Lets you specify whether or not the block is to generate a deviation alarm when the PV deviates from a user specified “advisory” SP value. The default selection is unchecked (Disabled). • Advisory SP Value (ADVSP) – Lets you set an advisory SP value in PV engineering units, when Advisory SP Processing is enabled. When PV exceeds or deviates from this value, the block generates an advisory deviation alarm. • Enable PV Tracking (PVTRAKOPT) – Lets you specify if PV tracking is to be applied to this block or not. When PV tracking is enabled, this option sets the

Configuration Tab	Description
	SP equal to PV when the operation of a cascade loop is interrupted by either initialization, operator or program operation (such as, setting the MODE to MANual). This option is normally enabled for PIDs in a cascade loop. The default selection is unchecked (disabled). See the PV tracking section for this block for more details. • Enable PV Tracking in Auto/Init (PVTRAKOPTAI) – Lets you specify if PV Tracking option for the specific case when the block is in Auto mode and it is undergoing initialization (INITMAN is ON) is be enabled or not. The default selection is unchecked (disabled). • Enable SP Ramping (SPTVOPT) – Lets you specify if an operator can initiate a set point ramp action or not. It provides a smooth transition from the current set point value to a new one. The default selection is box unchecked (disabled). See the Set point ramping section for this block for more details. • Normal Ramp Rate (SPTVNORMRATE) – Lets you specify a ramp rate in engineering units per minute for the SP ramping function, when it is enabled. This lets an operator start the SP ramping function without specifying a ramp time. The default selection is Not a Number (NaN). See the Set point ramping section for this block for more details. • Max. Ramp Deviation (SPTVDEVMAX) – Lets you specify a maximum ramp deviation value in engineering units per minute for the SP ramping function, when it is enabled. Keeps PV within the specified deviation range for a ramping SP by stopping the SP ramp until the PV input catches up with the SP value. The default value is NaN, which means no ramp deviation check is made. See the Set point ramping section for this block for more details • Enable SP Push: (PUSHSP) – Lets you specify that the PID SP will be pushed from an Inter Cluster Gateway when the PID is the secondary of a cascade that extends over two Experion clusters.
Output	• High Limit (%) (OPHILM) – Lets you specify the

Configuration Tab	Description
	output high limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you enter a High Limit of 90%, the high limit in engineering units is 90% times 450 or 405 + 50 (CVEULO) equals 455. This check is not applied for a function block that is in the MANual mode. The default value is 105%. • Low Limit (%) (OPLOLM) – Lets you specify the output low limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you enter a Low Limit of 10%, the low limit in engineering units is 10% times 450 or 45 + 50 (CVEULO) equals 95. This check is not applied for a function block that is in the MANual mode. The default value is -5%. • Extended High Limit (%) (OPEXHILM) – Lets you specify the output extended high limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you use the default value of 106.9%, the extended high limit in engineering units is 106.9% times 450 or 481.05 + 50 (CVEULO) equals 531.05. This check is not applied for a function block that is in the MANual mode. The default value is 106.9%. • Extended Low Limit (%) (OPEXLOLM) – Lets you specify the output extended low limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you use the default value of -6.9%, the extended low limit in engineering units is -6.9% times 450 or -31.05 + 50 (CVEULO) equals 18.95. This check is not applied for a function block that is in the MANual mode. The default value is -6.9%. • Rate of Change Limit (%) (OPROCLM) – Lets you specify a maximum output rate of change limit for both the positive and negative directions of the output in percent per minute. This lets you prevent an excessive rate of change in the output so you can match the slew rate of the control element to the control dynamics. We recommend that you configure this value before you tune the loop, so tuning can accommodate any slow down in response time caused by this rate limiting.

Configuration Tab	Description
	This check is not applied for a function block that is in the MANual mode. The default value is Not a Number (NaN), which means no rate limiting is applied. • Minimum Change (%) (OPMINCHG) – Lets you specify an output minimum change limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). This lets you define how much the OP must change before the function block outputs a new value. It filters out changes that are too small for the final control element to respond to. This check is not applied for a function block that is in the MANual mode. The default value is 0, which means no change limiting is applied. • Safe OP (%) (SAFEOP) – Lets you specify the safe output value as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 0 to 500 and you enter a Safe OP of 50%, the safe output value in engineering units is 50% times 500 or 250. The default value is Not a Number (NaN), which means the OP is held at its last good value. ' • CVEU Range High (CVEUHI) –Lets you specify the high output range value in engineering units that represents 100% full scale CV output for the block. The default value is 100. • CVEU Range Low (CVEULO) – Lets you specify the low output range value in engineering units that represents the 0 full scale CV output for the block. The default value is 0 (zero). • Output Bias (OPBIAS.FIX) – Lets you specify a fixed bias value in engineering units that is added to the Calculated Variable (CV) output value. See the Output Bias section for this function block for details. The default value is 0, which means no value is added. • Output Bias Rate (OPBIAS.RATE) – Lets you specify an output floating bias ramp rate in engineering units per minute. This bias rate is only applied when the floating bias in non-zero. See the Output Bias section for this function block for details. The default value is Not a Number (NaN), which means no floating bias is calculated. As a result, if the primary does not accept the block's initialization value, a bump in OP occurs.

Configuration Tab	Description
Alarms	• Type – Identifies the types of alarms this block supports. Of course, these alarms also interact with other block configuration options such as the Safety Interlock Option (SIOPT) and Bad Control Option (BADCTLOPT). The types are: – OP High (OPHIALM.FL) – OP Low (OPLOALM.FL) – Deviation High (DEVHIALM.FL) – Deviation Low (DEVLOALM.FL) – Advisory Deviation (ADVDEVALM.FL) – Safety Interlock (SIALM.FL) – Bad Control (BADCTLALM.FL) • Enable Alarm (ADVDEVOPT and SIALM.OPT) – Lets you enable or disable Advisory Deviation and/or Safety Interlock alarm types. A check in the box means the alarm is enabled. The default selections are unchecked or Disabled for Advisory Deviation and checked or Yes (enabled) for Safety Interlock. You can also configure the ADVDEVOPT and SIALM.OPT parameters as a block pins, configuration and/or monitoring parameters so they appear on the block in the Project and Monitoring tree views, respectively. • Trip Point – Lets you specify the following trip points for the given alarm. The default value is NaN, which disables the trip point. • OPHIALM.TP (Output High Alarm Trip Point) • OPLOALM.TP (Output Low Alarm Trip Point • DEVHIALM.TP (Deviation High Alarm Trip Point) • DEVLOALM.TP (Deviation Low Alarm Trip Point) • ADVDEVALM.TP (Advisory Deviation Alarm Trip Point) • Priority – Lets you set the desired priority level individually for each alarm type (OPHIALM.PR,

Configuration Tab	Description
	OPLOALM.PR, DEVHIALM.PR, DEVLOALM.PR, ADVDEVALM.PR, SIALM.PR, BADCTLALM.PR,). The default value is LOW. The levels are: – NONE - Alarm is neither reported nor annunciated. – JOURNAL - Alarm is logged but it does not appear on the Alarm Summary display. – LOW, HIGH, URGENT - Alarm is annunciated and appears on the Alarm Summary display. • Severity – Lets you assign a relative severity individually for each alarm type (OPHIALM.SV, OPLOALM.SV, DEVHIALM.SV, DEVLOALM.SV, ADVDEVALM.SV, SIALM.SV, BADCTLALM.SV,) as a number between 0 to 15, with 15 being the most severe. This determines the alarm processing order relative to other alarms. The default value is 0. • Deadband Value (ALMDB) – Lets you specify a deadband value that applies to all analog alarms to prevent nuisance alarms due to noise at values near the trip point. The default value is 1. Note that this value is loaded to the individual alarm parameters (for example, OPHIALM.DB and OPLOALM.DB) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB. • Deadband Time (ALMTM) – Lets you specify a time in seconds to define how long an analog alarm must exist before it is set true. The default value is 0, which means the alarm is set true as soon as the value exceeds the deadband value. Note that this value is loaded to the individual alarm parameters (for example, OPHIALM.TM and OPLOALM.TM) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB. • Deadband Units (ALMDBU) – Lets you specify if the

Configuration Tab	Description
	deadband value represents percent or engineering units. The default value is percent. Note that this value is loaded to the individual alarm parameters (for example, OPHIALM.DBU and OPLOALM.DBU) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB.
SCM	• SCM Mode Tracking Option (MODETRACK) – Lets you select the desired Mode Tracking function for the SCM associated with this block's Control Module. It defines how the FB will set the state of the MODEATTR based upon the MODE of the SCM. See the *Sequential Control Module User's Guide* for more information on this function. The default selection is ONESHOT. The selections are: – None – ONESHOT – SEMICONT – CONTRTN – CONT • Option Type – Lets you specify the action the function block is to take when the SCM goes into an abnormal state. The Starting State Option (STARTOPT) applies when the SCM state is Checking, Idle, or Complete. The Stop/Abort State Option (STOPOPT) applies when the SCM state is Stopping or Stopped, Aborting or Aborted. The Hold State Option (HOLDOPT) applies when the SCM state is Holding or Hold. The Restart State Option (RESTARTOPT) applies when the SCM state is Resume or Run. The NONE and LASTREQ are the only selections for the Restart State Option. You can select from one of these types for the other options as applicable for the given regulatory control function block: – NONE - No changes. – MAN - Set MODEREQ = MANUAL. – AUTO - Set MODEREQ = AUTOMATIC.

Configuration Tab	Description
	– CAS - Set MODEREQ = CASCADE. – FIXEDOP - Set OPREQ = Configured Value. – HOLDPV - Set SPREQ = PV. – FIXED SP - Set SPREQ = Configured Value and SPRATEREQ = NaN. – RAMPEDSP - Set SPTVREQ = Configured Value and SPRATEREQ = Configured Rate. • Value (STARTVAL, STOPVAL, HOLDVAL) – Depending upon Option Type selection, lets you specify an output or set point value within the respective range. For output, within OPEXLOLM to OPEXHILM and within SPLOLM to SPHILM, for set point. The default value is NaN (Not a Number). • Rate (STARTRATE, STOPRATE, HOLDRATE) – When the RAMPEDSP option is selected, lets you specify a rate value (STARTRATE, STOPRATE, HOLDRATE) for setting the SPRATEREQ for an SP ramping function.
Block Pins	Lets you select the available parameters that you want to expose as input/output pins on the function block graphic in Control Builder.
Configuration Parameters	Lets you select the available parameters that you want to appear on the face of the function block in the Project tab in Control Builder.
Monitoring Parameters	Lets you select the available parameters that you want to appear on the face of the function block in the Monitoring tab in Control Builder.
Block Preferences	Lets you change several block-viewing preferences including the color of the block's faceplate.
Insertion Type	Lets you include an insertion type from a CAB instances in the block. See *CAB insertion configuration considerations for regulatory control blocks* for more information

Function

A PID requires two inputs – a process variable (PV) and a set point (SP):

- PV is pulled from another function block.
  PV is typically pulled from a Data Acquisition (DATAACQ) function block which performs PV limit checking and alarming.
- SP is pulled from another function block, or stored by the operator or a user program.
  If SP is pulled from a primary, the PID's Mode must be Cascade; and if it is stored by the operator or a user program, Mode must be Manual or Automatic. If Mode is Cascade, the PID must perform timeout checking on SP (to make sure the primary is periodically updating it).

A PID also has the following optional inputs. Typically, these are flags which may be stored by the operator or user program to change the normal operation of the PID.

- ESWAUTO, ESWCAS, ESWMAN and SI – Indicates if an external source, such as a user program, wants to change the PID's Mode:
  - If ESWAUTO = On, the external source wants to change the Mode to Auto.
  - If ESWCAS = On, the external source wants to change the Mode to Cascade.
  - If ESWMAN = On, the external source wants to change the Mode to Manual.
  - If SI = On, the external source wants to invoke the PID's safety interlock logic.

If a BACKCALC connection is made to the secondary, the PID reads BACKCALCIN from the secondary before calculating its OP:

- BACKCALCIN is a "data container", which means it contains many pieces of information but is accessed by a single read. Among other things, the information in BACKCALCIN indicates if the secondary is wound-up or if it wants the PID to initialize.
- The individual BACKCALCIN/BACKCALCOUT connections for each output used are automatically built by Control Builder as implicit/hidden connections. This means you do not have to manually wire BACKCALC connections in Control Builder.
- The secondary builds BACKCALCIN when it receives a read request from the primary. This way, BACKCALCIN is guaranteed to contain the most current status.

Functional scenario

This scenario is based on the functional block diagram of a typical cascade loop shown in the following figure and it assumes the following:

- The PID2's Mode is Cascade. As a result, SP is pulled from a primary (PID1), and the PID2 must perform timeout checking on it.
- Both PID1 and 2 pull PV from Data Acquisition (DATAACQ) function blocks as shown in following figure.
- The PID1 has an active output. As a result, it reads BACKCALCIN from and provides OP to the secondary (PID2).
- The PID2 will never be wound-up, and never request the PID1 to initialize. In addition, the PID1 will never be wound-up, and never request its SP to initialize.
- The PV, SP and OP connections are all good which means there are no communication errors or timeouts.

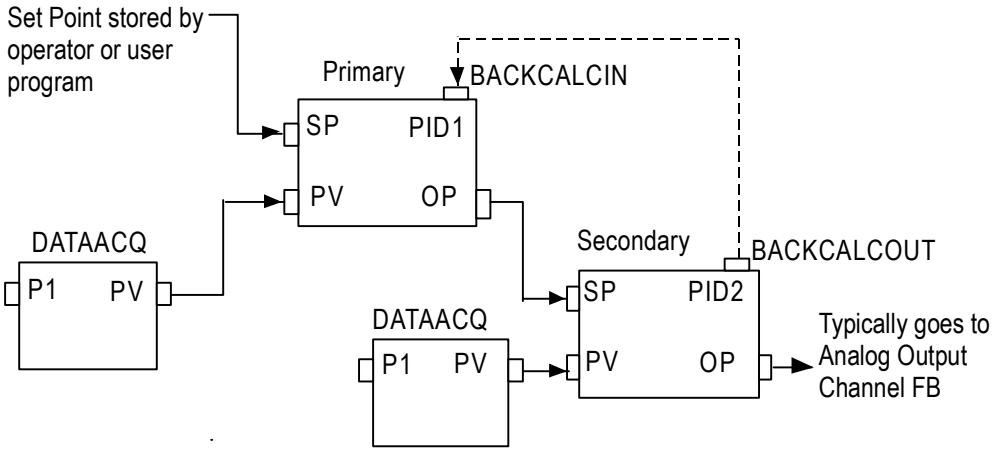


Figure 21 Functional block diagram of typical PID cascade operation.

The functional steps associated with this PID operating scenario are listed in the following table.

Step	Action
1	The PID1 provides a value to the PID2 SP variable (before the PID1 executes).
2	The PID1's "Execute" method is called by the CEE (Control Execution Environment). The PID execution period is configurable.
3	The PID2 performs timeout checking on SP (to make sure the variable has been updated). The SP timeout value is configurable.
4	The PID1 checks PVSOURCE and decides whether or not to fetch PV. If PVSOURCE = Auto, it brings PV from the DATAACQ; otherwise, it simply uses the current value of PV.
5	The PID1 checks SI, ESWAUTO, ESWCAS and ESWMAN to see if an external source wants to invoke Safety Interlock processing or change the Mode.
6	The PID1 reads BACKCALCIN from the secondary, and decides if windup or initialization processing is required. The BACKCALOUT to BACKCALIN connection is hidden.
7	The PID1 performs SP processing. (SP processing options are specified at configuration time.)
8	The PID1 calculates an output, based on PV and SP values and the configured algorithm.
9	The PID1 performs limit checking and alarming (if required) on OP.
10	The PID1 stores OP to the secondary.
11	The PID1's "Execute" method completes.

Configuration examples

- Single PID Loop: The following figure and its companion callout description table show a sample configuration that uses a PID block to form a single control loop for quick reference. The view in following figure depicts a loaded configuration in Monitoring mode.

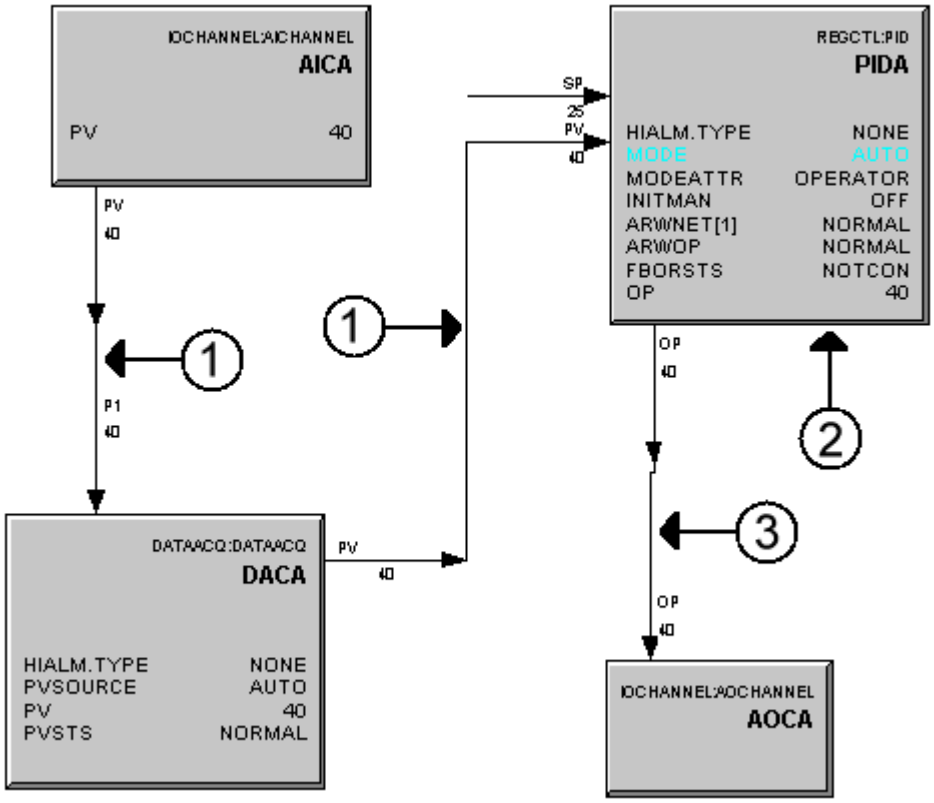


Figure 22 Example of CB configuration using a PID block for single loop control.

The following table includes descriptions of the callouts in the figure above.

Callout	Description
1	Use the PV parameter connection to carry data from the analog input to the PID block. The default PV connection is exposed, but the implicit/hidden connection function automatically makes a connection to a value/status parameter (PVVALSTS) when it is required.
2	Use the BACKCALCIN/BACKCALCOUT connection to carry secondary data from the AOC block to the primary PID block. If the PIDA block were a secondary block, its BACKCALCOUT pin connection would be connected to the BACKCALCIN pin connection on its primary PID block. The individual BACKCALCIN/BACKCALCOUT connections for each output used are automatically built by Control Builder as implicit/hidden connections. The secondary data includes this information. • Anti-Reset Windup Status: Indicates if the secondary's initializable input (which is this block's output) is at its high or low limit. • Initialization Request Flag: Used to request continuous initialization. If the flag is set (and this block is configured to accept secondary initialization), this block initializes itself for one cycle, and resumes normal processing on the next. • Oneshot Initialization Flag: Used to request oneshot initialization. If the flag is set (and this block is configured to accept secondary initialization), this block initializes itself for one cycle, and resumes normal processing on the next. • Initialization Value: Used for continuous and oneshot initialization. • Override Status: If a block is in an override strategy, this flag indicates whether it is the selected strategy or not. If the block is in an unselected strategy (and configured to accept secondary initialization), it invokes its override feedback processing. • Override Feedback Value: Similar to initialization value; this is calculated to prevent "wind-up" in unselected primaries. • Override Offset: Only applies to PID type function blocks. If a PID is in an unselected override strategy, this flag indicates how it should calculate its output. • Engineering Units: The engineering units (EU) of the secondary's initializable input. For example, If the secondary's input is SP, it sends SPEUHI and SPEULO to the primary. The primary then sets its CV range (CVEUHI and CVEULO) to this.
3	Use the OP parameter connection to send output data to the Analog Output

Callout	Description
	Channel (AOC) block. The default OP connection is exposed, but the implicit/hidden connection function automatically makes a connection to a value/status parameter (OPX/OPEUX) when it is required.

- Cascade PID Loop: The following figure and its companion callout description table show a sample configuration that uses two PID blocks to form a cascade control loop for quick reference. The view in the following figure depicts a loaded configuration in Monitoring mode.

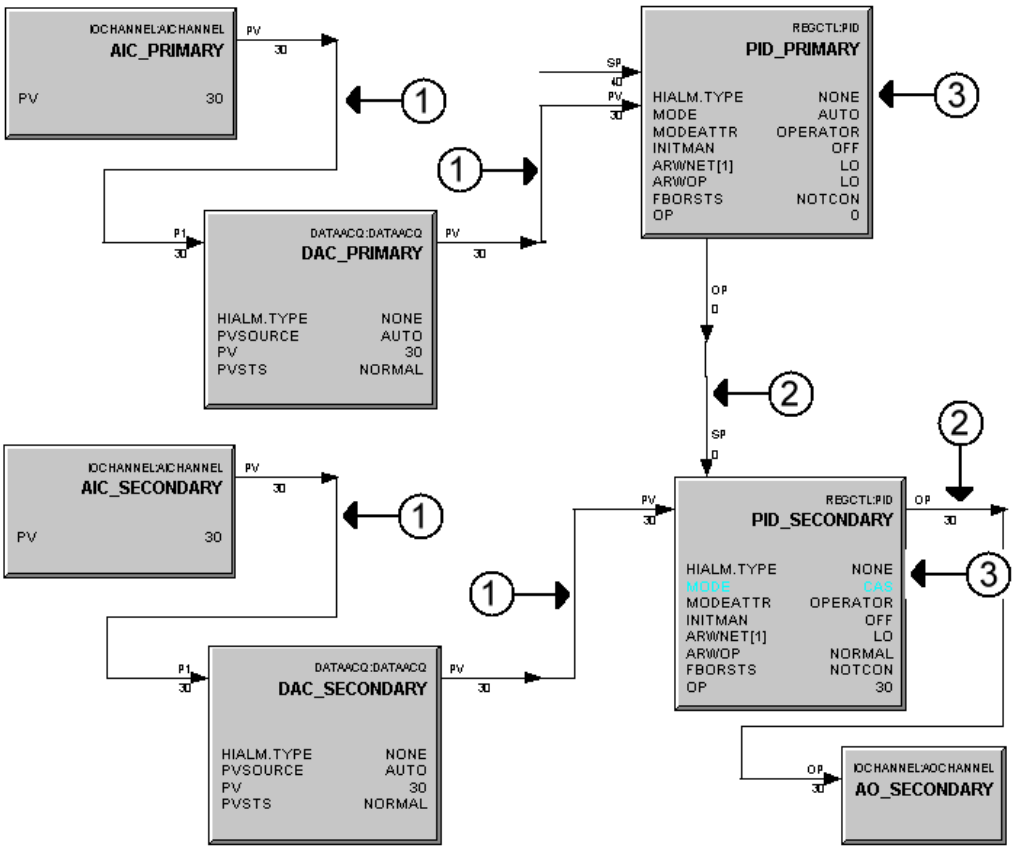


Figure 23 Example of CB configuration using two PID blocks for cascade loop control.

The following table includes descriptions of the callouts in the figure above.

Callout	Description
1	Use the PV parameter connection to carry data from the analog input to the PID block. The default PV connection is exposed, but the implicit/hidden connection function automatically makes a connection to a value/status parameter (PVVALSTS) when it is required.
2	Use the OP parameter connection to send output data to another block. The default OP connection is exposed, but the implicit/hidden connection function automatically makes a connection to a value/status parameter (OPX/OPEUX) when it is required.
3	Use the BACKCALCIN/BACKCALCOUT connection to carry secondary data from the AOC block to the secondary PID block and from the secondary PID block to the primary PID block. The individual BACKCALCIN/BACKCALCOUT connections for each output used are automatically built by Control Builder as implicit/hidden connections. See the description for Callout 2 in the table for the previous figure (*Example of CB configuration using a PID block for single loop control.*) for more detailed information about the elements that makeup the secondary data.

Operating modes and mode handling

The PID block operates in the following modes:

- MAN (MANual)
  - If mode is MANual, OP may be stored by the operator or a user program; PV and SP are ignored – if a primary exists, it goes to the initialized state.
- AUTO (AUTOmatic)
  - If mode is AUTOmatic, SP (or SPP) may be stored by the operator or a user program; if a primary exists, it goes to the initialized state. SP contains set point value in engineering units and SPP contains the value in percent.
- CAS (CAScade)
  - If mode is CAScade, SP is pulled from a primary; if the primary is off-control (that is, inactive or initializing) or the connection is bad, the PID block invokes timeout processing.

Required inputs

The required number of inputs is determined by the mode of the PID block.

- If Mode is CAScade, two inputs are required – PV and SP.
- If Mode is AUTOmatic or MANual, only PV is required.
  - SP is an initializable input; PV is non-initializable.
  - PV must be pulled from another block; you cannot store to it – typically it is connected to the output of an auxiliary or data acquisition (DATAACQ) block.
  - If Mode is CAScade, SP is pulled from another block; if Mode is AUTOmatic, it may be stored by the operator or a user program.
  - The PID block may have one primary or none, depending on whether SP is configured or not; there is one primary per initializable input.

Input ranges and limits

- You must specify a PV engineering unit range, PVEUHI and PVEULO.
  - PVEUHI and PVEULO define the full range of PV in engineering units.
    PVEUHI represents the 100% of full scale value.
    PVEULO represents the 0% of full scale value.
  - PVEUHI and PVEULO also define the engineering unit range of SP – PV and SP are assumed to have the same range.
- The PID block assumes PV is within PVEUHI and PVEULO – it applies no range check – however, PV typically comes from a data acquisition (DATAACQ) block which applies its own limit and range check.
- SPHILM and SPLOLM define set point operating limits in engineering units.
  - The operator is prevented from storing a set point value that is outside these limits; if the primary or a user program attempts to store a value outside of the limits, the PID block clamps it to the appropriate limit and sets the primary's windup status.
- SP contains set point value in engineering units and SPP contains the value in percent.
  - If Mode is AUTOmatic, the operator or a user program may store to either SP or SPP.

Initializable outputs

"Initializable output" and "initializable input" are variable attributes, similar to data type or access level. A variable with the "initializable" attribute has an associated BACKCALC variable, and when a connection is created between an initializable input and initializable output, you can also create a BACKCALC connection. Control Builder automatically builds the required BACKCALC connections, so you don't have to create them manually. These "implicit" build connections are "hidden" from view and the related parameter pins are not exposed on the control chart.

For example, if you connect OP from a PID block to a PID block or an AOCHANNEL block, Control Builder automatically creates the BACKCALCOUT to BACKCALCIN connection.

- OP = Calculated output, in percent.
- OPEU = Calculated output, in engineering units.

You may create a connection to OP or OPEU but not both. Therefore, this block may have only one secondary. If you do not create a connection to OP or OPEU, then the block does not have a secondary. Alternately, if you connect OP or OPEU to a non-initializable input, then this block does not have a secondary. (Note that the default OP connection pin is exposed on the blocks and the implicit/hidden connection function automatically makes the appropriate value/status parameter (OPX/OPEUX) connection when required. For example, if you connect the output from a primary PID block (PIDA.OP) to the set point of a secondary PID block (PIDB.SP), the implicit/hidden connection is made to PIDA.OPX to provide value/status data.)

ATTENTION

Be sure you use a FANOUT block to make multiple output connections. We recommend that you do not make multiple connections from a single PID output.

Control initialization

The PID block brings initialization requests from its secondary through BACKCALC. In addition, the secondary may propagate oneshot initialization requests to this block.

- Note that SECINITOPT may be used to ignore initialization requests from the secondary.
- If the secondary is requesting initialization, the PID block:
  - initializes its output
    CV = initialization value from the secondary
  - sets initialization request parameters for its primary
    INITREQ = On
    INITVAL = SP

Output bias

If the PID block algorithm is configured as Equation E, the output bias (OPBIAS) is added to the algorithm's Calculated Value (CV) and the result is stored in CV. CV is later checked against OP limits and, if no limits are exceeded, copied to the output.

If the PID block algorithm is configured as Equation A, B, C, or D, no output bias (OPBIAS) is applied.

The OPBIAS is the sum of the user-specified fixed bias (OPBIAS.FIX) and a calculated floating bias (OPBIAS.FLOAT). The purpose of the floating bias is to provide a bumpless transfer when the function block initializes or changes mode as long as the PID block is the first initializable block.

- OPBIAS is recomputed under the following conditions to avoid a bump in the output. (Note that the PID block only applies OPBIAS.FLOAT to the output for the latter two conditions, when it is the first initializable block.)
  - When the function block starts up (that is, goes Active).
  - When the function block initializes (for example, the secondary requests initialization).
  - When the mode changes to Auto or Cascade.

ATTENTION

When the mode is Manual, OPBIAS is not used, since the output is not calculated. This means that the OPBIAS is not recomputed, when the mode changes to Manual.

- You may set the OPBIAS value only if the function block is Inactive or Mode equals Manual. This is done to prevent a bump in the output when the bias is changed. The following occurs when you set the OPBIAS value.
  - The total bias (OPBIAS) and fixed bias (OPBIAS.FIX) are both set to the entered value.
  - The floating bias (OPBIAS.FLOAT) is set to zero.

ATTENTION

When the function block goes Active or the Mode changes to Auto or Cascade, OPBIAS and OPBIAS.FLOAT are recomputed.

- There are no limit checks applied when you set an OPBIAS value. However, after the bias is added to CV, the result is compared against the output limits and clamped, if necessary.
- You configure the value for the fixed bias (OPBIAS.FIX) and it is never overwritten by the floating bias (OPBIAS.FLOAT). This means the total bias will eventually equal the OPBIAS.FIX , if you configure OPBIAS.RATE to ramp down OPBIAS.FLOAT.
- The OPBIAS.FLOAT is calculated as follows.

$$\text{OPBIAS.FLOAT} = CV_{INIT} - (CV_{UNBIASED} + \text{OPBIAS.FIX})$$

Where:

CV_{INIT} = initialization value received from the secondary

$CV_{UNBIASED}$ = unbiased calculated value (based on input from the primary)

OPBIAS.FIX = fixed bias (user-specified)

- If the primary accepts this block's initialization request, then CV + OPBIAS.FIX should be the same as CVINIT and OPBIAS.FLOAT will be zero. In most cases, OPBIAS.FLOAT will be zero. However, if the primary does not accept this block's initialization request because the primary is a FANOUT block or it was configured to ignore initialization, then OPBIAS.FLOAT value will not be zero.

  If OPBIAS.FLOAT is not zero, you can configure it to ramp down to zero through the OPBIAS.RATE parameter.

- You configure the OPBIAS.RATE to apply a ramprate to the OPBIAS.FLOAT. It is only used when the OPBIAS.FLOAT is not zero. The OPBIAS.RATE is expressed in Engineering Units per minute and may have the following values.

  – Zero:
    If OPBIAS.RATE is zero, a OPBIAS.FLOAT is calculated and bumpless transfer is guaranteed. However, if OPBIAS.FLOAT is not zero, it will never ramp down.

  – Non-zero:
    If OPBIAS.RATE is not zero, an OPBIAS.FLOAT is calculated and bumpless transfer is guaranteed. If the OPBIAS.FLOAT is not zero, it is ramped to zero at the rate you configured for the OPBIAS.RATE parameter.

    The function block ramps the OPBIAS.FLOAT to zero by applying the following calculation each time it executes.

    OPBIAS.FLOAT = OPBIAS.FLOAT – (OPBIAS.RATE / cycles_per_Min)

  Where:

    cycles_per_min = number of times the function block executes per minute (calculated)

  – NaN:
    When the OPBIAS.RATE is Not a Number (NaN), no OPBIAS.FLOAT is calculated. This means a bump in the output will occur, if the primary does not accept this block's initialization value.

Output ranges and limits

- CVEUHI and CVEULO define the full range of CV in engineering units.
  - If the PID block has a secondary, its CV range must be the same as the secondary's input range – if this PID function has a secondary, it brings the secondary's input range through BACKCALC and sets its CV range to that.
  - If the PID block has no secondary, CVEUHI and CVEULO must be specified.
  - Note that this PID block brings the secondary's input range regardless of SECINITOPT (that is, regardless of whether the secondary's initialization and override data are used).
- OPHILM and OPLOLM define the normal high and low limits for OP as a percent of the CV range – these are user-specified values.
  - OP is clamped to these limits if the algorithm's calculated result (CV) exceeds them, or another block or user program attempts to store an OP value that exceeds them, however, the operator may store an OP value that is outside of these limits.
- OPEXHILM and OPEXLOLM define the extended high and low limits for OP as a percent of the CV range – these are user-specified values.
  - The operator is prevented from storing an OP value that exceeds these limits.

Output limit behavior for PID block recovering from Bad Control

If PID block is in Manual mode or has shed to Manual mode and is recovering from a Bad Control condition, its output clamps to the normal high and low limits (OPHILM/OPLOLM) rather than the extended high and low limits (OPEXHILM/OPEXLOLM) when the secondary block comes out of initialization.

Since the PID block is in Manual mode, the output (OP) can be set to the extended limit from the Limit value, when the secondary block comes out of initialization with an initialization value (INITVAL[1..8]) higher/lower than normal output limits, respectively.

TIP

Be sure the output limits (OPHILM/OPLOLM) of the PID block are the same or within the secondary's limits.

Direct or reverse control

A PID block may be configured for direct-control action or reverse-control action.

Changing the control action effectively changes the sign of the gain.

- With direct-control action, an increase in the error (PV - SP) increases the PID output (CV).
- With reverse-control action, an increase in the error (PV - SP) decreases the PID output (CV).

For example, if SPP = 50% and PVP = 51%, then the error is 1%.

- With direct-control action, if PVP changes to 52%, the error increases causing CV to increase.
- With reverse-control action, if PVP changes to 52%, the error increases causing CV to decrease. .

Set Point Ramping

The Set Point Ramping option lets you ramp from the current set point value to a target set point value. You enable this option by selecting the Enable SP Ramping check box on the block's parameter configuration form. This is equivalent to setting the SPTVOPT parameter to Enable. You can also configure the following related parameters through the configuration form or the equivalent parameters.

Parameter	Description
Normal Ramp Rate (SPTVNORMRATE)	Normal ramp rate value in engineering units that you enter. The value can be Not a Number (NaN) or greater than zero. If value is NaN, it means a "step change" in the SP, which is the same as a ramp time of zero. This parameter lets you start SP ramping without specifying a ramp time. This function block calculates a ramp time (SPTVTIME) and ramp rate (SPTVRATE) as follows, when SP ramping is enabled: • If SPTVNORMRATE is a value other than zero or NaN: SPTVRATE = SPTVNORMRATE SPTVTIME = |(SPTV – SP)| / SPTVRATE • Otherwise,: SPTVRATE = NaN SPTVTIME = 0 (That is, do a step change.)

Parameter	Description
Max. Ramp Deviation (SPTVDEVMAX)	Lets you specify a maximum deviation in engineering units per minute allowed between PV and SP during ramping. The value can be NaN or greater than zero. If value is NaN, it means no ramp deviation checking is done. If the maximum ramp deviation value is other than NaN, SP ramping stops when the absolute value of the deviation (|PV – SP|) exceeds the maximum deviation. The deviation flag (SPTVDEVFL) is set, and SP ramping state (SPTVSTATE) remains in Run. Ramping resumes as soon as the absolute value of the deviation returns within the maximum deviation limit. This also resets the deviation flag (SPTVDEVFL). If you have entered a ramp time (SPTVTIME) and ramping is interrupted by maximum ramp deviation, the actual ramp time (SPTVTIME) will be greater than the time you specified.

You can configure these other SP ramping related parameters to appear as block pins or monitoring parameters that can be viewed on the block during Control Builder monitoring, as shown in the following figure. You can access these parameters to invoke and monitor SP ramping while monitoring the control strategy through Control Builder or the PID Loop Point Detail display in Station.

Parameter	Description
SPTV	SP target value that you enter. You can only set SPTV when the SPTVOPT is Enabled, the SPTVSTATE is Off or Preset, and the block's mode is Auto or Manual. When you set SPTV with the block's Control Module active, this occurs: • The block calculates a ramp time (SPTVTIME). • The SPTVSTATE goes to Preset. • A "P" modifier appears next to the SP value on the PID detail display in Station.
SPTVDEVFL	SP target value deviation flag indicates when deviation exceeds the maximum ramp deviation limit.

Parameter	Description
SPTVRATE	SP target value ramp rate. This rate is calculated as shown above for the SPTVNORMRATE and as follows: • If you specify a ramp time (SPTVTIME) value other than zero: SPTVRATE = |(SPTV – SP)| / SPTVTIME Otherwise: SPTVRATE = NaN • If you change the SPTVNORMRATE, this block recalculates the ramp time (SPTVTIME) and ramp rate (SPTVRATE) as follows: If ramp time (SPTVTIME) is a value other zero: SPTVRATE = SPTVNORMRATE SPTVTIME = |(SPTV – SP)| / SPTVRATE Otherwise,: SPTVRATE = NaN SPTVTIME = 0 (That is, do a step change.)
SPTVTIME	SP target value time. This time is calculated in conjunction with SPTVRATE as described above or is entered by you. You can only set SPTV when the SPTVOPT is Enabled, the SPTVSTATE is Off or Preset, and the block's mode is Auto or Manual.
SPTVSTATE	SP target value state. The possible states are: • Off, • Preset, or • Run You can only set the SPTVSTATE when the Control Module containing this block is active and the block's mode is Auto. When you set SPTVSTATE to Run from Preset, this occurs: • An "R" modifier appears next to the SP value on the PID detail display in Station. • SP begins to ramp toward SPTV and SPTVTIME decreases. When SPTVTIME reaches zero, SP equals SPTV and the SPTVSTATE goes to Off.

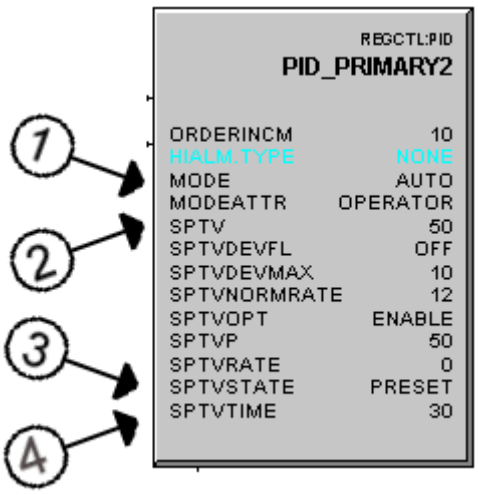


Figure 24 PID block with SP ramping parameters configured for monitoring.

The following table includes descriptions of the callouts in the figure above.

Callout	Description
1	Block's mode must be Auto and SPTVSTATE must be Preset, before you can start SP ramping by setting SPTVSTATE to Run with SPTV set to desired value.
2	The SPTVSTATE automatically goes to Preset, when: • You set a value for SPTV or SPTVTIME. • Mode changes to Manual while SPTVSTATE is Run. • Block is initialized (INITMAN = ON) while SPTVSTATE is Run. However, a oneshot initialization does not cause a change in SPTVSTATE. The SPTVSTATE automatically goes to Off, when: • SP is set by you, a program or another function block. • Mode changes to Cascade or Backup Cascade. • Control Module goes Inactive.

Callout	Description
3	You can only set a value for SPTV and SPTVTIME, when: • SPTVSTATE is Off or Preset, and • Mode is Auto or Manual.

ATTENTION

- When SP ramping is Enabled, the SPTVSTATE must be Off before you can make changes to the SP limits (SPHILM and SPLOLM).
- If the anti-reset windup status (ARWNET) indicates that SP is woundup (Hi, Lo or HiLo), SP ramping stops. When ARWNET indicates that SP has returned to normal, SP ramping continues from where it stopped.

PV tracking

The PV Tracking option sets SP equal to PV when a cascade is broken due either to function block initialization or operator or program action (such as, setting the mode to Manual).

You select the Enable PV Tracking selection on the block configuration form to enable the function (PVTRAKOPT = Track).

Typically, PV tracking is configured for PID blocks in a cascade configuration strategy. This allows the PIDs to resume control with no error after initialization or when they are taken out of Manual mode.

If PV tracking is configured, the PID block sets SP equal to PV (subject to SP limits) when either of the following conditions exist:

- PID block is in Manual mode
- PID block is initializing and not in Auto mode.

ATTENTION

- PV tracking does not occur on recovery from a bad PV.
- PV tracking does not occur if PID block is in Auto mode.
  - If PID block is in Auto mode, it means SP is normally stored by the user.
  - If PV tracking is initiated, this value is lost.
- The PVTRAKOPTAI parameter provides a PV Tracking option for the specific case when the block is in Auto mode and it is undergoing initialization (INITMAN is On). The PV Tracking option parameter (PVTRAKOPT) does not include the Auto/Init case, but does cover several other situations as noted above.

PID equations

The PID block provides five different equations for calculating the PID – the CTLEQN parameter is used to specify the desired equation.

- Equation A – all three terms (Proportional, Integral, Derivative) act on the error (PV - SP) as follows:

$$CV = K * L^{-1}\left[\left(1 + \frac{1}{T1_S} + \frac{T2_S}{1 + a * T2_S}\right) * \left(PVP_S - SPP_S\right)\right]$$

- Equation B – the proportional and integral terms act on the error (PV - SP) and the derivative term acts on changes in PV as follows:

$$CV = K * L^{-1}\left[\left(1 + \frac{1}{T1_S} + \frac{T2_S}{1 + a * T2_S}\right) * PVP_S - \left(1 + \frac{1}{T1_S}\right) * SPP_S\right]$$

- This equation is used to eliminate derivative spikes in the control action as a result of quick changes in SP.
- Equation C – the integral term acts on the error (PV - SP) and the proportional and derivative terms act on changes in PV as follows:

$$CV = K * L^{-1}\left[\left(1 + \frac{1}{T1_S} + \frac{T2_S}{1 + a * T2_S}\right) * PVP_S - \left(\frac{1}{T1_S}\right) * SPP_S\right]$$

- This equation provides the smoothest and slowest response to SP changes.
- Equation D – integral control only as follows:

$$CV = L^{-1}\left[\frac{1}{T1_S} * \left(PVP_S - SPP_S\right)\right]$$

- Equation E – proportional only as follows:

$$CV = K * (PV - SP) + OPBIAS.FIX + OPBIAS.FLOAT$$

  – Output bias processing adds a fixed bias (user specified) and floating bias (calculated to provide bumpless transfer after initialization or mode change) to the unbiased CV.

ATTENTION

To prevent a bump in the output, you must configure the OPBIAS.RATE parameter for a value (in Engineering Units per minute) other than 0.0 (zero) or NaN (Not a Number) to enable the ramping function for the floating bias.

  – Reverse-control action causes the sign of the unbiased CV to be reversed.

  – If both options are selected, the unbiased CV is reversed first, and then the fixed and floating bias are added – neither the bias nor the final CV are reversed

Where:

CV	=	output of PID (Equations A, B, C, D) in percent or output of P-controller (Equation E only) in engineering units
K	=	gain (proportional term)
L^{-1}	=	inverse of the LaPlace transform
PV	=	process input value in engineering units
PVP	=	PV in percent

a	=	1/16 fixed rate amplitude
s	=	La Place operator
SP	=	set point value in engineering units
SPP	=	SP in percent
T1	=	integral time constant in minutes
T2	=	derivative time constant in minutes
OPBIAS.FIX	=	fixed bias (Equation E only)
OPBIAS.FLOAT	=	floating bias (Equation E only)

Gain options

If PID equation A, B, or C is selected, any of the following gain equations may be chosen:

- Linear Gain – provides a proportional control action that is equal to a constant (K) times the error.
  - This is the most commonly-used gain option – K is a user-specified constant and has a default value of 1.0.
- Gap Gain – used to reduce the sensitivity of the control action when PV is in a user-specified band (gap) around the set point.
  - Gap size and control action are specified at configuration time through the following parameters:

KLIN	Linear (normal) gain – to be used when PV is outside the gap.
KMODIFGAP	Gain-modification factor – to be used when PV is inside the gap. Range of KMODIFGAP = 0.0 to 1.0.
GAPLOLM	Lower limit of gap – in same engineering units as PV.
GAPHILM	Upper limit of gap – in same engineering units as PV.

  - Gain (K) is derived as follows:

When PV is outside the gap:

K = KLIN

When PV is inside the gap (SP - GAPLOLM <= PV <= SP + GAPHILM):

K = KLIN * KMODIFGAP

- Nonlinear Gain – provides control action that is proportional to the square of the error, rather than the error itself.
  - Gain (K) is derived as follows:

$$K = KLIN * \left[NLFORM + NLGAIN * \frac{|PV - SP|}{(PVEUHI - PVEULO)} \right]$$

Where:

KLIN = linear (normal) gain (user-configured)

NLFORM = nonlinear gain form (user-configured; may be 0 or 1)

NLGAIN = nonlinear gain (user-configured)

- External Gain – where, when gain (K) is selected, it is modified by an input value that can come from either the process, another function block, or a user program.
  - The main use of this option is to compensate for nonlinear process gain – you can tune the PID gain independently of the normal operating point of the process.
  - For example, in controlling the level of a tank whose cross-section is not constant, the gain could be modified to compensate for the nonlinear rate of level change that is caused by the changing shape of the tank.
  - Gain (K) is derived as follows:

K = KLIN * KMODIFEXT

Where:

KLIN = linear (normal) gain (user-configured)

KMODIFEXT = external gain modifier (such as from a user program)

Tuning Constant Change Considerations

WARNING

You cannot always reverse output (OP) resulting from changes you make to a tuning constant gain (K), integral time (T1) or derivative time (T2) in an online control loop.

You cannot undo a change in a tuning constant in an online control loop by simply changing the constant back to its original value. The output (OP) does not jump back to its original prior value just because you return the constant to its prior value. In this case, you must put the loop in MANUAL mode and set the output (OP) to the desired value before returning the loop to AUTO mode.

Timeout monitoring

If mode is CAScade, the PID block performs timeout monitoring on SP – if a good SP value is not received within a predefined time (TMOUTTIME), the PID block invokes timeout processing.

The maximum time between updates is specified by TMOUTTIME (in seconds)

- Enable timeout monitoring by setting TMOUTTIME to a non-zero value.
- Disable timeout monitoring by setting TMOUTTIME to zero.

Timeout processing

If mode is CAScade and SP times out, the PID block does the following:

- Sets the input timeout flag (TMOUTFL)
- Keeps SP at its last good value.
- Changes the mode to a user-specified TMOUTMODE.
- Requests the primary to initialize.

The PID block sets its cascade request flag (CASREQFL), if SP times out and sheds to AUTOmatic mode. This indicates that the block is waiting to return to the CAScade mode, and it will as soon as it brings a good SP value. When it receives a good SP value, the block does the following:

- Changes the mode back to CAScade.

- Updates the SP.

You **cannot** set the CASREQFL. However, it can be cleared by setting the block's MODE to MANUAL.

ATTENTION

If the input is from a connection in another controller in a peer-to-peer architecture, the actual timeout time equals the configured TMOUTTIME plus the CDA timeout time. The CDA timeout time equals four times the configured CEE subscription rate. For example, if the CEE subscription rate is 100 milliseconds and the TMOUTTIME is 5 seconds, the actual timeout time for the block is 4 times 100ms plus 5s or 5.4 seconds.

Windup handling

When a windup condition is reached, the PID block stops calculating the integral term, but continues to calculate the proportional and derivative term.

- A windup condition exists if:
  - PID block has a secondary and the secondary is in windup.
  - PID block's output exceeds one of the user-specified output limits (OPHILM, OPLOLM).

Windup processing

Every regulatory control type block maintains anti-reset windup status for its output (ARWOP) and each of its initializable inputs (ARWNET). The following table lists the posssible values for ARWOP and ARWNET parameters.

If the Value is . . .	Then, the Associated Parameter . . .
Normal	is free to move in either direction.
Hi	is at its high limit and it may only be lowered.
Lo	is at its low limit and it may only be raised.
HiLo	may not move in either direction.

Manual Mode Interaction

When the MODE of a regulatory control block is changed to Manual (Man), the block sets its windup status (ARWNET) to HiLo. This means that every block upstream in a cascade strategy will set its windup status (ARWNET and ARWOP) to HiLo.

ARWOP computation

The ARWOP indicates if the output (OP) can be raised or lowered. The PID-type function blocks use ARWOP to restrict integral control. When ARWOP contains a value other than **Normal**, the PID block stops integral control in the windup direction. Integral control continues in the other direction, as does proportional and derivative control. But, windup status has no impact on proportional and derivative control.

If a function block has a secondary, it fetches the secondary's windup status and recomputes its ARWOP. The conditions within the function block, such as output being at its high limit, also affect the ARWOP. The ARWOP is computed as follows, assuming the block has only one output or that it is not a FANOUT block.

If Any of the Following are True . . .	Then, ARWOP Equals . . .
This block is inactive.	**HiLo**
A secondary exists but this block cannot fetch secondary data from it (communications or configuration error).	
A secondary exists and its windup state equals HiLo	
This block is in initialization (INITMAN = On).	
A secondary exists and it is requesting this block to initialize.	
A secondary exists and its windup state equals Hi.	**Hi**
This block's output is at its high limit (OPHIFL = On).	
A secondary exists and its windup state equals Lo.	**Lo**
This block's output is at its low limit (OPLOFL = On).	

ARWNET computation

When ARWNET is HiLo, stores to SP are not limited, rather this is the status propagated to the primary. The only *limiting* anti-reset windup status ever does is to stop integral action in one or both directions on PID blocks. For any other regulatory control type block, ARWNET is not used for any kind of limiting. The ARWNET is computed as follows, assuming the block has only one output or that it is not a FANOUT block.

If Any of the Following are True . . .	Then, ARWNET Equals . . .
This block is inactive.	**HiLo**
The ARWOP equals HiLo.	
This block is in Manual mode (MODE = Man)	
The calculated value (CV) range (CVEUHI / CVEULO) is NaN.	
The CV is NaN	
This block is connected to a non-initializable primary	
The ARWOP equals Hi (Pid function blocks have a configurable Control Action option (CTLACTN). If CTLACTN = Reverse , ARWNET will track ARWOP; but if CTLACTN = Direct , ARWNET will be the opposite of ARWOP.)	**Hi**
The input from the primary is at a high limit. For example, SPHIFL = On.	
This block's output has reached its positive rate-of-change limit (OPROCPOSFL = On)	

If Any of the Following are True . . .	Then, ARWNET Equals . . .
The ARWOP equals Lo (Pid function blocks have a configurable Control Action option (CTLACTN). If CTLACTN = Reverse , ARWNET will track ARWOP; but if CTLACTN = Direct , ARWNET will be the opposite of ARWOP.)	**LO**
Input from the primary is at a low limit. For example, SPLO.FL = On.	
This block's output has reached its negative rate-of-change limit (OPROCNEGFL = On)	

Anti-Reset Windup Status

The anti-reset windup network in (ARWNETIN) and anti-reset windup output in (ARWOPIN) parameters are added in the standard anti-reset windup (ARW) computation logic. They are user configurable and allow stores from Sequential Control Modules (SCMs) and control algorithm block (CAB) programs.

The ARWNETIN and ARWOPIN parameters would be *ORed* into the existing standard logic so it is not lost. The following table summarizes the influence the ARWNETIN and ARWOPIN parameters have on the ARWNET and ARWOP parameters, which are not user configurable.

ARWNETIN or ARWOPIN Parameter Is. . .	Standard Computation Logic Is . . .	ARWNET or ARWOP Parameter Is . . .
NORMAL	NORMAL	NORMAL
NORMAL	HI	HI
NORMAL	LO	LO
NORMAL	HILO	HILO
HI	NORMAL	HI
HI	HI	HI
HI	LO	HILO
HI	HILO	HILO

ARWNETIN or ARWOPIN Parameter Is. . .	Standard Computation Logic Is . . .	ARWNET or ARWOP Parameter Is . . .
LO	NORMAL	LO
LO	HI	HILO
LO	LO	LO
LO	HILO	HILO
HILO	NORMAL	HILO
HILO	HI	HILO
HILO	LO	HILO
HILO	HILO	HILO

Override feedback processing

If the PID block is in a cascade strategy with a downstream OVRDSEL (Override Selector) block, it receives override feedback data. The data consists of an override status, override feedback value and an override offset flag. The status indicates if this block is in the selected or unselected strategy (as determined by the OVRDSEL block). The offset flag only applies to PID-type blocks.

Note: SECINTOPT may be used to ignore override requests from the secondary.

When the override status changes from selected to unselected, the PID block does the following:

- Recomputes CV:
  - If the override offset flag is Off:

    CV = override feedback value from secondary

  - If the override offset flag is On:
    - For PID with Equation D,

      Offset = (PVP-SPP)

    - For PID with any other Equation(A,B,C,E)

Offset = K * (PVP-SPP)

- If PID is using reverse control action:
  - then Offset = -Offset
- If the PID is using direct control action:
  - then Offset = Offset
- Additionally, the Offset term is set to 0.0 if

  → Offset > 0.0 and the downstream OvrdSel block is a High selector

  or

  → Offset < 0.0 and the downstream OvrdSel block is a Low selector

CV = override feedback value from secondary + Offset

The CV is clamped to OPHILM if it is greater than OPHILM and to OPLOLM if it is less than OPLOLM.

- Computes a feedback value for its primary:

feedback value for primary = PV

Where:

K = overall gain

PV = PV in engineering units

PVP = PV in percent

SPP = SP in percent

PID parameters

REFERENCE - INTERNAL

Refer to the *Control Builder Components Reference* for a complete list of the parameters used with the PID block.

PID-PL (Profit Loop PKS) Block

Description

The PID-PL block is a regulatory control block combining the functionality of PID controller with a robust, model-based, predictive controller and optimizer, Profit Loop PKS. This block inherits all the design features of a PID easily replacing the **PID** block in any control scheme. As such, it may be used in a single control loop, cascaded to other controllers, and configured for cross node supervisory control in an ACE.

In many cases, Profit Loop PKS provides superior control and should be used in place of the standard PID algorithms. These cases include:

- Processes with significant delay or inverse (wrong-way) response
- Property control employing infrequent measurement updates (such as from a gas chromatograph)
- Noisy processes where valve wear is problematic
- Dynamically changing processes
- Range control – Processes where range control will be beneficial, such as tank surge control. In this case, range control constrains the PV within a user specified range (gap) rather than to a fixed setpoint

Additional descriptive details are provided in the following discussions:

- Benefits
- Graphic presentation
- Modes
- Configurable attributes

Benefits

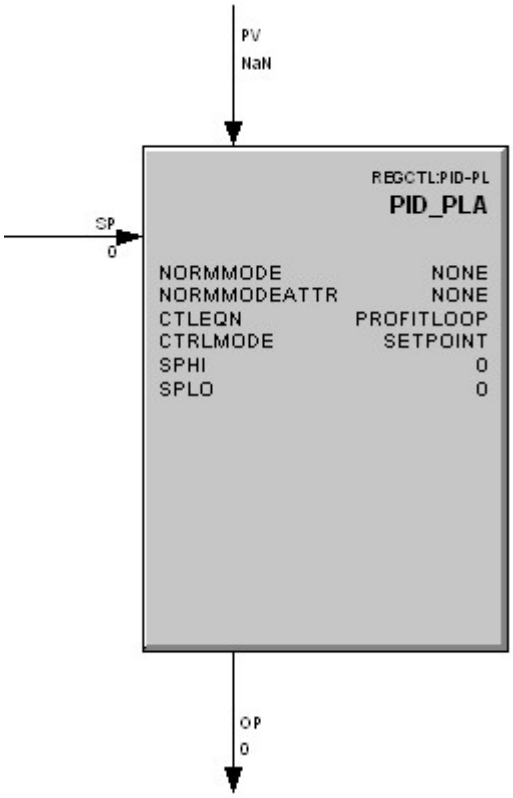
Key benefits of Profit Loop PKS include:

Benefit	Discussion
Superior control	Because it is a predictive controller, Profit Loop PKS directly accounts for transportation delays and other difficult to control behavior. This approach leads to superior control of processes with significant deadtime, inverse response, or noisy measurement.
Valve life extension	A well-designed robust controller maintains a balance between aggressive control performance and insensitivity to model uncertainty / noise. To achieve this balance, Profit Loop PKS employs a minimum movement algorithm. The control solution makes the least amount of movement to the control valve in order to meet the control objectives. This algorithm dramatically reduces valve chattering and stem reversals, especially when there is significant process measurement noise, which, in turn, leads to longer valve life.
Single tuning handle	Uses a single tuning “knob,” the performance ratio. If the controller is sluggish and unresponsive, decreasing the performance ratio will improve performance. If the controller is oscillatory or operating conditions change frequently, increasing the performance ratio will slow the controller response to gain robustness. Using a single handle as compared to the traditional three mode PID controller dramatically simplifies retuning of the loop while in service.
Integrated tools	Profit Loop PKS includes Profit Loop Assistant, a suite of tools for configuring, monitoring, and maintaining Profit Loop PKS blocks. The Profit Loop PKS Assistant is a companion intended to simplify your Profit Loop PKS configuration activities. With the Assistant, a good understanding of the math and concepts involved in model predictive control is not required; the Profit Loop PKS Assistant will aid you in the development of the model. In addition, the Profit Loop PKS Assistant provides diagnostic tools to assist with troubleshooting loop performance problems. For example, you can check for control valve stiction or enter specific tuning parameters for a PID-PL loop.
Efficiency	Specially designed implementation of model predictive control technology for small applications. The design minimizes the computations required, leading to more controllers per node running at faster execution speeds.
Profit Controller	Based on Honeywell’s award-winning Profit Controller technology.

Benefit	Discussion
Range control	Employs the patented Range Control Algorithm found in Honeywell's Profit Controller advanced control software. Unlike traditional PID, Profit Loop PKS is not restricted to a single target (the setpoint), but may actively constrain the response within a user-specified range. This capability makes it ideally suited to tank capacity (surge) control where the tank level must be bounded but is otherwise free to move. Here, the control objective is to minimize change to the in-flow or out-flow of the tank.
Optimization	When the process is not constrained to a setpoint, Profit Loop PKS provides freedom to choose the ultimate resting value for the process. The process may completely float within the range, or a secondary "optimization" objective may be imposed to drive the process to an optimal state.
Dual objective control	The key advantage of optimization over setpoint control is that the response of the optimizer and controller can be tuned separately, providing dual objective control.
Discrete analysis	As a predictive controller, Profit Loop PKS models the behavior of the PV based on changes in the OP. If PV measurements are not available, Profit Loop PKS proceeds with the control calculation using the current model estimate. Profit Loop PKS's ability to compensate for missing process measurement data makes it ideal for control of processes with discrete analyzers, such as gas chromatographs. Using the internal model, Profit Loop PKS avoids inter-sample integral wind-up associated with a PID controller.

Graphic presentation

The PID-PL block looks like this graphically:



Modes

The PID-PL operates in one of three distinct modes:

1. When CTLEQN is any one of the PID equations (EQA, EQB, and so on) and not PROFITLOOP, this block behaves as a standard PID control block. (For help on the this mode, see PID Block in the Reference Data for Functional Block Types, Regulatory Control Blocks section of this document.)

2. When CTLEQN is PROFITLOOP and CTRLMODE is set to SETPOINT, this block operates as a setpoint controller. Profit Loop PKS calculates a control output (OP) that drives the process variable (PV) to the setpoint (SP).

3. When CTLEQN is PROFITLOOP and CTRLMODE is set to RANGE, this block calculates a control output (OP) to merely constrain the process variable (PV) between an upper bound (SPHI) and lower bound (SPLO). A secondary objective can be applied to set the PV within the range, but specification of this objective is optional.

The remainder of this discussion concentrates on the latter two operating modes. Where appropriate, the term "setpoint control" will be applied to the second operating mode, and "range control" will be applied to the third. These two modes are distinguished by the setting of the CTRLMODE parameter.

Configurable attributes

Each PID-PL block supports the following user-configurable attributes. The following table lists the given name of the tab in the parameter configuration form and then describes the attributes associated with that tab. Several attributes parameters) are identical to those for the PID block; references are made as appropriate. This information is also included in Control Builder's on-line context-sensitive Help.

Configuration Tab	Description
Main	All attributes configured on this tab are the same as those for the **PID** block. Refer to either the online Help in Control Builder for details, or to *Reference Data for Functional Block Types*, *Regulatory Control Blocks*, PID Blockin this document.
Algorithm	Many attributes configured on this tab are the same as those for the **PID** block. Attributes specific to the PID-PL block are as follows: • Control Equation Type (CTLEQN) – For the PID-PL block, the Control Equation Type *must* be PROFITLOOP. If you want to change from PID-PL (Profit Loop PKS) to a standard PID block, the block must be inactive. For details on changing from **PID** to PID-PL (or PID-PL to **PID**), see the *Experion Control Building Guide,* Working with Profit Loop PKS, *Converting a PID-based Control Loop to PID-PL*. • ProfitLoop Control Mode (CTRLMODE) – Lets you select the means of control for the PID-PL block: – SETPOINT: With this selection, the function block will attempt

Configuration Tab	Description
	to have the PV track the setpoint (SP). – RANGE: With this selection, the function block will attempt to have the PV stay within high and low setpoint limits (SPHI and SPLO). • ProfitLoop Performance Ratio (PRFRATIO) – This setting is for non-integrating processes (D[1] <> 0). It defines how hard the function block will "push" to a setpoint or range limit. Enter a performance ratio value of 0.1 to 10.0 for the desired ratio of closed loop control response to open loop response. Note: – The default performance ratio of 1.0 represents steady-state control and reaches the control objective at the natural process response time. – Decreasing this ratio (PRFRATIO < 1) leads to faster, more aggressive, control. The process will be driven to its setpoint more quickly than its natural open loop response. – Increasing the ratio (PRFRATIO > 1) leads to slower, but more robust, control. The process will be driven to its setpoint more slowly than its natural open loop response. – For integrating processes, use Closed-loop response (see below) to specify integrator performance. In this case, Performance Ratio is set to 1. • ProfitLoop Closed-Loop Response (CLOSEDLOOPRESP) – Enter a time of greater than 0.1 minute for the desired closed-loop response time. This is the expected settling time for the control loop, which is the amount of time required to line out the controller after a setpoint change or large disturbance. For non-integration processes, CLOSEDLOOPRESP is the product of the open loop response time, OPENLOOPRESP, and the performance ratio, PRFRATIO. For details on non-integrating processes, see the Performance Ratio entry above. For details on remaining attributes on this tab, which are the same as those for the **PID** block, refer to either the online Help for Control Builder, or to *Reference Data for Functional Block Types*, *Regulatory Control Blocks*, PID Block In this document.
SetPoint	Many attributes configured on this tab are the same as those for the **PID** block. Attributes specific to the PID-PL block are as follows: • SP Range SP High (SPHI) – For PID-PL blocks with a Control

Configuration Tab	Description
	Mode of RANGE, this entry lets you specify the upper range limit, in Engineering Units. The value must be greater than or equal to SPLO, and less than SPHILM, that is: SPLO ≤ **SPHI** < SPHILM. • SP Range SP Low (SPLO) – For PID-PL blocks with a Control Mode of RANGE, enter the lower range limit, in Engineering Units. The value must be less than or equal to SPHI, and greater than SPLOLM, that is: SPLOLM < **SPLO** ≤ SPHI. • ProfitLoop Range Control Ramping, SPLO Ramp Rate (SPLORAMPRATE) – For PID-PL blocks with a Control Mode of RANGE, enter the ramp rate to be applied to external (operator or program) changes in the lower control limit, SPLO. The rate must be greater than 0 and expressed in Engineering Units/minute. When the active low limit, SPLOACTIVE, is below the operator-set value (SPLO), the active limit ramps toward SPLO at the rate specified here. Setting this rate to NaN disables ramping. • ProfitLoop Range Control Ramping, SPHI Ramp Rate (SPHIRAMPRATE) – For PID-PL blocks with a Control Mode of RANGE, enter the ramp rate to be applied to external (operator or program) changes in the higher control limit, SPHI. The rate must be greater than 0 and expressed in Engineering Units/minute. When the active high limit, SPHIACTIVE, is above the operator-set value (SPHI), the active limit ramps toward SPHI at the rate specified here. Setting this rate to NaN disables ramping. For details on remaining attributes on this tab, which are the same as those for the **PID** block, refer to either the online Help for Control Builder, or to *Reference Data for Functional Block Types*, *Regulatory Control Blocks*, PID Block in this document.
Output	Many attributes configured on this tab are the same as those for the **PID** block. The attribute specific to the PID-PL block is as follows: Valve Travel Reduction (VALVETRAVELRDCT) – Allows you to select the level of noise filtering to be applied to model bias. Possible selections are: • NORMAL: Apply no additional noise filtering.

Configuration Tab	Description
	• MODERATE: Apply some additional filtering. • FULL: Apply maximum additional filtering. Reducing the noise level leads to a reduction in valve travel with fewer valve reversals. There is little impact on the responsiveness of the control algorithm to fast disturbances. This filter complements the PV filter on the data acquisition block. For details on remaining attributes on this tab, which are the same as those for the **PID** block, refer to either the online Help for Control Builder, or to *Reference Data for Functional Block Types*, *Regulatory Control Blocks*, PID Block in this document.
Advanced	This tab is used for configuring the process model for your Profit Loop PKS controller. This is manual entry of a model in Laplace form. In addition, optimization, PV, and OP settings can be made, as detailed in the following discussions.
Advanced, Model area	The entries in this area of the Advanced tab allow you to define the transfer function model of process dynamics to be used in the control of your process. Use this form if you have a good understanding of the math and concepts involved in model predictive control and you do not wish to use the Profit Loop PKS Assistant. Entries or changes in this area are treated differently, depending on whether you are working in Project view or Monitoring view: • For Project mode: If entries or changes are made in the Model area, the parameter values are updated in the Engineering Repository Database (ERDB) immediately. • For Monitoring mode: If entries or changes are made in the Model area , they are displayed, but the function block is not updated until the Update Model command is selected. Attributes and actions possible on this tab should be considered. Attributes include the following: • Transfer Function # of Numerator Coefficients – The number of numerator coefficients, from 1 to 5, in the Laplace transfer function model of the process. This number should be at least one less than the number of denominator coefficients. • Transfer Function Numerator Coefficients – The desired value for each coefficient in the array defining the numerator polynomial in

Configuration Tab	Description
	the Laplace transfer function model of the process. N[1] is the coefficient preceding the s^0 (constant) term, while N[5] is the coefficient preceeding the s^4 term. If the coefficient will not be used, enter a value of 0. • Transfer Function # of Denominator Coefficients – The number of denominator coefficients, from 1 to 5, in the Laplace transfer function model of the process. This number should be at least one greater than the number of numerator coefficients. • Transfer Function Denominator Coefficients – The desired value for each coefficient in the array defining the denominator polynomial in the Laplace transfer function model of the process. D[1] is the coefficient preceding the s 0 (constant) term, while D[5] is the coefficient preceeding the s^4 term. • Process Gain – A nonzero value, for the gain associated with the Laplace transfer function model of the process. Use the value corresponding to the defined process gain units, which are indicated in the label of this field. Process gain units may be defined in percentage, %PV/%OP or Engineering Units, PVEU/OPEU. • Deadtime – A value of 0 - 3000 execution intervals, for the process deadtime (transportation lag) as defined in the Laplace transfer function model. Your entry depends on the execution period on the Control Module. • Process Gain Units – The units to be used in conjunction with the process gain: – PVEU/OPEU: Gain expressed in engineering units. – %PV/%OP: Gain expressed as a percentage of range. Actions that can be taken on this tab, related to the definition of your model, include: • Update Model – (UPDATEMODEL) – The model defined in the PID-PL configuration form can be downloaded to the active controller. Clicking this button updates model values in the ERDB, calculates the model, and calculates the controller. This button is inactive if the model in the configuration form matches the model in the active controller. • Reset Model (RESETMODEL) – The model in the PID-PL

Configuration Tab	Description
	configuration form can be reset to match the model in the active controller. The model in the controller will differ from the one defined in the configuration forms if you have made changes in the forms since the model was last updated, or a model has been defined in the Profit Loop PKS Assistant and downloaded to the controller. This button is inactive if the model in the configuration form matches the model in the active controller. • Launch Assistant – The Profit Loop PKS Assistant, a companion intended to simplify your Profit Loop PKS configuration activities, can be started from the Advanced tab. The Assistant provides several tools to help you with the model definition process. IMPORTANT! If you have a defined model in the Control Builder, PID-PL configuration form (Advanced tab), and you use the Profit Loop PKS Assistant to redefine the model, then download the model from the Assistant to the controller, the models will be different. As with other types of changes to the controller, you can upload the controller configuration to store the changes in Control Builder's Monitor-side Engineering Repository Database (ERDB), and then perform an update to store the changes in the Project-side ERDB. Refer to the *Experion Control Building Guide* for information on performing these procedures.
Advanced, Optimization area	With range control, because the process is no longer constrained to a setpoint, you are free to choose the ultimate resting value for the process. The process may completely float within the range, or you may wish to impose a secondary "optimization" objective to drive the process to an optimal state. Use this area to do so. For details on how optimization works, see the Control Builder online Help, Profit Loop PKS topic, "Understanding Optimization." • Optimizer Mode – When the Control Mode is set to Range (on the Algorithm form), you can allow the process to completely float within the range, or you may impose a secondary "optimization" objective to drive the process to an optimal state. Allowing the process to float within the range may be acceptable for dynamic control, but it may not be suitable for planning and long-term operations. The desired optimization modes are: – NONE: No optimization. – MINIMIZE: Optimize toward the lower limit of the range

Configuration Tab	Description
	(minimize the PV). – MAXIMIZE: Optimize toward the upper limit of the range (maximize the PV). – OPTTARGET: Optimize toward a specific target value (a user-specified SP). – DUALRANGE: Strive to keep the optimal solution between optimization limits, SPLOLMOPT and SPHILMOPT. All modes except OPTTARGET calculate the optimization limits as deviations from the PID-PL block's high and low operating range (SPHI and SPLO). In contrast, the OPTTARGET mode calculates the optimization limits as deviations from setpoint (SP). • Optimizer Speed – Dictates how fast the optimizer drives the process towards the optimal target. Enter a value of 0.1 to 10.0 to . This is a unitless value. Notes: With a speed of 6.0, the optimizer tries to bring the process to its steady-state value by the end its of control horizon (the optimizer is as aggressive as the controller). With a speed of 3, the optimizer is twice as slow; with 12, twice as fast. The default value for OPTSPEED is 2. • Profit Loop PKS uses the following formula: $\frac{6 \times \text{CLOSEDLOOP RESP}}{\text{OPTSPEED}} = \text{OPTRESP}$ – Where: CLOSEDLOOPRESP = Closed-loop response time OPTSPEED = Optimization speed OPTRESP = Settling time associated with changes induced through optimization • SP Offset, Low Limit (SPLOOPTOFFSET) – Profit Loop PKS can be configured to optimize the process over a limited operating range, which is more restrictive than the normal control range bounded by SPLO and SPHI. To do this, – the Control Mode (on the Algorithm form) must be set to RANGE, and – the Optimizer Mode (on the Advanced form) must be set to

Configuration Tab	Description
	other than NONE. – Set the SP low limit for optimization by entering the amount by which you want to restrict the lower limit (the “offset”). The optimized low limit is then calculated as follows: SPLOLMOPT = SPLO + SPLOOPTOFFSET The impact of this offset depends on the selected Optimizer Mode. • SP Offset, High Limit (SPHIOPTOFFSET) – Profit Loop PKS can be configured to optimize the process over a limited operating range, which is more restrictive than the normal control range bounded by SPLO and SPHI. To do this, – the Control Mode (on the Algorithm form) must be set to RANGE, and – the Optimizer Mode (on the Advanced form) must be set to other than NONE. – Set the SP high limit for optimization by entering the amount by which you want to restrict the higher limit (the “offset”). The optimized high limit is then calculated as follows: SPHILMOPT = SPHI - SPHIOPTOFFSET The impact of this offset depends on the selected Optimizer Mode. • OP Offset, Low Limit(OPLOOPTOFFSET) – Profit Loop PKS can be configured to optimize the process over a limited operating range, which is more restrictive than the normal control range bounded by OPLOLM and OPHILM. To do this, – the Control Mode (on the Algorithm form) must be set to RANGE, and – the Optimizer Mode (on the Advanced form) must be set to other than NONE. – Set the OP low limit for optimization by entering the amount by which you want to restrict the lower limit (the “offset”). The optimized low limit is then calculated as follows: OPLOLMOPT = OPLOLM + OPLOOPTOFFSET – The impact of this offset depends on the selected Optimizer Mode. • OP Offset, High Limit (OPHIOPTOFFSET) – Profit Loop PKS can

Configuration Tab	Description
	be configured to optimize the process over a limited operating range, which is more restrictive than the normal control range bounded by OPLOLM and OPHILM. To do this, – the Control Mode (on the Algorithm form) must be set to RANGE, and – the Optimizer Mode (on the Advanced form) must be set to other than NONE. – Set the OP high limit for optimization by entering the amount by which you want to restrict the higher limit (the "offset"). The optimized high limit is then calculated as follows: OPHILMOPT = OPHILM - OPHIOPTOFFSET The impact of this offset depends on the selected Optimizer Mode.
Advanced, PV Configuration area	Async PV Option (PVASYNCOPT) –This setting tells the function block when *not* to expect a new PV measurement (when not to expect a change), so that the function block does not react to obsolete information. • CONTINUOUS: The PV measurement is updated at least once with every execution period. Select this setting when the sensor involved generates a continuously updated signal, such as a thermocouple or strain gauge. • ONPVCHANGE: The PV measurement is updated when the PV parameter changes. This selection is appropriate with such sensors as gas chromatographs. IMPORTANT! Do not select ONPVCHANGE if noise alters the PV value between analyzer updates. • EXTERNALSYNC: The PV measurement is updated when an external flag, connected to NEWSAMPLE, is set. IMPORTANT! Use logic blocks to set and reset the NEWSAMPLE flag. Users are responsible for designing and constructing the appropriate logic.
Alarms	Most attributes configured on this tab are the same as those for the **PID** block. Attributes specific to the PID-PL block are as follows: • Predicted PV High – Trip Point (PREDPVHIALM.TP) – For PID-PL blocks, at every execution period, Profit Loop PKS estimates the

Configuration Tab	Description
	steady-state value for both the PV and OP. The predicted PV high alarm can be used to indicate when the predicted steady-state PV value exceeds the value specified here for more than a specified time (bad PV shed time). – A value greater than or equal to zero enables the predicted PV high alarm trip point. – NaN (Not a Number) disables the predicted PV high alarm. • Predicted PV High – Priority (PREDPVHIALM.PR) – The priority level for the corresponding alarm type can be selected, using the following categories: – NONE: The alarm is not reported to the system and is not annunciated. – JOURNAL: The alarm is written to the event file, where it can be accessed for alarm and event reports, or event displays on a Station. JOURNAL priority alarms do not appear in the Alarm Summary display. – LOW, HIGH, URGENT: These selections cause the alarm to be displayed in the alarm zone at the bottom of a Station display to bring new alarms to the attention of operators and/or directed to an alarm/event printer. The alarm also appears in the Alarm Summary display, in the color specified here: 1. LOW: Low-intensity yellow. 2. HIGH: Yellow. 3. URGENT: Red – Note that the default selection is LOW, and that alarm levels are prioritized as follows: JOURNAL --> LOW --> HIGH --> URGENT (highest priority) • Predicted PV High – Severity (PREDPVHIALM.SV) – A relative severity to the predicted PV high alarm can be assigned by specifying a weight in the range of 0 to 15. The specified weight determines when the specified alarm is processed relative to other alarms. Tips and notes: – Generally, 15 is considered most severe when specifying alarm severity. – The default value is 0. • Predicted PV Low – Trip Point (PREDPVLOALM.TP) – For PID-PL blocks, at every execution period, Profit Loop PKS estimates the

Configuration Tab	Description
	steady-state value for both the PV and OP. The predicted PV low alarm can be used to indicate when the predicted steady-state PV value drops below a particular value (entered on the PID-PL Block Parameters - Algorithm form) for more than a specified time. – Enter a value greater than or equal to zero, to enable the predicted PV low alarm trip point. – Enter NaN (Not a Number) to disable the predicted PV low alarm. • Predicted PV Low – Priority (PREDPVLOALM.PR) – Select the appropriate priority level for the corresponding alarm type based on the following categories: – NONE: The alarm is not reported to the system and is not annunciated. – JOURNAL: The alarm is written to the event file, where it can be accessed for alarm and event reports, or event displays on a Station. JOURNAL priority alarms do not appear in the Alarm Summary display. – LOW, HIGH, URGENT: These selections cause the alarm to be displayed in the alarm zone at the bottom of a Station display to bring new alarms to the attention of operators and/or directed to an alarm/event printer. The alarm also appears in the Alarm Summary display, in the color specified here: 1. LOW: Low-intensity yellow 2. HIGH: Yellow 3. URGENT: Red – Note that the default selection is LOW, and that alarm levels are prioritized as follows: JOURNAL --> LOW --> HIGH --> URGENT (highest priority) • Predicted PV Low – Severity (PREDPVLOALM.SV) – A relative severity to the predicted PV low alarm can be assigned by specifying a weight in the range of 0 to 15. The specified weight determines when the specified alarm is processed relative to other alarms. Tips and notes: – Generally, 15 is considered most severe when specifying alarm severity. – The default value is 0.

Configuration Tab	Description
	• Bad PV Shed Time (Sec) (BADPVALM.TP) – If the PID-PL block PV falls out of range (PVSTS changes to Bad), you can specify an amount of time (greater than zero, in seconds) that the block will be allowed to run before a bad PV alarm is generated. In this case, Profit Loop PKS discontinues bias updating and operates strictly from its model predictions, for the specified amount of bad PV shed time. This approach offers additional flexibility not found with other regulatory control blocks (which, by necessity freeze their output and issue a bad control alarm). CAUTION! because the model prediction is no longer synchronized with the process, the prediction can wander; therefore control action should be stopped after some period of time. For details on remaining attributes on this tab, which are the same as those for the PID block, refer to either the online Help for Control Builder, or to *Reference Data for Functional Block Types*, *Regulatory Control Blocks*, PID Block in this document.
SCM	All attributes configured on this tab are the same as those for the PID block. Refer to either the online Help in Control Builder for details, or to *Reference Data for Functional Block Ty*pes, *Regulatory Control Blocks*, PID Block in this document.
Identification	Lets you view information pertinent to the qualfication and version control system and enter block comments, if desired.
Dependencies	Lets you view block hierarchical information.
Block Pins	Lets you select the available parameters that you want to expose as input/output pins on the function block graphic in Control Builder.
Configuration Parameters	Lets you select the available parameters that you want to appear on the face of the function block in the Project tab in Control Builder.
Monitoring Parameters	Lets you select the available parameters that you want to appear on the face of the function block in the Monitoring tab in Control Builder.
Block Preferences	Lets you change several block-viewing preferences including the color of the block's faceplate.
Template Defining	Lets you view and define parameters for associcated templates.

Configuration Tab	Description
Insertion Type	Lets you include an insertion type from a CAB instances in the block. See *CAB insertion configuration considerations for regulatory control blocks* for more information

Function

The Profit Loop PKS control algorithm belongs to a class of controllers known as "model predictive control." These controllers rely on a dynamic model to predict future movement in the process variable. If this predicted PV does not meet the control objectives (maintain at current setpoint), control action is taken to realign the PV with its objectives. In contrast, a PID controller uses past and current error trajectories to restore the PV to its SP within one control move, regardless of the long-term consequences of the move.

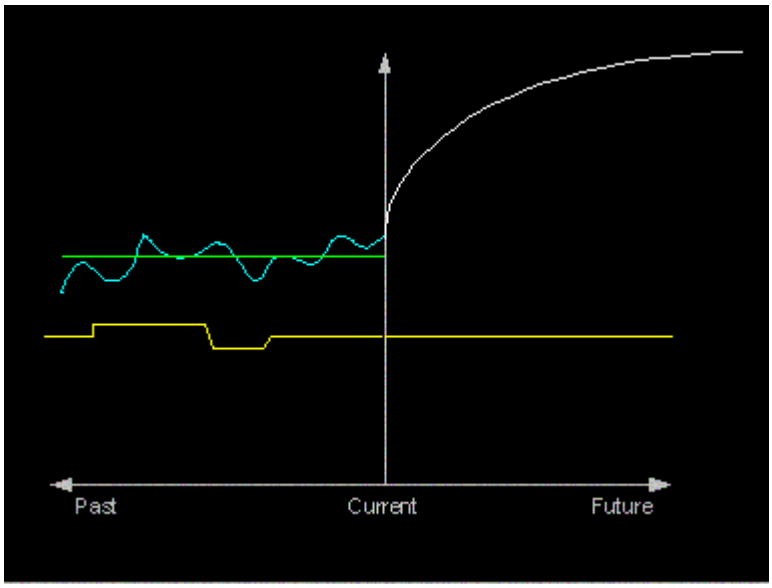
Additional functional details about Profit Loop PKS are provided in the following discussions:

- Model prediction
- Model biasing
- Control action
- Range control
- Optimization
- Predictive alarming
- External gain updating

Model prediction

The first step in any model predictive control scheme is to predict the future trajectory for the PV assuming no further movement in the OP. To make these predictions, Profit Loop PKS uses a dynamic model to relate past OP movement to future PV movement.

Model prediction is shown graphically in the following figure.



In the preceding figure, the white line represents the predicted future PV trajectory. (This is the line in the upper right quadrant of the graph.)

For Profit Loop PKS, the dynamic model is entered as a Laplace transfer function of the form:

$$G\frac{n_5 s^4 + n_4 s^3 + n_3 s^2 + n_2 s + n_1}{d_5 s^4 - d_4 s^3 + d_3 s^2 + d_2 s + d_1} e^{-Ts}$$

where
G is the process gain;
T is the deadtime;

n and d_1 are the process dynamics.

The Laplace variable, s, has units of min^{-1}, requiring dynamics and deadtimes to be entered in minutes.

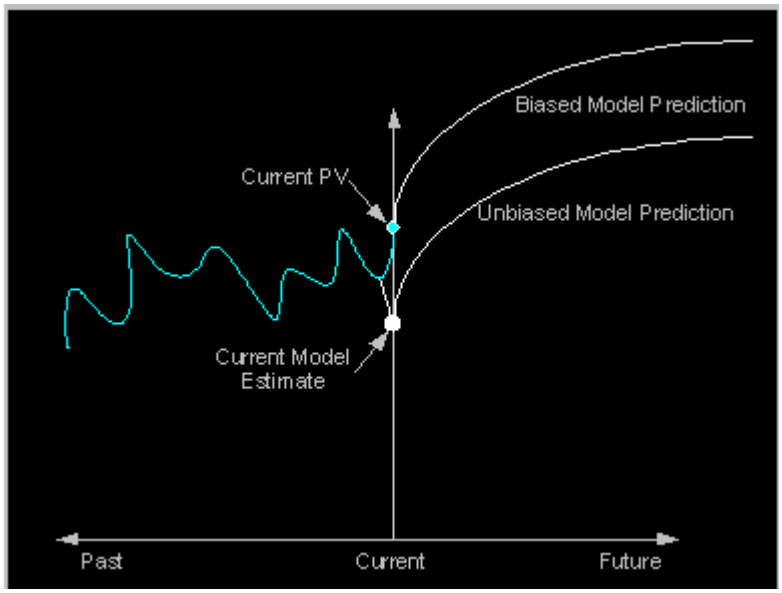
PID-PL buffers user entered model changes to prevent partial updating of the model. Model changes are not accepted on a running function block until the UPDATEMODEL flag has been set. This flag will automatically clear on completion of the update.

If you decide to abort the model update, the RESETMODEL flag can be set. This flag resets the user interface to the active model and clears itself.

Model biasing

In practice, there will always be a mismatch between the model prediction and the real process measurement. This difference can be attributed to a number of sources: inaccuracy of the model, measurement noise, external process disturbances, etc. If the model does not correct for this difference, the model prediction will slowly wander from the actual PV and the function block's integral action is lost.

To account for the mismatch, Profit Loop PKS compares its PV prediction for the current time to the current process measurement. The difference between these values is referred to as the bias and is added to the future PV trajectory.



In practice, the above bias is susceptible to high frequency signal noise, which could ultimately lead to excessive control action. To eliminate the noise effect, Profit Loop filters the raw bias applying the filtered value to future PV trajectories.

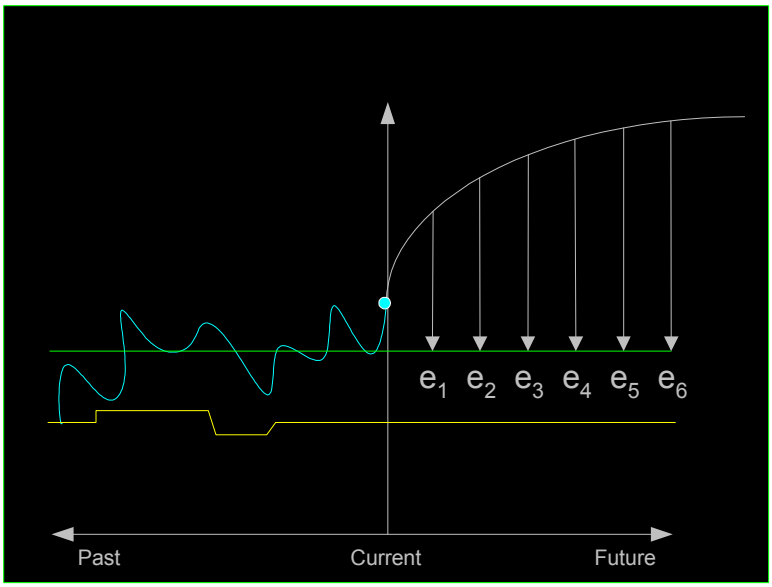
For especially noisy measurements, Profit Loop PKS employs a proprietary noise reduction filter to the bias. Unlike a simple PV filter, this filter eliminates measurement noise but reacts quickly to persistent external disturbances.

MODELPV is the value the current model prediction without any biasing. Under normal circumstances changes in the unbiased model prediction should track actual process changes (both in magnitude and time). If there is a significant difference between the responses, the process model should be updated using the Profit Loop Assistant tools.

To simplify model validation, a reset bias button is available on the detailed displays. This button sets MODELPV equal to the current PV, for easy monitoring of further changes in their values.

Control action

Once a future trajectory has been calculated, control action is implemented to force this process trajectory toward its control objective. Typically, this involves the calculation of the control actions necessary to bring the process variable to its setpoint over the course of the trajectory. Alternatively, the controller minimizes future errors.

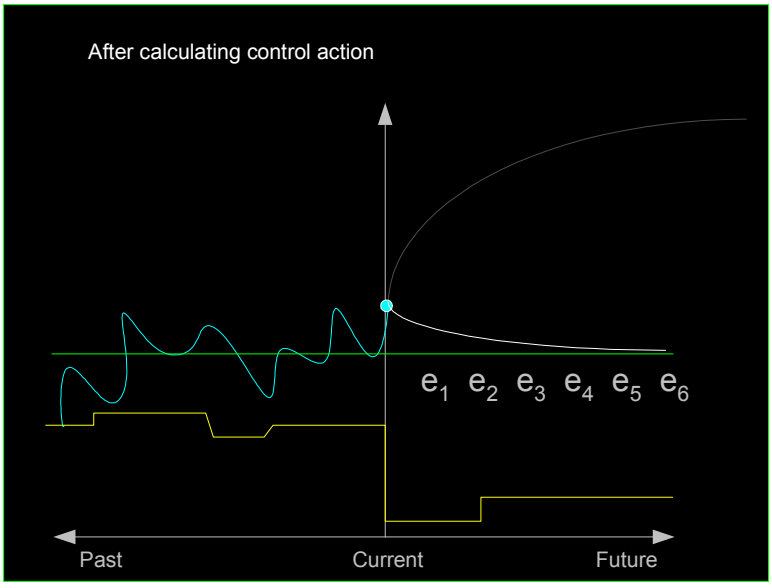


While the exact calculation of the control action is too complicated for this document, it involves the inversion of the process model thus relating error to control action. If the error is zero (the control objective is met), no further control action is required in this execution cycle, and the OP is unchanged.

When minimizing future errors, there are several ways to define optimality. For instance, one algorithm may weight more heavily the initial errors, a second the final error, and a

third equally weight all errors. Using different optimality criteria leads to different solutions.

Profit Loop PKS focuses on the later part of the trajectory from the closed-loop response time onward. It then determines the minimum control action necessary to bring the process variable to its setpoint (and keep it there) before the user-specified closed-loop response time. Because this algorithm uses the minimum energy to meet its control objective, it is more robust to inaccuracies in the model than other model predictive control algorithms.



To tune the function block, you must specify the closed-loop response time. To make tuning simpler, this response time is normalized by the open loop response time, and the ratio of the response times, the performance ratio is entered. A performance ratio of 1 indicates that the function block should bring the process to its setpoint value in approximately open loop response time minutes. This function block action is similar to the response from a steady-state-only controller.

When the performance ratio is less than 1, more aggressive control is required as the process is driven to its setpoint faster than its natural (open loop) response. In contrast, a performance ratio greater than 1 generates laxer control but is more robust to modeling errors.

When the process is not self-correcting (contains an integrator), the open loop response time is not defined, and consequently, the use of a performance ratio is meaningless. Under these circumstances, you enter the closed-loop response time directly, and the performance ratio is reset to 1.

Range control

With the above algorithm, it is possible to control the PV within a user-entered range instead of to a hard target (SP). Errors then represent the deviation of the future trajectory outside of the operating range. If the future trajectory lies inside the operating range, there is no error.

Conceptually, Profit Loop's range control option is PID gap control with a gap gain factor of 0. However, unlike gap control, Profit Loop PKS considers the long-term process response and not the current process value, applying control action only when the projected PV trajectory is out of bounds.

The parameters SPHI and SPLO define the operating range for the gap, in the same engineering units as SP. This block ensures these values do not exceed the absolute SP range limits, SPHILM and SPLOLM, capping the operating range limit when required.

Furthermore, if the input SP value is outside the range specified by SPHI and SPLO, this function block clamps SP to the appropriate limit. This additional restriction on SP only applies when the range control option is selected. Otherwise, the SPHI and SPLO parameters track the current setpoint, SP.

The user-entered parameter, CTRLMODE, indicates whether Profit Loop PKS controls within a range, (RANGE), or to a setpoint (SETPOINT). The parameter has no meaning if PID control is selected.

Optimization

With range control, the steady-state operating conditions are allowed to float (within the range); there is no unique resting value. While this may be acceptable for dynamic control, it may not be suitable for planning and long-term operations.

To define the steady-state operations, Profit Loop PKS includes a small optimizer that allows you to specify the desired steady-state operating conditions. Depending on the control objectives (as specified by the OPTMODE parameter), you can minimize the process variable (MINIMIZE), maximize this variable (MAXIMIZE), aim to a user-entered target (OPTTARGET), or aim toward a narrower PV range but not to a unique value (DUALRANGE). Furthermore, you can disable this option completely (NONE).

The rate at which the process approaches steady-state operations is typically slower than the rate at which dynamic constraints are resolved. This allows Profit Loop PKS to be configured for two control objectives – quick resolution of dynamic errors with a slower approach to optimal operations.

The optimization rate is normalized by the closed-loop response time and specified by the parameter, OPTSPEED:

$$TimeToOptimal = \frac{6 \times CloseLoopResponseTime}{OptSpeed}$$

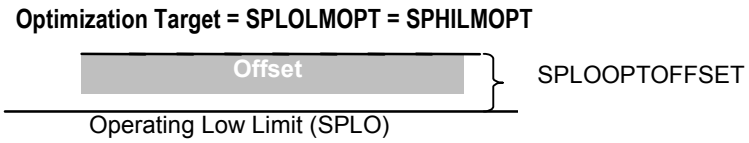
When OPTSPEED = 6, the optimizer tries to bring the process to its steady-state value by the end its control horizon. When OPTSPEED = 3, the optimizer is twice as slow; when 12, twice as fast. The default value for OPTSPEED is 2.

If the optimizer is allowed to force the process against one of its operating limits, there will be times when the optimizer overshoots the limit causing dynamic control action in the opposite direction. If left unchecked, this can lead to process oscillations around the optimal value.

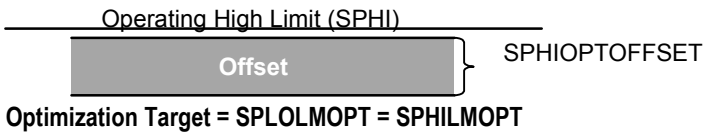
To circumvent this problem, the optimizer is restricted to a narrower range than the controller. User specified offsets dictate how narrow the optimization range is. For all optimization modes except OPTTARGET, Profit Loop PKS calculates the optimization limits as deviations from the function block high and low operating range (SPHI and SPLO).

The following figures illustrate how Profit Loop PKS uses optimizer offsets to set high and low optimization limits.

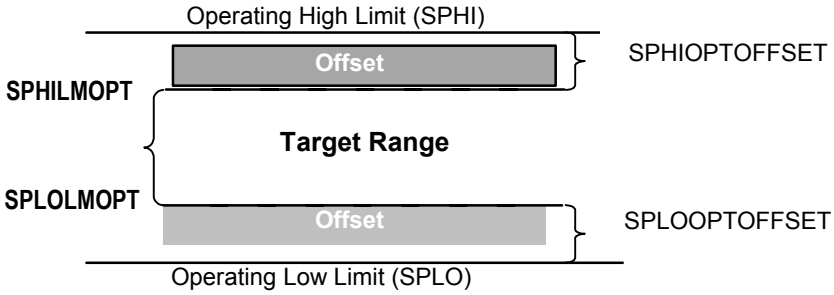
Optimization target for Minimization



Optimization Target for Maximization

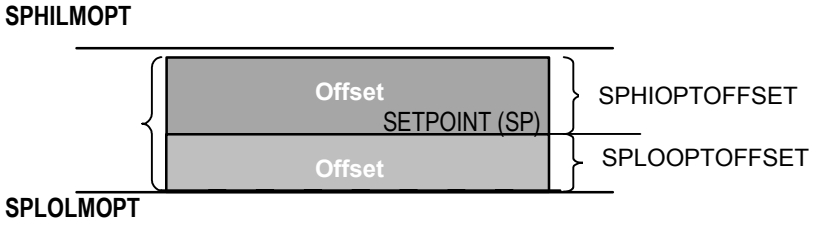


Optimization Target for Dual Range



For OPTTARGET optimization, Profit Loop PKS calculates the optimization limits as deviations from setpoint (SP). Using non-zero optimizer offsets is equivalent to DUAL RANGE optimization but with the optimization range tracking any setpoint changes. The following figure illustrates this situation.

Optimization Target for OptTarget



Similar offsets, OPLOOPTOFFSET and OPHIOPTOFFSET, limit the OP range, OPLOLM and OPHILM, for optimization.

Predictive alarming

At every execution period, the Profit Loop PKS algorithm estimates the steady-state value for both the PV and OP (STEADYSTATEPV and STEADYSTATEOP). This function block may be configured to generate an alarm when the predicted steady-state PV exceeds a user-specified trip point (PREDPVHIALM for high PV predictions and PREDPVLOALM for low predictions). Both alarms are analog alarms and, as such, support individual trip points, priorities, and severities.

External gain updating

PID-PL supports the external updating of the process model gain (gain scheduling). This is similar to the EXT gain option of a PID function block, except that the process model gain (and not the PID function block gain) is altered. You can use this option to compensate for nonlinear process gain changes. For example, if you are controlling the level of a tank whose cross-section is not constant, you can use this option to modify the model gain compensating for the nonlinear rate of level change caused by the changing shape of the tank.

The PID-PL block supports two methods for updating the gain:

- An operator or another function block changes a process gain multiplier, PROCGAINMULT. The actual process gain (active gain) is then computed as shown in the following equation:

$$ProcGainAct = ProcGain \times ProcGainMult$$

- Because the gain multiplier is restricted to be greater than 0, the process model gain never changes direction.
- A user program or another function block changes the active value, PROCGAINACT, directly. This value is restricted to be non-zero, to prevent division by zero, but may otherwise change directions.

Similar to gain scheduling, the process deadtime can be altered through an external input. Use this option to compensate for nonlinear process delay changes. Two methods of deadtime updating are supported:

- An operator or another function block changes a process deadtime bias, PROCDEADTIMEBIAS. The actual process delay (active deadtime) is then computed as:

$$ProcDeadtimeAct = ProcDeadtime + ProcDeadtimeBias$$

- A user program or another function block changes the active value, PROCDEADTIMEACT, directly. To ensure safe operation, the active deadtime is restricted between 0% and 120% of the engineer-entered deadtime, PROCDEADTIME.

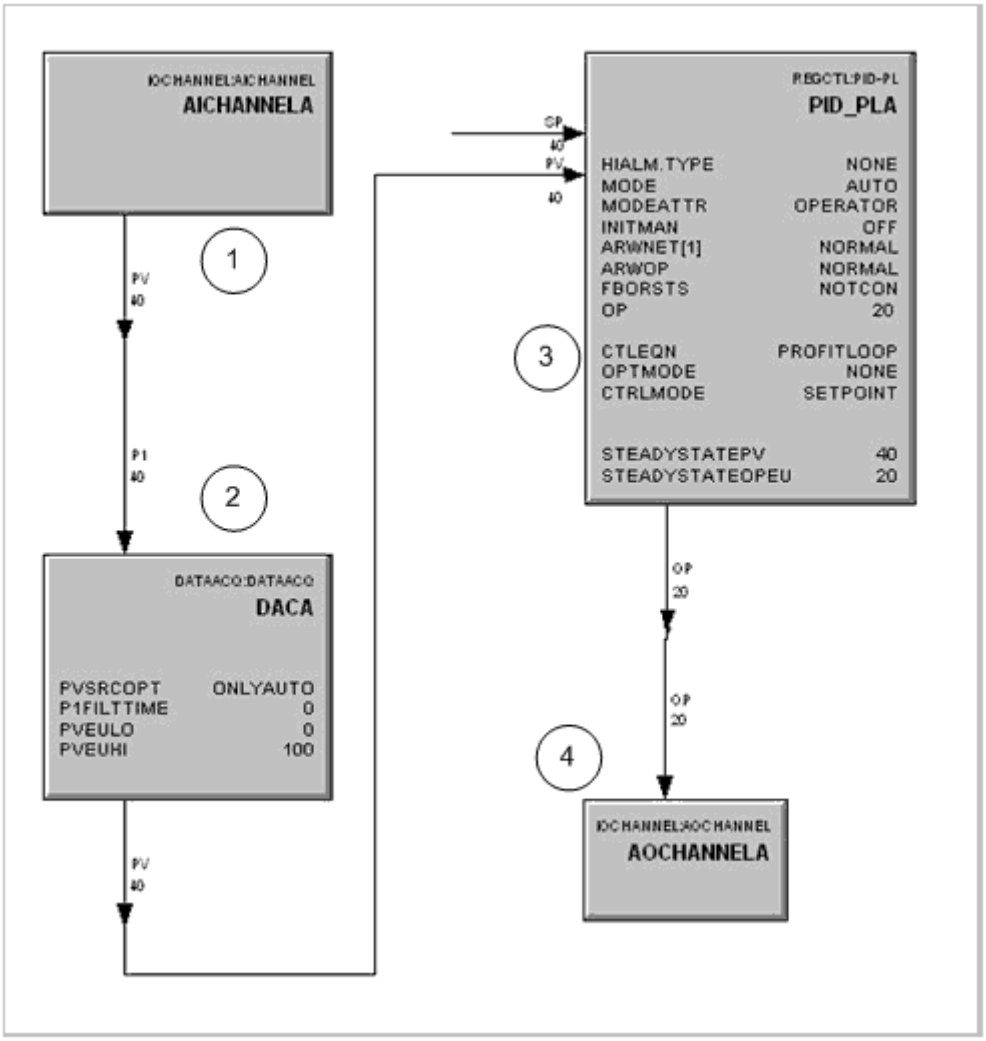
Configuration examples

Four examples are discussed in the following paragraphs:

- Setpoint control
- Range control
- Discrete analyzer
- Gain scheduling

Setpoint control

The following figure and table show a sample configuration that uses a PID-PL block to form a single control loop set for setpoint control. Note that this configuration is nearly identical to the single-loop PID configuration.

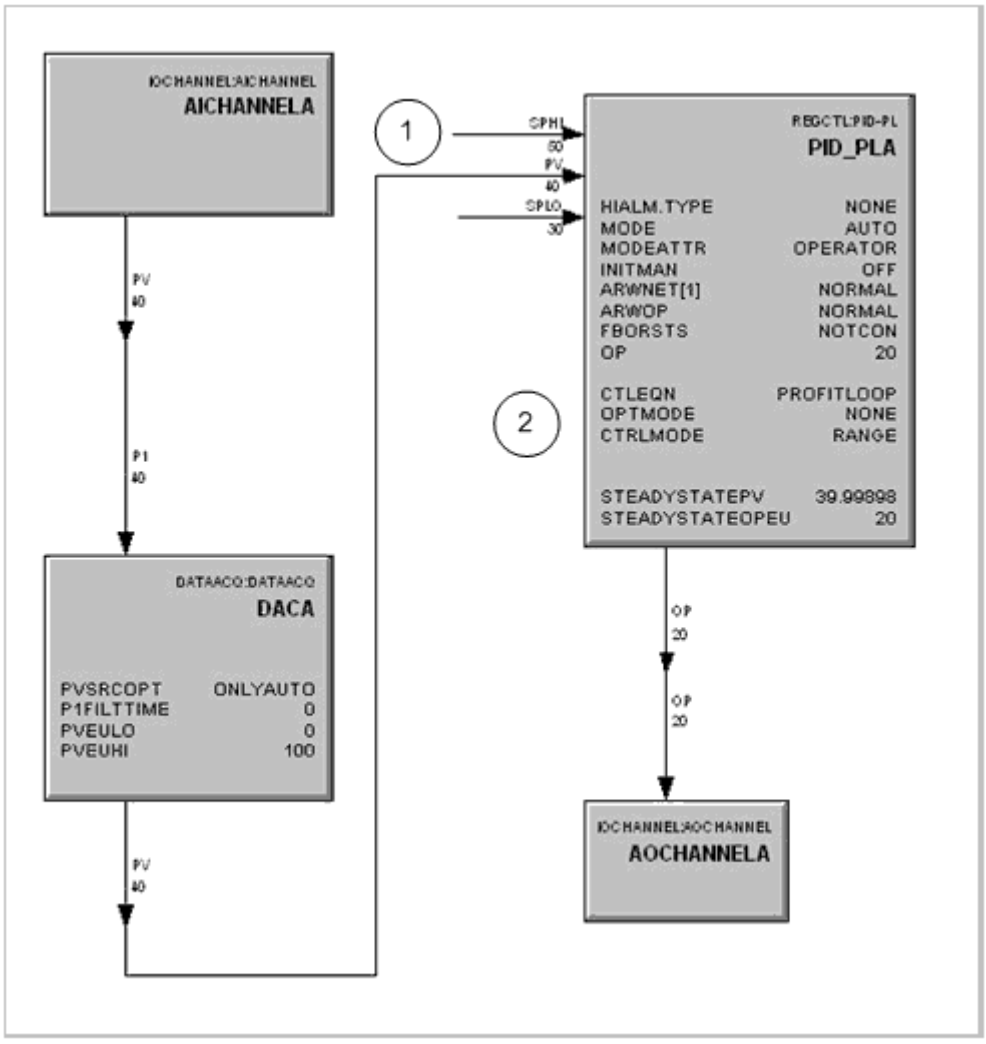


The following table explains the callouts in the preceding figure.

Callout	Description
1	Use the PV parameter connection to carry data from the analog input to the data acquisition block.
2	Use the data acquisition block to filter and scale the PV input. This block also maintains several PV alarms.
3	Use the PID-PL block for control. A separate SP block pin is exposed for on-line parameter changes. To implement Profit Loop PKS control, the control equation is set to PROFITLOOP and control mode to SETPOINT.
4	Use the OP connection to transfer control block output to analog output.

Range control

The following figure and table show a sample configuration that uses a PID-PL block to form a single control loop set for range control.

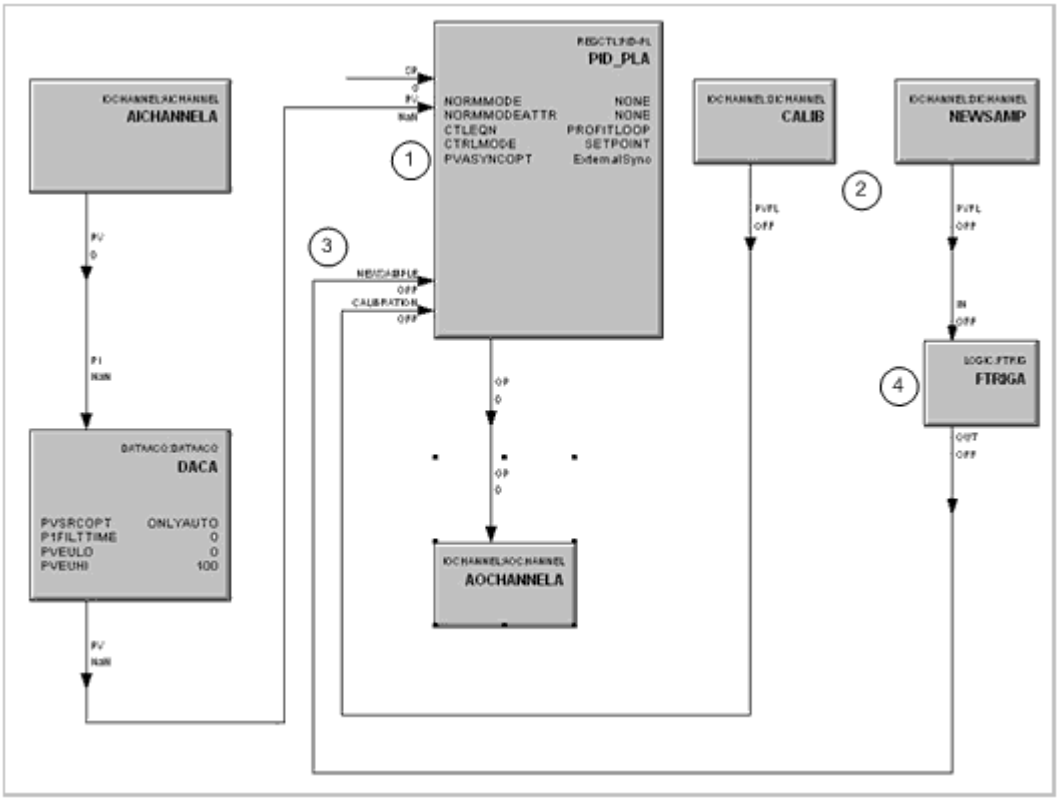


The following table explains the callouts in the preceding table.

Callout	Description
1	Configuration is nearly identical to setpoint control configuration. Use block pins SPHI and SPLO to allow on-line change of the control range. SP is not used in this case.
2	CTRLMODE is set to RANGE.

Discrete analyzer

The following figure and table show a sample configuration for interfacing a PID-PL block to a discrete analyzer. In this example, the analyzer maintains two digital signals—an on-off flag to indicate when the analyzer is in calibration, and a 30-second pulse signal whenever a new measurement is available.



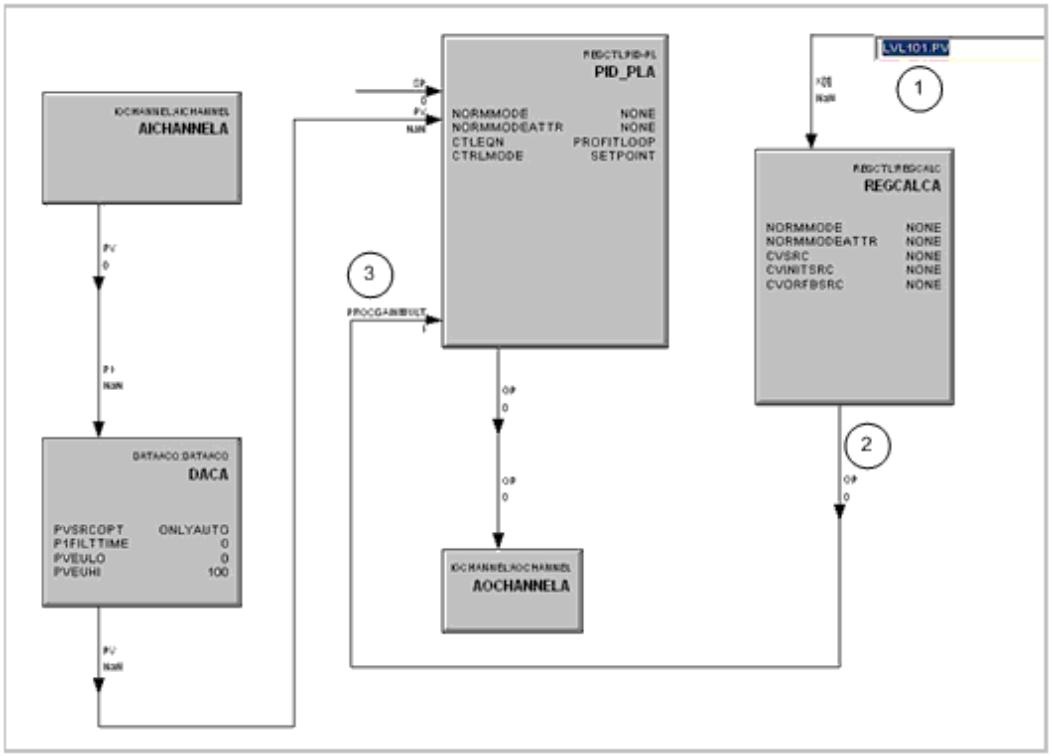
The following table explains the callouts in the preceding figure.

Callout	Description
1	Because the analyzer maintains a set of control signals, PVASYNCOPT is set to ExternalSync to force bias updating only when the analyzer indicates that a new reading is available.

Callout	Description
2	The analyzer signals are accessed through two digital-input channels.
3	The calibration flag runs directly to the calibration pin on the PID-PL block. As long as this signal is ON, the analyzer is presumed to be calibrating. The new sample input is buffered by the FTRIG block before it links to the new sample pin on the PID-PL block.
4	FTRIG captures the leading edge of the new sample pulse and sets the new sample flag on the PID-PL block. The new sample parameter remains ON for one execution period before it is reset by FTRIG.

Gain scheduling

The following figure and table show a sample configuration that uses gain scheduling with PID-PL. In this example, the gain of a level controller depends on the level in the tank. This may occur if the geometry of the tank changes as the level raises and falls.



The following table explains the callouts in the preceding figure.

Callout	Description
1	For clarity, the tank level is shown as a separate independent block connection. In practice, the input to the regulatory control block may be connected to the output of the data acquisition block.
2	A regulatory control algorithm is employed to calculate an appropriate base gain multiplier based on the geometry of the tank and the level in the tank.
3	The gain multiplier is connected to the PROCGAINMULT parameter on the PID-PL block. This input requires the gain multiplier to be between 0.01 and 100. An alternative approach connects a raw gain to the PROCGAINACT parameter (not shown). This approach circumvents the range restrictions on the gain multiplier.

Operating modes and mode handling

The PID-PL block operates in the following modes:

- **MAN** (MANual) – If mode is MANual, OP may be stored by the operator or a user program; PV and SP are ignored. If a primary exists, it goes to the initialized state.
- **AUTO** (AUTOmatic) – If mode is AUTOmatic, SP (or SPP) may be stored by the operator or a user program; if a primary exists, it goes to the initialized state. SP contains the setpoint value in engineering units and SPP contains the value in percent.
- **CAS** (CAScade) – If mode is CAScade, SP is pulled from a primary; if the primary is off-control (that is, inactive or initializing) or the connection is bad, this block invokes timeout processing.

These modes apply regardless of whether the control equation is set for PID control or Profit Loop PKS.

In addition to the block mode, Profit Loop PKS operates in the following control modes:

- SETPOINT – Controller alters OP so PV tracks the SP.
- RANGE – Controller alters OP so PV is constrained between SPLO and SPHI.

If range control is employed, Profit Loop PKS uses the optimization mode to determine the desired steady-state operating conditions:

- NONE – PV is allowed to freely float between SPHI and SPLO. If projected to remain in range, no additional control action is taken.
- MINIMIZE – At steady-state, PV is minimized.
- MAXIMIZE – At steady-state, PV is maximized.
- OPTTARGET (OPTimize to TARGET) – At steady-state, PV equals SP.
- DUALRANGE – At steady-state, PV is constrained between SPLOLMOPT and SPHILMOPT, but may float anywhere within this optimization range. The optimization range is more constraining than the control range, SPLO to SPHI.

Required inputs

The PID-PL block requires two inputs: PV and SP. See Required inputs for the **PID** block for details on these inputs.

For range control, the PID-PL block requires two additional inputs: SPHI and SPLO. These parameters cannot be initialized. They can be pulled from another block, set through operator entry, or stored by a user program.

Input ranges and limits

For PID or Profit Loop PKS setpoint control, the PID range limits apply. See *Input ranges and limits* for the **PID** block for details.

For range control,

- PVEUHI and PVEULO define the engineering unit range for MODELPV, SP, SPHI, and SPLO.
- SPHILM and SPLOLM define the operating limits in engineering units for SPHI, and SPLO. If a value is entered outside these limits but within the engineering unit range, this block clamps the value at its appropriate limit. Otherwise, an out of bounds error is generated.
- SPHI and SPLO define the operating limits in engineering units for SP. The operator is prevented from storing a setpoint value that is outside these limits. If the primary or a user program attempts to store a value outside of the limits, this block clamps it to the appropriate limit and sets the primary's windup status.

Initializable outputs

The PID-PL block supports a single initializable output. Like PID, this calculated output can be either in percent, OP, or in engineering units, OPEU. See Initializable outputs for the **PID** block for more details.

Control initialization

The PID-PL block brings initialization requests from its secondary through the hidden BACKCALC connection. In addition, the secondary may propagate one-shot initialization requests to this block.

- SECINITOPT may be used to ignore initialization requests from the secondary.
- If the secondary is requesting initialization, the PID-PL block:
  - initializes its output
    CV = initialization value from the secondary.
  - uses the initialized CV value in model predictions
  - sets initialization request parameters for its primary
    INITREQ = On
    INITVAL = SP
- If a PID-PL block configured for range control is cascaded to a primary and OPTMODE does not equal OPTTARGET, then the primary has no bearing on the operation of its secondary. Under these circumstances, an initialization request is sent to the primary.

Output bias

Profit Loop PKS does not support an output bias. For details on the biasing of PID algorithms, see *Output bias* for the **PID** block.

Output ranges and limits

The output range of a PID-PL block is identical to the **PID** block. See Output ranges and limits for the **PID** block for details.

Output limit behavior for PID-PL block recovering from Bad Control

The output limit behavior of a PID-PL block is identical to the **PID** block. See the Output limit behavior. . . for the **PID** block for details.

Direct or reverse control

A PID-PL block may be configured for direct-control action or reverse-control action, effectively changing the sign of the controller gain. See *Direct or reverse control* for the **PID** block for implementation details.

With Profit Loop PKS, the sign of the controller gain is determined by the sign of the process model gain. To prevent mismatch between the model gain and control action,

changes to CTLACTN are not allowed. Furthermore, when the model gain changes (including initial loading of the function block), CTLACTN is set to match the new gain direction.

- A positive process model gain leads to reverse control action.
- A negative process model gain leads to direct control action.

Set Point Ramping

The Set Point Ramping option lets you ramp from the current setpoint value to a target setpoint value. For Profit Loop PKS, this option is applicable to setpoint control and range control when OPTMODE is OPTTARGET only. See Set Point Ramping for the **PID** block for implementation details.

Range Control Ramping

For range control, range control limit ramping provides a smooth transition from the current SPHI and SPLO limits to newly entered values by ramping range limit changes over a period of time. The SPHI and SPLO ramp rates are independent and specified through the parameters, SPHIRAMPRATE and SPLORAMPRATE (engineering units / min). Entering NaN provides immediate ramping to the new limit, and consequently, disables this option.

Mathematically, limit ramping can be represented as:

$$SPHIACTIVE = \max(SPHIACTIVE - SPHIRAMPRATE * \text{Execution Period}, SPHI)$$

$$SPLOACTIVE = \min(SPLOACTIVE + SPLORAMPRATE * \text{Execution Period}, SPLO)$$

where
SPHIACTIVE and SPLOACTIVE represent the effective range limits at any given execution period.

SP limit ramping only applies when the operating range becomes more restrictive (Newly entered SPHI < Current SPHI or newly entered SPLO > Current SPLO). Otherwise, the range limit change is immediately applied.

When the current range limit is far from its newly entered value, it may take a considerable amount of time before the active value reaches its new limit. This is an unnecessary delay if the process is operating well within the new operating range. To avoid this delay, Profit Loop employs a proprietary algorithm to bring the active value “close” to the newly entered limit before it starts ramping. This one shot correction avoids bumping the process, yet applies ramping in a more meaningful fashion.

Furthermore, this feature eliminates problems associated with fat fingering (accidentally depressing a key twice, e.g. entering 500 instead of 50) a range limit. For example, consider an SPHI change from 99 to 100. If an operator enters 1000 instead of 100, the SPHIACTIVE immediately moves to 1000. If the operator re-enters the limit as 100, there may be a considerable delay before SPHIACTIVE ramps from 1000 back to 100. One shot correction eliminates most of this delay.

PV tracking

The PV Tracking option sets SP equal to PV when automatic control is disabled (e.g. cascade is broken, mode is manual). Having SP track PV avoids output bumps when automatic control is reestablished. See *PV tracking* for the **PID** block for implementation details.

For range control, SPHI and SPLO also track out of range PV values when automatic control is disabled and PV tracking is enabled. The following rules apply:

- If SPLO is greater than SP then set SPLO equal to SP
- If SPHI is less than SP then set SPHI equal to SP

ATTENTION

- SPHI and SPLO track the current SP (and only indirectly the PV).
- Range limits are sensitive to transient changes in the SP/PV. Once a range limit is relaxed, it is not re-constricted. However, in contrast with setpoint control, an operator can alter the SPHI or SPLO limit, subject to the above limitations, while PV tracking is active.
- The PVTRAKOPTAI parameter provides a PV Tracking option for the specific case when the block is in Auto mode and it is undergoing initialization (INITMAN is On). The PV Tracking option parameter (PVTRAKOPT) does not include the Auto/Init case, but does cover several other situations as noted above.

PV Bad Value Handling

When the PV status (PVSTS) is BAD, Profit Loop PKS discontinues bias updating and operates strictly off of its model predictions. This offers additional flexibility not found with other regulatory control blocks that, by necessity, freeze their output and issue a BadCtl alarm. However, because the model prediction is no longer synchronized with the process, the prediction can wander and, generally, control action should be stopped after some user specified period of time.

For Profit Loop PKS, the PID-PL block supports a Bad PV duration alarm. This alarm trips if the PV is BAD for longer than the user-entered duration. Tripping of this alarm leads to bad control (BadCtl).

For PID control, the Bad PV duration is reset to 0 tripping on the first detection of a bad PV.

ATTENTION

The Bad PV duration alarm is in addition to any other Bad PV alarms, specifically the Bad PV alarm in the DATAACQ block.

PV Calibration

On occasion, one may wish to intentionally disable the PV measurement without inactivating the control module or placing the function block in MAN. This is most evident when the PV's sensor is recalibrated and such recalibration causes undesirable control action. A calibration flag, CALIBRATION, is provided for this purpose.

When CALIBRATION is set, Profit Loop PKS:

- Disables updating of bias between model prediction and PV
- Continues control action based on model prediction only

Asynchronous PV Inputs

Many specialty process sensors are discontinuous, generating results on an infrequent basis relative to the control execution period. This is especially true among gas chromatographs where a new reading is generated every 2-40 minutes.

When these sensors are used in PID control, control error continues to integrate between sensor updates. This may lead to unnecessary overshoot in the control action and oscillation in the PV response.

With Profit Loop PKS, the PID-PL block updates only when a new sensor reading is available. This has the effect of disabling integration between sensor readings eliminating function block overshoot. Between sensor readings, Profit Loop

- Disables updating of bias between model prediction and PV
- Continues control action based on model prediction only

Profit Loop uses the parameter PVASYNCOPT to specify when and how a PV is updated. When PVASYNCOPT set to:

- **Continuous** – PV (and PV bias) is updated every execution cycles regardless of its value. This is the correct setting when the sensor involved generates a continuously updated signal, such as a thermocouple or strain gauge.
- **OnPvChange** – PV bias is updated only when there is a change in the PV value from one execution period to the next.
  - Note: Avoid this setting if noise alters the PV value between analyzer updates.
- **External Sync** – PV bias is updated when an external flag indicates a new analyzer reading. This external flag should be connected to the Boolean parameter, NEWSAMPLE.

ATTENTION

- Logic blocks should be employed to set and reset the NEWSAMPLE flag. It is the users responsibility to design and construct the appropriate logic.
- The PID-PL block makes no attempt to set or reset this flag.
- It is a good practice to buffer the external update signal with an FTRIG (RTRIG) block. This block sets the NEWSAMPLE flag on leading edge (lagging edge) change in the external signal and automatically resets the NEWSAMPLE flag after one execution period.

Control Equations

The PID-PL block supports the standard 5 PID equations (EqA through EqE) found on a **PID** block as well as the Profit Loop equation. You specify which equation to use through the CTLEQN parameter.

To provide expansion for future PID type equations, the Profit Loop option is shown at the bottom of the CTLEQN list after options EqF and EqG. EqF and EqG are provided for possible future expansion of the PID equation set. If you select either of these options, an error message is displayed indicating an invalid entry.

With standard PID-type function blocks, the control equation can only be altered while the control module is inactive, preventing algorithm initialization errors. This requirement is too restrictive for Profit Loop, which maintains a running initialization value regardless of the mode of operation. Consequently, PID-PL supports the hot swap from a PID equation to Profit Loop on an active control module. Hot swapping is supported in one direction only; to revert back to PID control, user must first inactive the control module.

For more information on PID equations, see *PID equations* for the **PID** block.

Tuning Constant Change Considerations

WARNING

You cannot always reverse output (OP) resulting from changes you make to a tuning constant gain (K), integral time (T1) or derivative time (T2) in an online control loop.

You cannot undo a change in a tuning constant in an online control loop by simply changing the constant back to its original value. The output (OP) does not jump back to its original prior value just because you return the constant to its prior value. In this case, you must put the loop in MANUAL mode and set the output (OP) to the desired value before returning the loop to AUTO mode.

Timeout monitoring

The PID-PL block monitors for communication timeouts between primary and secondary controllers of a cascade pair. This block uses the same methodology as the **PID** block. See *Timeout monitoring* for the **PID** block for implementation details.

Windup handling

Profit Loop PKS maintains integral action by updating a bias between its model prediction and the current process measurement, PV. If the function block is in windup, the input to the process model may not represent what was actually implemented by the control loop – causing an additional bias.

To prevent this additional bias, Profit Loop PKS actively unwinds a secondary control module in windup. To do this, the primary's OP is slowly moved in the direction to relieve windup. When the primary reaches this value, the secondary function block will be at the edge of its windup condition – oscillating in and out of windup.

To force the OP to unwind and to prevent OP moves that exacerbate windup, Profit Loop temporarily adjusts its internal OP operating limits. These adjusted limits are displayed in the OPHIACTIVE and OPLOACTIVE parameters. Ordinarily, these limits are identical to the OPHILM and OPLOLM, respectively.

Windup processing

Every regulatory control type block maintains anti-reset windup status for its output (ARWOP) and each of its initializable inputs (ARWNET). The following table lists the posssible values for ARWOP and ARWNET parameters.

If the Value is . . .	Then, the Associated Parameter . . .
Normal	is free to move in either direction.
Hi	is at its high limit and it may only be lowered.
Lo	is at its low limit and it may only be raised.
HiLo	may not move in either direction.

Manual Mode Interaction

When the MODE of a regulatory control block is changed to Manual (Man), the block sets its windup status (ARWNET) to HiLo. This means that every block upstream in a cascade strategy will set its windup status (ARWNET and ARWOP) to HiLo.

ARWOP computation

The ARWOP indicates if the output (OP) can be raised or lowered. The PID-type function blocks use ARWOP to restrict integral control. When ARWOP contains a value other than **Normal**, the PID block stops integral control in the windup direction. Integral control continues in the other direction, as does proportional and derivative control. But, windup status has no impact on proportional and derivative control.

If a function block has a secondary, it fetches the secondary's windup status and recomputes its ARWOP. The conditions within the function block, such as output being at its high limit, also affect the ARWOP. The ARWOP is computed as follows, assuming the block has only one output or that it is not a FANOUT block.

If Any of the Following are True . . .	Then, ARWOP Equals . . .
This block is inactive.	**HiLo**
A secondary exists but this block cannot fetch secondary data from it (communications or configuration error).	
A secondary exists and its windup state equals HiLo	
This block is in initialization (INITMAN = On).	
A secondary exists and it is requesting this block to initialize.	

If Any of the Following are True . . .	Then, ARWOP Equals . . .
A secondary exists and its windup state equals Hi.	**Hi**
This block's output is at its high limit (OPHIFL = On).	
A secondary exists and its windup state equals Lo.	**Lo**
This block's output is at its low limit (OPLOFL = On).	

ARWNET computation

When ARWNET is HiLo, stores to SP are not limited, rather this is the status propagated to the primary. The only *limiting* anti-reset windup status ever does is to stop integral action in one or both directions on PID blocks. For any other regulatory control type block, ARWNET is not used for any kind of limiting. The ARWNET is computed as follows, assuming the block has only one output or that it is not a FANOUT block.

If Any of the Following are True . . .	Then, ARWNET Equals . . .
This block is inactive.	**HiLo**
The ARWOP equals HiLo.	
This block is in Manual mode (MODE = Man)	
The calculated value (CV) range (CVEUHI / CVEULO) is NaN.	
The CV is NaN	
This block is connected to a non-initializable primary	
The ARWOP equals Hi (Pid function blocks have a configurable Control Action option (CTLACTN). If CTLACTN = Reverse , ARWNET will track ARWOP; but if CTLACTN = Direct , ARWNET will be the opposite of ARWOP.)	**Hi**

If Any of the Following are True . . .	Then, ARWNET Equals . . .
The input from the primary is at a high limit. For example, SPHIFL = On.	
This block's output has reached its positive rate-of-change limit (OPROCPOSFL = On)	
The ARWOP equals Lo (Pid function blocks have a configurable Control Action option (CTLACTN). If CTLACTN = Reverse , ARWNET will track ARWOP; but if CTLACTN = Direct , ARWNET will be the opposite of ARWOP.)	**LO**
Input from the primary is at a low limit. For example, SPLO.FL = On.	
This block's output has reached its negative rate-of-change limit (OPROCNEGFL = On)	

Anti-Reset Windup Status

The anti-reset windup network in (ARWNETIN) and anti-reset windup output in (ARWOPIN) parameters are added in the standard anti-reset windup (ARW) computation logic. They are user configurable and allow stores from Sequential Control Modules (SCMs) and control algorithm block (CAB) programs.

The ARWNETIN and ARWOPIN parameters would be *ORed* into the existing standard logic so it is not lost. The following table summarizes the influence the ARWNETIN and ARWOPIN parameters have on the ARWNET and ARWOP parameters, which are not user configurable.

ARWNETIN or ARWOPIN Parameter Is. . .	Standard Computation Logic Is . . .	ARWNET or ARWOP Parameter Is . . .
NORMAL	NORMAL	NORMAL
NORMAL	HI	HI
NORMAL	LO	LO
NORMAL	HILO	HILO

ARWNETIN or ARWOPIN Parameter Is. . .	Standard Computation Logic Is . . .	ARWNET or ARWOP Parameter Is . . .
HI	NORMAL	HI
HI	HI	HI
HI	LO	HILO
HI	HILO	HILO
LO	NORMAL	LO
LO	HI	HILO
LO	LO	LO
LO	HILO	HILO
HILO	NORMAL	HILO
HILO	HI	HILO
HILO	LO	HILO
HILO	HILO	HILO

Override feedback processing

If the PID-PL block is in a cascade strategy with a downstream OVRDSEL (Override Selector) block, it receives override feedback data. The data consists of an override status, override feedback value and an override offset flag. The status indicates if this block is in the selected or unselected strategy (as determined by the OVRDSEL block).

When the override status changes to unselected, Profit Loop PKS uses the override feedback value instead of the OP in its model predictions. In this manner, it prevents an anomalous bias due to unrealized control action.

For more details on override feedback processing, see *Override feedback processing* for the **PID** block.

PID-PL parameters

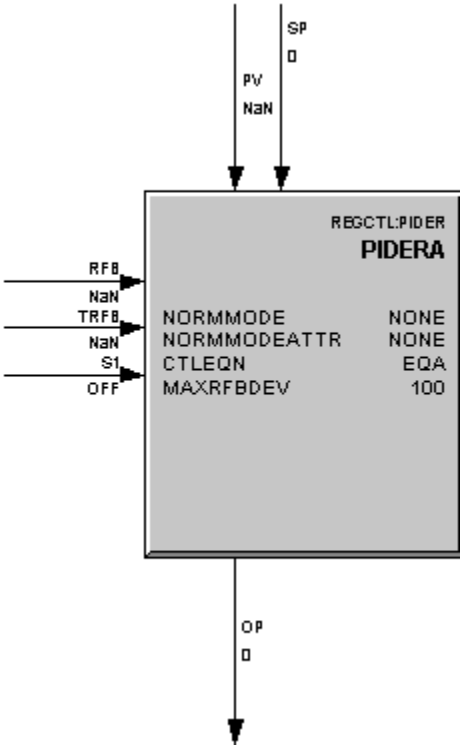
REFERENCE - INTERNAL

Refer to the *Control Builder Components Reference* for a complete list of the parameters used with the PID-PL block.

PIDER (PID with External Reset Feedback) Block

Description

The PIDER block is a regulatory control block that operates as a proportional-integral-derivative (PID) controller that accepts a reset feedback signal, a tracking value, and a tracking control switch. It supports the same Ideal form of calculating the PID terms as the PID block. It also prevents windup when the secondary does not propagate windup status or control initialization data back to the primary of a remote (foreign) controller. The PIDER block looks like this graphically:



The PIDER block accepts five analog inputs – a process variable (PV), a set point (SP), a reset feedback value (RFB), a tracking value (TRFB), and a tracking control switch (S1). The difference between PV and SP is the error and this block calculates a control output (OP) that should drive the error to zero.

The reset feedback (RFB) signal comes from the remote controller's PV, and the tracking value (TRFB) comes from its PV or SP. By monitoring the remote controller's PV and SP, the PIDER block can determine if the remote controller is responding. If the remote controller is not responding, it can prevent its own output from winding up.

The tracking control switch (S1) determines the output of the PIDER block. The S1 parameter is usually stored through an output connection from another function block, or by a user program. When S1 is Off, the PID control value output ($CV_{PID)}$ is combined with the reset feedback (RFB) value to obtain the control value (CV); and when it is On, CV is set equal to the tracking value (TRFB).

The following equations are supported:

- Proportional, Integral, and Derivative (PID) on the error
- Proportional and Integral (PI) on the error and Derivative (D) on changes in PV
- Integral (I) on the error and Proportional and Derivative (PD) on changes in PV
- Integral (I) only

The PIDER block may be used in a single control loop or with multiple PIDs in a cascade strategy.

Configuration example

The following illustration shows a PIDER block configured in a cascade control strategy.

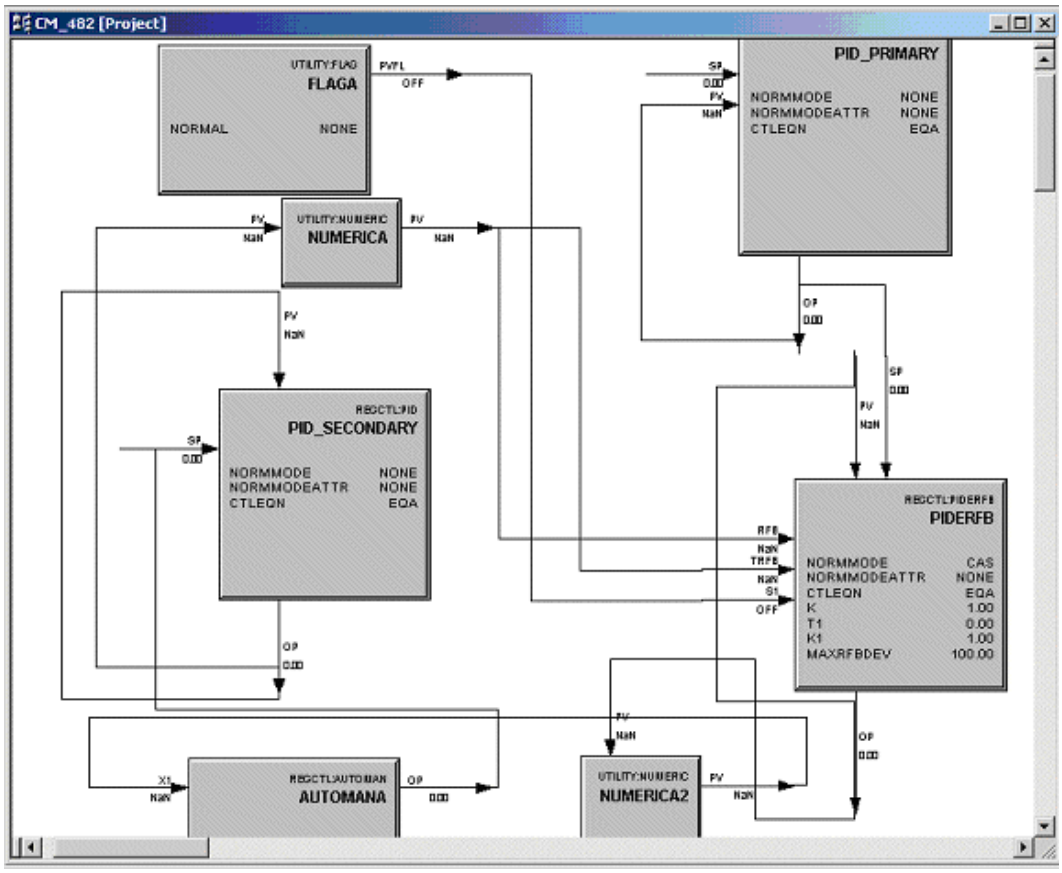


Figure 25 PIDER block used in a cascade control strategy with PID, AUTOMAN, and NUMERIC blocks

Each PIDER block supports the following user configurable attributes. The following table lists the given name of the “Tab” in the parameter configuration form and then briefly describes the attributes associated with that Tab. This data is only provided as a quick document reference, since this same information is included in the on-line context sensitive Help.

Configuration Tab	Description
Main	• Name – Block (Tag) name of up to 16 characters long. Must be unique within the CM block containing it. • Item Name - The name of the Entity that the Control Module containing the block will be associated with in the Enterprise Model Builder hierarchy. • Description (DESC) – Block descriptor of up to 24 characters long. • Engineering Units (EUDESC) – Lets you specify a text string of up to 16 characters to identify the variable values associated with this block. For example, you could specify DEGF for temperature values in degrees Fahrenheit. This name is used on any associated displays and generated reports. • Execution Order in CM (ORDERINCM) – Specifies the execution order of the block in the CM relative to other blocks contained in this CM. Enter as a number between 1 to 32767. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. • PVEU Range High (PVEUHI) – Lets you specify the high input range value in engineering units that represents 100% full scale PV input for the block. The default value is 100. • PVEU Range Low (PVEULO) – Lets you specify the low input range value in engineering units that represents the 0 full scale PV input for the block. The default value is 0 (zero). • Manual PV Option (PVMANOPT) – Lets you specify the mode and output the block is to assume when PVSTS changes to MANual. The selections are: – NOSHED - No changes. – SHEDHOLD - Sets MODE to MANual and MODEATTR to Operator, disables external mode switching (ESWPERM), and holds output at the last good value. – SHEDLOW - Sets MODE to MANual and MODEATTR to Operator, disables external mode

Configuration Tab	Description
	switching (ESWPERM), and sets output to its extended low limit (OPEXLOLM) value. – SHEDHIGH - Sets MODE to MANual and MODEATTR to Operator, disables external mode switching (ESWPERM), and sets output to its extended high limit (OPEXHILM) value. – SHEDSAFE - Sets MODE to MANual and MODEATTR to Operator, disables external mode switching (ESWPERM), and sets output to the configured safe output (SAFEOP) value. • The default selection is SHEDHOLD. • The block requests its primary to initialize after a mode shed or lets the primary know that it is woundup, if it does shed its mode. An operator can change the block's mode after it is shed, but, the operator must first set PVMANOPT to NOSHED, so the mode doesn't shed again. When PVSTS returns to normal, the block clears its primary initialization request but remains in MANual mode after a mode shed. An operator must return the block to its normal mode. If mode was not shed, the block clears its windup condition and does a oneshot initialization. It also requests the primary to do a oneshot initialization. • Normal Mode (NORMMODE) – Lets you specify the MODE the block is to assume when the Control to Normal function is initiated through the Station display. Selections are MANual, AUTOmatic, CAScade, BackupCAScade and NONE. All selections are not valid for a given block. The default selection is NONE. • Normal Mode Attribute (NORMMODEATTR) – Lets you specify the mode attribute (MODEATTR) the block is to assume when the Control to Normal function is initiated through the Station display. Selections are NONE, OPERATOR and PROGRAM. The default selection is NONE. • Mode (MODE) – Lets you set the block's current MODE. The selections are MANual, AUTOmatic, CAScade, BackupCAScade, NONE, and NORMAL. All selections are not valid for a given block. The default

Configuration Tab	Description
	selection is MANual. MODE identifies who may store values to the block's initializable inputs or output. Blocks strictly enforce the MODE assignment. For example, if the MODE is CAScade and the block is getting its SP from another block, an operator is prevented from overwriting the SP value. However, If there is a breakdown in the cascade loop, the MODE can be changed so an operator can write a value to the SP. Some function blocks perform automatic mode switching (or mode shedding), while others require manual intervention. The block's MODE is derived at "runtime" based on current conditions. MODE processing checks for the following conditions, and changes the block's MODE as appropriate. – External request for MODE switching. – Safety interlock request. • Mode Attribute (MODEATTR) – Lets you set the block's mode attribute. The selections are NONE, OPERATOR, PROGRAM, and NORMAL. The default selection is OPERATOR. MODEATTR identifies who may store values to the output (OP), when the block's MODE is MANual. • Permit Operator Mode Changes (MODEPERM) – Lets you specify if operators are permitted to make MODE changes or not. The default is Enabled (checked). A store to MODE does not change the NORMMODE. • Permit External Mode Switching (ESWPERM) – Lets you specify if external MODE switching through user configured interlocks is permitted or not, if you have at least an Engineering access level. The default is Disabled (unchecked). • Enable External Mode Switching (ESWENB) – Lets you specify if external MODE switching through user configured interlocks is enabled or not, if ESWPERM is checked (Permitted). The default is Disabled (unchecked). • Enable Secondary Initialization Option (SECINITOPT) – Lets you specify if the block is to ignore initialization and override requests from the secondary or not. The default selection is Enabled (checked, do not ignore).

Configuration Tab	Description
	• Safety Interlock Option (SIOPT) – Lets you specify MODE and OP block is to assume upon a safety interlock alarm. The selections are NO_SHED, SHEDHOLD, SHEDLOW, SHEDHIGH, and SHEDSAFE. The default selection is SHEDHOLD. • Bad Control Option (BADCTLOPT) – Lets you specify MODE and OP block is to assume if CV goes BAD. The selections are NO_SHED, SHEDHOLD, SHEDLOW, SHEDHIGH, and SHEDSAFE. The default selection is NOSHED.
Algorithm	• Control Equation Type – Lets you select the control equation the block is to use. The selections are EQA, EQB, EQC, EQD, and EQE. See the PID Equations section for this block for details. The default selection is EQA. • Control Action – Lets you specify if the block is to provide direct or reverse control action. The default selection is REVERSE, which means output decreases as input increases. • T1 (minutes) (T1) – Lets you set integral time to be used for the integral term in the control equation. • T1 High Limit (minutes) (T1HILM) – Lets you define the high limit value in minutes for the integral time setting. The default limit is 1440. • T1 Low Limit (minutes) (T1LOLM) – Lets you define the low limit value in minutes for the integral time setting. The default limit is 0. • T2 (minutes) (T2) – Lets you set the derivative time to be used for the derivative term in the control equation. • T2 High Limit (minutes) (T2HILM) – Lets you define the high limit value in minutes for the derivative time setting. The default limit is 1440. • T2 Low Limit (minutes) (T2LOLM) – Lets you define the low limit value in minutes for the derivative time setting. The default limit is 0. • High Gain Limit (GAINHILM) – Lets you set a high limit for the gain (K) value. If this value is exceeded, K

Configuration Tab	Description
	is clamped to this limit. The default value is 240. • Low Gain Limit (GAINLOLM) – Lets you set a low limit for the gain (K) value. If K is less than this value, it is clamped to this limit. The default value is 0. • Gain Options (GAINOPT) – Lets you select the type of gain term to be used in the PID equation. The default value is LIN. The selections are: – LIN - The LINear gain option provides a proportional control action that is equal to a constant (K) time the error (PV -SP). This applies to equations A, B, and C. – GAP - The GAP gain option reduces the sensitivity of the control action when the PV is in a user-specified band (gap) around the set point. This applies to equations A, B, and C. – NONLIN - The NONLIN gain option provides a proportional control action that is equal to the square of the error, rather than the error itself. This applies to equations A, B, and C. – EXT - The EXTernal gain option modifies the gain (K) by an input value from either the process, another function block, or a user program. You can use this option to compensate for nonlinear process gain - lets you tune the PID gain independent of the normal operating point of the process. For example, If you are controlling the level of tank whose cross-section is not constant, you can use the EXT option to modify the gain to compensate for the nonlinear rate of level change, which is caused by the changing shape of the tank. This applies to equations A, B, and C. • Overall Gain (K) – Lets you set the overall gain value used to calculate the proportional term in the PID equation. The default value is 1. • Gap High Limit (GAPHILM) – Lets you define the high limit value in PV engineering units to be used when calculating GAP gain. • Gap Low Limit (GAPLOLM) – Lets you define the low limit value in PV engineering units to be used when

Configuration Tab	Description
	calculating GAP gain. • Gap Gain Factor (KMODIFGAP)– Lets you specify the value to be used for calculating overall gain (K) when the PV input is within the user specified band (GAPLOLM - GAPHILM) around the SP. The value range is 0.0 to 1.0. • Linear Gain Factor (KLIN) – Lets you specify the value to be used for calculating the overall gain (K) in association with GAP, NONLIN, or EXT gain option. • Non-Linearity Form (NLFORM) –Lets you specify the non-linearity form (0 or 1) to be used for calculating gain (K) based on the formula shown for Non Linear Gain in the Gain Options section. The default value is 1. • Non-Linear Gain Factor (NLGAIN) –Lets you specify the non-linear gain value to be used for calculating gain (K) based on the formula shown for Non Linear Gain in the Gain Options section. The default value is 0. • External Gain Factor (KMODIFEXT) – Lets you specify an input value from either the process, another function block, or a user program to be used to modify the gain (K) calculation per this formula: K = KLIN * KMODIFEXT See External Gain in the Gain Options section for more details. The default value is 1. • Legacy Gap: An option which allows the user to revert back to older GAP and Nonlinear gain calculations so that there is no change in behavior after migration.from TPS to Experion The option when enabled will allow the Experion controllers to have the same behavior as the TPS xPM controller. See **Error! Reference source not found.**
SetPoint	• SP (SP) – Lets you specify an initial set point value. The default value is 0. • High Limit (SPHILM) – Lets you specify a high limit value for the SP. If the SP value exceeds this limit, the block clamps the SP to the limit value and sets the SP

Configuration Tab	Description
	high flag (SPHIFL). The default value is 100. • Low Limit SPLOLM) – Lets you specify a low limit value for the SP. If the SP value falls below this limit, the block clamps the SP to the limit value and sets the SP low flag (SPLOFL). The default value is 0. • Mode (TMOUTMODE) – Lets you select the desired MODE the block is to assume, if an initializable input times out, which means the input has not been updated within a designated timeout time. The selections are AUTOmatic, BCAScade, CAScade, MANual, NONE, and NORMAL. The default selection is MANual. • Time (TMOUTTIME) – Lets you specify a time in seconds that must expire before the block assumes that its input update has timed out. The block must be in CAScade mode for it to monitor its primary input for timeout. The default setting is 0, which means the timeout function is disabled. If the input is from a connection in another controller in a peer-to-peer architecture, the actual timeout time equals the configured TMOUTTIME plus the CDA timeout time. The CDA timeout time equals four times the configured CEE subscription rate. For example, if the CEE subscription rate is 100 milliseconds and the TMOUTTIME is 5 seconds, the actual timeout time for the block is 4 times 100ms plus 5s or 5.4 seconds. • Enable Advisory SP Processing (ADVDEVOPT) – Lets you specify whether or not the block is to generate a deviation alarm when the PV deviates from a user specified “advisory” SP value. The default selection is unchecked (Disabled). • Advisory SP Value (ADVSP) – Lets you set an advisory SP value in PV engineering units, when Advisory SP Processing is enabled. When PV exceeds or deviates from this value, the block generates an advisory deviation alarm. • Enable PV Tracking (PVTRAKOPT) – Lets you specify if PV tracking is to be applied to this block or not. When PV tracking is enabled, this option sets the SP equal to PV when the operation of a cascade loop is interrupted by either initialization, operator or

Configuration Tab	Description
	program operation (such as, setting the MODE to MANual). This option is normally enabled for PIDs in a cascade loop. The default selection is unchecked (disabled). See the PV tracking section for this block for more details. • Enable PV Tracking in Auto/Init (PVTRAKOPTAI) – Lets you specify if PV Tracking option for the specific case when the block is in Auto mode and it is undergoing initialization (INITMAN is ON) is be enabled or not. The default selection is unchecked (disabled). • Enable SP Ramping (SPTVOPT) – Lets you specify if an operator can initiate a set point ramp action or not. It provides a smooth transition from the current set point value to a new one. The default selection is box unchecked (disabled). See the Set point ramping section for this block for more details. • Normal Ramp Rate (SPTVNORMRATE) – Lets you specify a ramp rate in engineering units per minute for the SP ramping function, when it is enabled. This lets an operator start the SP ramping function without specifying a ramp time. The default selection is Not a Number (NaN). See the Set point ramping section for this block for more details. • Max. Ramp Deviation (SPTVDEVMAX) – Lets you specify a maximum ramp deviation value in engineering units per minute for the SP ramping function, when it is enabled. Keeps PV within the specified deviation range for a ramping SP by stopping the SP ramp until the PV input catches up with the SP value. The default value is NaN, which means no ramp deviation check is made. See the Set point ramping section for this block for more details. • Enable SP Push: (PUSHSP) – Lets you specify that the PIDER SP will be pushed from an Inter Cluster Gateway when the PIDER is the secondary of a cascade that extends over two Experion clusters.
Output	• High Limit (%) (OPHILM) – Lets you specify the output high limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If

Configuration Tab	Description
	the CV range is 50 to 500 and you enter a High Limit of 90%, the high limit in engineering units is 90% times 450 or 405 + 50 (CVEULO) equals 455. This check is not applied for a function block that is in the MANual mode. The default value is 105%. • Low Limit (%) (OPLOLM) – Lets you specify the output low limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you enter a Low Limit of 10%, the low limit in engineering units is 10% times 450 or 45 + 50 (CVEULO) equals 95. This check is not applied for a function block that is in the MANual mode. The default value is -5%. • Extended High Limit (%) (OPEXHILM) – Lets you specify the output extended high limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you use the default value of 106.9%, the extended high limit in engineering units is 106.9% times 450 or 481.05 + 50 (CVEULO) equals 531.05. This check is not applied for a function block that is in the MANual mode. The default value is 106.9%. • Extended Low Limit (%) (OPEXLOLM) – Lets you specify the output extended low limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you use the default value of -6.9%, the extended low limit in engineering units is -6.9% times 450 or -31.05 + 50 (CVEULO) equals 18.95. This check is not applied for a function block that is in the MANual mode. The default value is -6.9%. • Rate of Change Limit (%) (OPROCLM) – Lets you specify a maximum output rate of change limit for both the positive and negative directions of the output in percent per minute. This lets you prevent an excessive rate of change in the output so you can match the slew rate of the control element to the control dynamics. We recommend that you configure this value before you tune the loop, so tuning can accommodate any slow down in response time caused by this rate limiting. This check is not applied for a function block that is in the MANual mode. The default value is Not a Number

Configuration Tab	Description
	(NaN), which means no rate limiting is applied. • Minimum Change (%) (OPMINCHG) – Lets you specify an output minimum change limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). This lets you define how much the OP must change before the function block outputs a new value. It filters out changes that are too small for the final control element to respond to. This check is not applied for a function block that is in the MANual mode. The default value is 0, which means no change limiting is applied. • Safe OP (%) (SAFEOP) – Lets you specify the safe output value as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 0 to 500 and you enter a Safe OP of 50%, the safe output value in engineering units is 50% times 500 or 250. The default value is Not a Number (NaN), which means the OP is held at its last good value. ' • CVEU Range High (CVEUHI) –Lets you specify the high output range value in engineering units that represents 100% full scale CV output for the block. The default value is 100. • CVEU Range Low (CVEULO) – Lets you specify the low output range value in engineering units that represents the 0 full scale CV output for the block. The default value is 0 (zero). • Output Bias (OPBIAS.FIX) – Lets you specify a fixed bias value in engineering units that is added to the Calculated Variable (CV) output value. See the Output Bias section for this function block for details. The default value is 0, which means no value is added. • Output Bias Rate (OPBIAS.RATE) – Lets you specify an output floating bias ramp rate in engineering units per minute. This bias rate is only applied when the floating bias in non-zero. See the Output Bias section for this function block for details. The default value is Not a Number (NaN), which means no floating bias is calculated. As a result, if the primary does not accept the block's initialization value, a bump in OP occurs.
Alarms	• Type – Identifies the types of alarms this block supports. Of course, these alarms also interact with

Configuration Tab	Description
	other block configuration options such as the Safety Interlock Option (SIOPT) and Bad Control Option (BADCTLOPT). The types are: – OP High (OPHIALM.FL) – OP Low (OPLOALM.FL) – Deviation High (DEVHIALM.FL) – Deviation Low (DEVLOALM.FL) – Advisory Deviation (ADVDEVALM.FL) – Safety Interlock (SIALM.FL) – Bad Control (BADCTLALM.FL) • Enable Alarm (ADVDEVOPT and SIALM.OPT) – Lets you enable or disable Advisory Deviation and/or Safety Interlock alarm types. A check in the box means the alarm is enabled. The default selections are unchecked or Disabled for Advisory Deviation and checked or Yes (enabled) for Safety Interlock. You can also configure the ADVDEVOPT and SIALM.OPT parameters as a block pins, configuration and/or monitoring parameters so they appear on the block in the Project and Monitoring tree views, respectively. • Trip Point – Lets you specify the following trip points for the given alarm. The default value is NaN, which disables the trip point. • OPHIALM.TP (Output High Alarm Trip Point) • OPLOALM.TP (Output Low Alarm Trip Point • DEVHIALM.TP (Deviation High Alarm Trip Point) • DEVLOALM.TP (Deviation Low Alarm Trip Point) • ADVDEVALM.TP (Advisory Deviation Alarm Trip Point) • Priority – Lets you set the desired priority level individually for each alarm type (OPHIALM.PR, OPLOALM.PR, DEVHIALM.PR, DEVLOALM.PR, ADVDEVALM.PR, SIALM.PR, BADCTLALM.PR,). The

Configuration Tab	Description
	default value is LOW. The levels are: – NONE - Alarm is neither reported nor annunciated. – JOURNAL - Alarm is logged but it does not appear on the Alarm Summary display. – LOW, HIGH, URGENT - Alarm is annunciated and appears on the Alarm Summary display. • Severity – Lets you assign a relative severity individually for each alarm type (OPHIALM.SV, OPLOALM.SV, DEVHIALM.SV, DEVLOALM.SV, ADVDEVALM.SV, SIALM.SV, BADCTLALM.SV,) as a number between 0 to 15, with 15 being the most severe. This determines the alarm processing order relative to other alarms. The default value is 0. • Deadband Value (ALMDB) – Lets you specify a deadband value that applies to all analog alarms to prevent nuisance alarms due to noise at values near the trip point. The default value is 1. Note that this value is loaded to the individual alarm parameters (for example, OPHIALM.DB and OPLOALM.DB) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB. • Deadband Time (ALMTM) – Lets you specify a time in seconds to define how long an analog alarm must exist before it is set true. The default value is 0, which means the alarm is set true as soon as the value exceeds the deadband value. Note that this value is loaded to the individual alarm parameters (for example, OPHIALM.TM and OPLOALM.TM) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB. • Deadband Units (ALMDBU) – Lets you specify if the deadband value represents percent or engineering units. The default value is percent.

Configuration Tab	Description
	Note that this value is loaded to the individual alarm parameters (for example, OPHIALM.DBU and OPLOALM.DBU) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB.
SCM	• SCM Mode Tracking Option (MODETRACK) – Lets you select the desired Mode Tracking function for the SCM associated with this block's Control Module. It defines how the FB will set the state of the MODEATTR based upon the MODE of the SCM. See the *Sequential Control Module User's Guide* for more information on this function. The default selection is ONESHOT. The selections are: – None – ONESHOT – SEMICONT – CONTRTN – CONT • Option Type – Lets you specify the action the function block is to take when the SCM goes into an abnormal state. The Starting State Option (STARTOPT) applies when the SCM state is Checking, Idle, or Complete. The Stop/Abort State Option (STOPOPT) applies when the SCM state is Stopping or Stopped, Aborting or Aborted. The Hold State Option (HOLDOPT) applies when the SCM state is Holding or Hold. The Restart State Option (RESTARTOPT) applies when the SCM state is Resume or Run. The NONE and LASTREQ are the only selections for the Restart State Option. You can select from one of these types for the other options as applicable for the given regulatory control function block: – NONE - No changes. – MAN - Set MODEREQ = MANUAL. – AUTO - Set MODEREQ = AUTOMATIC.

Configuration Tab	Description
	– CAS - Set MODEREQ = CASCADE. – FIXEDOP - Set OPREQ = Configured Value. – HOLDPV - Set SPREQ = PV. – FIXED SP - Set SPREQ = Configured Value and SPRATEREQ = NaN. – RAMPEDSP - Set SPTVREQ = Configured Value and SPRATEREQ = Configured Rate. • Value (STARTVAL, STOPVAL, HOLDVAL) – Depending upon Option Type selection, lets you specify an output or set point value within the respective range. For output, within OPEXLOLM to OPEXHILM and within SPLOLM to SPHILM, for set point. The default value is NaN (Not a Number). • Rate (STARTRATE, STOPRATE, HOLDRATE) – When the RAMPEDSP option is selected, lets you specify a rate value (STARTRATE, STOPRATE, HOLDRATE) for setting the SPRATEREQ for an SP ramping function.
Block Pins	Lets you select the available parameters that you want to expose as input/output pins on the function block graphic in Control Builder.
Configuration Parameters	Lets you select the available parameters that you want to appear on the face of the function block in the Project tab in Control Builder.
Monitoring Parameters	Lets you select the available parameters that you want to appear on the face of the function block in the Monitoring tab in Control Builder.
Block Preferences	Lets you change several block-viewing preferences including the color of the block's faceplate.
Insertion Type	Lets you include an insertion type from a CAB instances in the block. See *CAB insertion configuration considerations for regulatory control blocks* for more information

Function

The PIDER block requires five inputs -- PV, SP, RFB, TRFB and S1. RFB is the reset feedback value, TRFB is the tracking value, and S1 is a tracking control switch (a Boolean input). S1 indicates whether the PID output should be combined with the RFB or replaced by TRFB.

- PV is pulled from another function block.
  PV is typically pulled from a Data Acquisition (DATAACQ) function block which performs PV limit checking and alarming.
- SP is pulled from another function block, or stored by the operator or a user program.
  If SP is pulled from a primary, the PID's Mode must be Cascade; and if it is stored by the operator or a user program, Mode must be Manual or Automatic. If Mode is Cascade, the PID must perform timeout checking on SP (to make sure the primary is periodically updating it).
- RFB signal comes from the remote (foreign) controller's PV, and the tracking value comes from its PV or SP. If the PIDER block is used for external tracking features only, this input is not required. You cannot store a value to this parameter.
- TRFB is pulled from another function block. You cannot store a value to this parameter.
- S1 is triggered by another function block or set by a user-written program. When S1 is Off, the PID control value output ($CV_{PID)}$ is combined with the reset feedback (RFB) value to obtain the control value (CV); and when it is On, CV is set equal to the tracking value (TRFB).

A PIDER also has the following optional inputs. Typically, these are flags which may be stored by the operator or user program to change the normal operation of the PIDER.

- ESWAUTO, ESWCAS, ESWMAN and SI – Indicates if an external source, such as a user program, wants to change the PIDER's Mode:
  - If ESWAUTO = On, the external source wants to change the Mode to Auto.
  - If ESWCAS = On, the external source wants to change the Mode to Cascade.
  - If ESWMAN = On, the external source wants to change the Mode to Manual.
  - If SI = On, the external source wants to invoke the PIDER's safety interlock logic.

If a BACKCALC connection is made to the secondary, the PIDER reads BACKCALCIN from the secondary before calculating its OP:

- BACKCALCIN is a "data container", which means it contains many pieces of information but is accessed by a single read. Among other things, the information in BACKCALCIN indicates if the secondary is wound-up or if it wants the PIDER to initialize.
- The individual BACKCALCIN/BACKCALCOUT connections for each output used are automatically built by Control Builder as implicit/hidden connections. This means you do not have to manually wire BACKCALC connections in Control Builder.
- The secondary builds BACKCALCIN when it receives a read request from the primary. This way, BACKCALCIN is guaranteed to contain the most current status.

Operating modes and mode handling

The PIDER block operates in the following modes:

- MAN (MANual)
  - If mode is MANual, OP may be stored by the operator or a user program; PV and SP are ignored – if a primary exists, it goes to the initialized state.
- AUTO (AUTOmatic)
  - If mode is AUTOmatic, SP (or SPP) may be stored by the operator or a user program; if a primary exists, it goes to the initialized state. SP contains set point value in engineering units and SPP contains the value in percent.
- CAS (CAScade)
  - If mode is CAScade, SP is pulled from a primary; if the primary is off-control (that is, inactive or initializing) or the connection is bad, the PIDER block invokes timeout processing.

Required inputs

The required number of inputs is determined by the mode of the PIDER block.

- If Mode is CAScade, five inputs are required – PV, SP, RFB, TRFB and S1.
- If Mode is AUTOmatic or MANual, PV, RFB, TRFB and S1 are required.
  - SP is the only initializable input; other inputs are non-initializable.
  - PV must be pulled from another block; you cannot store to it – typically it is connected to the output of an auxiliary or data acquisition (DATAACQ) block.
  - If Mode is CAScade, SP is pulled from another block; if Mode is AUTOmatic, it may be stored by the operator or a user program.
  - The PIDER block may have one primary or none, depending on whether SP is configured or not; there is one primary per initializable input.
  - RFB and TRFB must be pulled from another block, you cannot store to them. The RFB input is optional. If the PIDER block is used for external tracking features only, the RFB input is not required.
  - S1 can be triggered by another function block or set by a user-written program.

Input ranges and limits

- You must specify a PV engineering unit range, PVEUHI and PVEULO.
  - PVEUHI and PVEULO define the full range of PV in engineering units. PVEUHI represents the 100% of full scale value. PVEULO represents the 0% of full scale value.
  - PVEUHI and PVEULO also define the engineering unit range of SP – PV and SP are assumed to have the same range.
- The PIDER block assumes PV is within PVEUHI and PVEULO – it applies no range check – however, PV typically comes from a data acquisition (DATAACQ) block which applies its own limit and range check.
- SPHILM and SPLOLM define set point operating limits in engineering units.
  - The operator is prevented from storing a set point value that is outside these limits; if the primary or a user program attempts to store a value outside of the limits, the PIDER block clamps it to the appropriate limit and sets the primary's windup status.
  - The PIDER block provides the SP high/low limits (SPHILM/SPLOLM) to the primary through BACKCALC. The primary uses this for its output range (CVEUHI/CVEULO).
- SP contains set point value in engineering units and SPP contains the value in percent.
  - If Mode is AUTOmatic, the operator or a user program may store to either SP or SPP.
  - The PIDER block monitors SP for time-out.
- The RFB and TRFB values typically come from a remote controller. The PIDER block applies no range check for these parameters.
- The S1 input is a Boolean flag and the values are only On and Off.

Initializable outputs

"Initializable output" and "initializable input" are variable attributes, similar to data type or access level. A variable with the "initializable" attribute has an associated BACKCALC variable, and when a connection is created between an initializable input and initializable output, you can also create a BACKCALC connection. Control Builder automatically builds the required BACKCALC connections, so you don't have to create them manually. These "implicit" build connections are "hidden" from view and the related parameter pins are not exposed on the control chart.

ATTENTION

The PIDER block does not support output initialization, and therefore cannot have a secondary. Initialization only occurs when the tracking control switch (S1) is On.

For example, if you connect OP from a PIDER block to a PID block or an AOCHANNEL block, Control Builder automatically creates the BACKCALCOUT to BACKCALCIN connection.

- OP = Calculated output, in percent.
- OPEU = Calculated output, in engineering units.

You may create a connection to OP or OPEU but not both. Therefore, this block may have only one secondary. If you do not create a connection to OP or OPEU, then the block does not have a secondary. Alternately, if you connect OP or OPEU to a non-initializable input, then this block does not have a secondary. (Note that the default OP connection pin is exposed on the blocks and the implicit/hidden connection function automatically makes the appropriate value/status parameter (OPX/OPEUX) connection when required. For example, if you connect the output from a primary PID block (PIDA.OP) to the set point of a secondary PID block (PIDB.SP), the implicit/hidden connection is made to PIDA.OPX to provide value/status data.)

ATTENTION

Be sure you use a FANOUT block to make multiple output connections. We recommend that you do not make multiple connections from a single PID output.

Control initialization

The PIDER block does not perform normal initialization and windup processing associated with secondaries. Initialization occurs when the tracking control switch (S1) is On.

This block provides the same initialization data to its primary as the normal PID block

- Note that SECINITOPT may be used to ignore initialization requests from the secondary.
- If the secondary is requesting initialization, the PID block:
  - initializes its output
    CV = initialization value from the secondary
  - sets initialization request parameters for its primary
    INITREQ = On
    INITVAL = SP

Output bias

If the PIDER block algorithm is configured as Equation A, B, C, or D, no output bias (OPBIAS) is applied.

The OPBIAS is the sum of the user-specified fixed bias (OPBIAS.FIX) and a calculated floating bias (OPBIAS.FLOAT). The purpose of the floating bias is to provide a bumpless transfer when the function block initializes or changes mode as long as the PIDER block is the first initializable block.

- OPBIAS is recomputed under the following conditions to avoid a bump in the output. (Note that the PIDER block only applies OPBIAS.FLOAT to the output for the latter two conditions, when it is the first initializable block.)
  - When the function block starts up (that is, goes Active).
  - When the function block initializes (for example, the secondary requests initialization).
  - When the mode changes to Auto or Cascade.

ATTENTION

When the mode is Manual, OPBIAS is not used, since the output is not calculated. This means that the OPBIAS is not recomputed, when the mode changes to Manual.

- You may set the OPBIAS value only if the function block is Inactive or Mode equals Manual. This is done to prevent a bump in the output when the bias is changed. The following occurs when you set the OPBIAS value.
  - The total bias (OPBIAS) and fixed bias (OPBIAS.FIX) are both set to the entered value.
  - The floating bias (OPBIAS.FLOAT) is set to zero.

ATTENTION

When the function block goes Active or the Mode changes to Auto or Cascade, OPBIAS and OPBIAS.FLOAT are recomputed.

- There are no limit checks applied when you set an OPBIAS value. However, after the bias is added to CV, the result is compared against the output limits and clamped, if necessary.
- You configure the value for the fixed bias (OPBIAS.FIX) and it is never overwritten by the floating bias (OPBIAS.FLOAT). This means the total bias will eventually equal the OPBIAS.FIX , if you configure OPBIAS.RATE to ramp down OPBIAS.FLOAT.
- The OPBIAS.FLOAT is calculated as follows.

$$OPBIAS.FLOAT = CV_{INIT} - (CV_{UNBIASED} + OPBIAS.FIX)$$

Where:

CV_{INIT} = initialization value received from the secondary

$CV_{UNBIASED}$ = unbiased calculated value (based on input from the primary)

OPBIAS.FIX = fixed bias (user-specified)

- If the primary accepts this block's initialization request, then CV + OPBIAS.FIX should be the same as CVINIT and OPBIAS.FLOAT will be zero. In most cases, OPBIAS.FLOAT will be zero. However, if the primary does not accept this block's initialization request because the primary is a FANOUT block or it was configured to ignore initialization, then OPBIAS.FLOAT value will not be zero.

  If OPBIAS.FLOAT is not zero, you can configure it to ramp down to zero through the OPBIAS.RATE parameter.

- You configure the OPBIAS.RATE to apply a ramprate to the OPBIAS.FLOAT. It is only used when the OPBIAS.FLOAT is not zero. The OPBIAS.RATE is expressed in Engineering Units per minute and may have the following values.

  – Zero:
    If OPBIAS.RATE is zero, a OPBIAS.FLOAT is calculated and bumpless transfer is guaranteed. However, if OPBIAS.FLOAT is not zero, it will never ramp down.

  – Non-zero:
    If OPBIAS.RATE is not zero, an OPBIAS.FLOAT is calculated and bumpless transfer is guaranteed. If the OPBIAS.FLOAT is not zero, it is ramped to zero at the rate you configured for the OPBIAS.RATE parameter.

    The function block ramps the OPBIAS.FLOAT to zero by applying the following calculation each time it executes.

    OPBIAS.FLOAT = OPBIAS.FLOAT – (OPBIAS.RATE / cycles_per_Min)

  Where:

    cycles_per_min = number of times the function block executes per minute (calculated)

  – NaN:
    When the OPBIAS.RATE is Not a Number (NaN), no OPBIAS.FLOAT is calculated. This means a bump in the output will occur, if the primary does not accept this block's initialization value.

Output ranges and limits

- CVEUHI and CVEULO define the full range of CV in engineering units.
  - CVEUHI and CVEULO must be specified by the user. The specified values must match the engineering units (EU) range of the RFB and TRFB signals, which are the range of the remote (foreign) controller or secondary.
- OPHILM and OPLOLM define the normal high and low limits for OP as a percent of the CV range – these are user-specified values.
  - OP is clamped to these limits if the algorithm's calculated result (CV) exceeds them, or another block or user program attempts to store an OP value that exceeds them.
  - The operator may store an OP value that is outside of these limits.
- OPEXHILM and OPEXLOLM define the extended high and low limits for OP as a percent of the CV range – these are user-specified values.
  - The operator is prevented from storing an OP value that exceeds these limits.
- MAXRFBDEV is the maximum deviation allowed between CV and RFB, in percent. It is used to provide windup protection for OP.
  - If the scaled, integrated deviation of CV from RFB exceeds MAXRFBDEV in the positive direction, the PIDER block sets the output windup status (ARWOP) to High, which will prevent CV from going higher. If the deviation exceeds MAXRFBDEV in the negative direction, it sets ARWOP to Low, which will prevent CV from going lower. This occurs only if the tracking control switch (S1) is Off.

Direct or reverse control

A PIDER block may be configured for direct-control action or reverse-control action.

Changing the control action effectively changes the sign of the gain.

- With direct-control action, an increase in the error (PV - SP) increases the PIDER output (CV).
- With reverse-control action, an increase in the error (PV - SP) decreases the PIDER output (CV).

For example, if SPP = 50% and PVP = 51%, then the error is 1%.

- With direct-control action, if PVP changes to 52%, the error increases causing CV to increase.
- With reverse-control action, if PVP changes to 52%, the error increases causing CV to decrease. .

Set Point Ramping

The Set Point Ramping option lets you ramp from the current set point value to a target set point value. You enable this option by selecting the Enable SP Ramping check box on the block's parameter configuration form. This is equivalent to setting the SPTVOPT parameter to Enable. You can also configure the following related parameters through the configuration form or the equivalent parameters.

Parameter	Description
Normal Ramp Rate (SPTVNORMRATE)	Normal ramp rate value in engineering units that you enter. The value can be Not a Number (NaN) or greater than zero. If value is NaN, it means a "step change" in the SP, which is the same as a ramp time of zero. This parameter lets you start SP ramping without specifying a ramp time. This function block calculates a ramp time (SPTVTIME) and ramp rate (SPTVRATE) as follows, when SP ramping is enabled: • If SPTVNORMRATE is a value other than zero or NaN: SPTVRATE = SPTVNORMRATE SPTVTIME = |(SPTV – SP)| / SPTVRATE • Otherwise,: SPTVRATE = NaN SPTVTIME = 0 (That is, do a step change.)

Parameter	Description
Max. Ramp Deviation (SPTVDEVMAX)	Lets you specify a maximum deviation in engineering units per minute allowed between PV and SP during ramping. The value can be NaN or greater than zero. If value is NaN, it means no ramp deviation checking is done. If the maximum ramp deviation value is other than NaN, SP ramping stops when the absolute value of the deviation (|PV – SP|) exceeds the maximum deviation. The deviation flag (SPTVDEVFL) is set, and SP ramping state (SPTVSTATE) remains in Run. Ramping resumes as soon as the absolute value of the deviation returns within the maximum deviation limit. This also resets the deviation flag (SPTVDEVFL). If you have entered a ramp time (SPTVTIME) and ramping is interrupted by maximum ramp deviation, the actual ramp time (SPTVTIME) will be greater than the time you specified.

You can configure these other SP ramping related parameters to appear as block pins or monitoring parameters that can be viewed on the block during Control Builder monitoring, as shown in the following figure. You can access these parameters to invoke and monitor SP ramping while monitoring the control strategy through Control Builder or the PID Loop Point Detail display in Station.

Parameter	Description
SPTV	SP target value that you enter. You can only set SPTV when the SPTVOPT is Enabled, the SPTVSTATE is Off or Preset, and the block's mode is Auto or Manual. When you set SPTV with the block's Control Module active, this occurs: • The block calculates a ramp time (SPTVTIME). • The SPTVSTATE goes to Preset. • A "P" modifier appears next to the SP value on the PIDER detail display in Station.
SPTVDEVFL	SP target value deviation flag indicates when deviation exceeds the maximum ramp deviation limit.

Parameter	Description
SPTVRATE	SP target value ramp rate. This rate is calculated as shown above for the SPTVNORMRATE and as follows: • If you specify a ramp time (SPTVTIME) value other than zero: SPTVRATE = |(SPTV – SP)| / SPTVTIME Otherwise: SPTVRATE = NaN • If you change the SPTVNORMRATE, this block recalculates the ramp time (SPTVTIME) and ramp rate (SPTVRATE) as follows: If ramp time (SPTVTIME) is a value other zero: SPTVRATE = SPTVNORMRATE SPTVTIME = |(SPTV – SP)| / SPTVRATE Otherwise,: SPTVRATE = NaN SPTVTIME = 0 (That is, do a step change.)
SPTVTIME	SP target value time. This time is calculated in conjunction with SPTVRATE as described above or is entered by you. You can only set SPTV when the SPTVOPT is Enabled, the SPTVSTATE is Off or Preset, and the block's mode is Auto or Manual.
SPTVSTATE	SP target value state. The possible states are: • Off, • Preset, or • Run You can only set the SPTVSTATE when the Control Module containing this block is active and the block's mode is Auto. When you set SPTVSTATE to Run from Preset, this occurs: • An "R" modifier appears next to the SP value on the PIDER detail display in Station. • SP begins to ramp toward SPTV and SPTVTIME decreases. When SPTVTIME reaches zero, SP equals SPTV and the SPTVSTATE goes to Off.

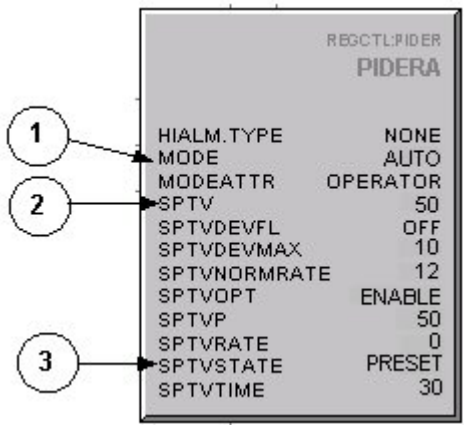


Figure 26 PIDER block with SP ramping parameters configured for monitoring.

The following table includes descriptions of the callouts in the figure above.

Callout	Description
1	Block's mode must be Auto and SPTVSTATE must be Preset, before you can start SP ramping by setting SPTVSTATE to Run with SPTV set to desired value.
2	The SPTVSTATE automatically goes to Preset, when: • You set a value for SPTV or SPTVTIME. • Mode changes to Manual while SPTVSTATE is Run. • Block is initialized (INITMAN = ON) while SPTVSTATE is Run. However, a oneshot initialization does not cause a change in SPTVSTATE. The SPTVSTATE automatically goes to Off, when: • SP is set by you, a program or another function block. • Mode changes to Cascade or Backup Cascade. • Control Module goes Inactive.

Callout	Description
3	You can only set a value for SPTV and SPTVTIME, when: • SPTVSTATE is Off or Preset, and • Mode is Auto or Manual.

ATTENTION

- When SP ramping is Enabled, the SPTVSTATE must be Off before you can make changes to the SP limits (SPHILM and SPLOLM).
- If the anti-reset windup status (ARWNET) indicates that SP is woundup (Hi, Lo or HiLo), SP ramping stops. When ARWNET indicates that SP has returned to normal, SP ramping continues from where it stopped.

PV tracking

The PV Tracking option sets SP equal to PV when a cascade is broken due either to function block initialization or operator or program action (such as, setting the mode to Manual).

You select the Enable PV Tracking selection on the block configuration form to enable the function (PVTRAKOPT = Track).

Typically, PV tracking is configured for PID blocks in a cascade configuration strategy. This allows the PIDs to resume control with no error after initialization or when they are taken out of Manual mode.

If PV tracking is configured, the block sets SP equal to PV (subject to SP limits) when either of the following conditions exist:

- block is in Manual mode
- block is initializing and not in Auto mode.

ATTENTION

- PV tracking does not occur on recovery from a bad PV.
- PV tracking does not occur if block is in Auto mode.
  - If block is in Auto mode, it means SP is normally stored by the user.
  - If PV tracking is initiated, this value is lost.
- The PVTRAKOPTAI parameter provides a PV Tracking option for the specific case when the block is in Auto mode and it is undergoing initialization (INITMAN is On). The PV Tracking option parameter (PVTRAKOPT) does not include the Auto/Init case, but does cover several other situations as noted above.

PID equations

The PIDER block provides four different equations for calculating the PID – the CTLEQN parameter is used to specify the desired equation.

- Equation A – all three terms (Proportional, Integral, Derivative) act on the error (PV - SP) as follows:

$$^{-1}\left[1 + \frac{1}{T1_S} + \frac{T2_S}{1 + a * T2_S} * PVP_S - SPP_S\right]$$

- Equation B – the proportional and integral terms act on the error (PV - SP) and the derivative term acts on changes in PV as follows:

$$CV = K * L^{-1}\left[\left(1 + \frac{1}{T1_S} + \frac{T2_S}{1 + a * T2_S}\right) * PVP_S - \left(1 + \frac{1}{T1_S}\right) * SPP_S\right]$$

- This equation is used to eliminate derivative spikes in the control action as a result of quick changes in SP.
- Equation C – the integral term acts on the error (PV - SP) and the proportional and derivative terms act on changes in PV as follows:

$$CV = K * L^{-1}\left[\left(1 + \frac{1}{T1_S} + \frac{T2_S}{1 + a * T2_S}\right) * PVP_S - \left(\frac{1}{T1_S}\right) * SPP_S\right]$$

- This equation provides the smoothest and slowest response to SP changes.
- Equation D – integral control only as follows:

$$CV = L^{-1}\left[\frac{1}{T1_S} * \left(PVP_S - SPP_S\right)\right]$$

ATTENTION

To prevent a bump in the output, you must configure the OPBIAS.RATE parameter for a value (in Engineering Units per minute) other than 0.0 (zero) or NaN (Not a Number) to enable the ramping function for the floating bias.

- Reverse-control action causes the sign of the unbiased CV to be reversed.
- If both options are selected, the unbiased CV is reversed first, and then the fixed and floating bias are added – neither the bias nor the final CV are reversed

Where:

CV	=	output of PID (Equations A, B, C, D) in percent
K	=	gain (proportional term)
L^{-1}	=	inverse of the LaPlace transform
PV	=	process input value in engineering units
PVP	=	PV in percent
a	=	1/16 fixed rate amplitude
s	=	La Place operator
SP	=	set point value in engineering units
SPP	=	SP in percent
T1	=	integral time constant in minutes
T2	=	derivative time constant in minutes

PIDER equation variations

In addtion to the PID equations above, the PIDER block supports the following variations depending on the status of the tracking switch (S1).

If the tracking switch S1 is **Off**, this block combines RFB with the PID output as follows:

$$CV = CV_{PID} + CV_{RFB}$$

If PID equation A, B, or C is configured, the following variation applies:

$$CV_{RFB} = K * L^{-1}\left[\frac{K1}{T1 * S}(rfb_s - CV_s)\right]$$

If PID equation D is configured, the following variation applies:

$$CV_{RFB} = L^{-1}\left[\frac{K1}{T1 * S}(rfb_s - CV_s)\right]$$

If the tracking switch (S1) is **On**, the PIDER block forces CV to track the tracking value (TRFB) input as follows:

$$CV = \frac{TRFB - CVEULO}{CVEUHI - CVEULO} * 100$$

Where:

CV = full value output in percent , PID combined with $CV_{RFB.}$

CV_{PID} = full value output of the PID block. This is an internal calculated parameter and is not available to user-written programs.

K = gain (proportional term)

K1 External reset feedback gain (scaling factor that is applied to RFB).

- K1 is clamped between the limits 0.0 and 1.0.

L^{-1} = inverse of the LaPlace transform

RFB = the external reset feedback value in engineering units (input).

rfb = $\frac{RFB - CVEULO}{CVEUHI - CVEULO} * 100$

s = La Place operator

S1 = the tracking control switch (input)

TRFB = the tracking value (input)

T1 = PID integral time constant in minutes (configured)

Gain options

If equation A, B, or C is selected, any of the following gain equations may be chosen:

- Linear Gain – provides a proportional control action that is equal to a constant (K) times the error.
  - This is the most commonly-used gain option – K is a user-specified constant and has a default value of 1.0.
- Gap Gain – used to reduce the sensitivity of the control action when PV is in a user-specified band (gap) around the set point.
  - Gap size and control action are specified at configuration time through the following parameters:

KLIN	Linear (normal) gain – to be used when PV is outside the gap.
KMODIFGAP	Gain-modification factor – to be used when PV is inside the gap. Range of KMODIFGAP = 0.0 to 1.0.
GAPLOLM	Lower limit of gap – in same engineering units as PV.

GAPHILM Upper limit of gap – in same engineering units as PV.

- Gain (K) is derived as follows:

When PV is outside the gap:

K = KLIN

When PV is inside the gap (SP - GAPLOLM <= PV <= SP + GAPHILM):

K = KLIN * KMODIFGAP

- Nonlinear Gain – provides control action that is proportional to the square of the error, rather than the error itself.
  - Gain (K) is derived as follows:

$$K = KLIN * \left[NLFORM + NLGAIN * \frac{|PV - SP|}{(PVEUHI - PVEULO)} \right]$$

Where:

KLIN = linear (normal) gain (user-configured)

NLFORM = nonlinear gain form (user-configured; may be 0 or 1)

NLGAIN = nonlinear gain (user-configured)

- External Gain – where, when gain (K) is selected, it is modified by an input value that can come from either the process, another function block, or a user program.
  - The main use of this option is to compensate for nonlinear process gain – you can tune the PID gain independently of the normal operating point of the process.
  - For example, in controlling the level of a tank whose cross-section is not constant, the gain could be modified to compensate for the nonlinear rate of level change that is caused by the changing shape of the tank.
  - Gain (K) is derived as follows:

$$K = KLIN * KMODIFEXT$$

Where:

KLIN = linear (normal) gain (user-configured)

KMODIFEXT = external gain modifier (such as from a user program)

Tuning Constant Change Considerations

WARNING

You cannot always reverse output (OP) resulting from changes you make to a tuning constant gain (K), integral time (T1) or derivative time (T2) in an online control loop.

You cannot undo a change in a tuning constant in an online control loop by simply changing the constant back to its original value. The output (OP) does not jump back to its original prior value just because you return the constant to its prior value. In this case, you must put the loop in MANUAL mode and set the output (OP) to the desired value before returning the loop to AUTO mode.

Timeout monitoring

If mode is CAScade, the block performs timeout monitoring on SP – if a good SP value is not received within a predefined time, the block invokes timeout processing.

The maximum time between updates is specified by TMOUTTIME (in seconds)

- Enable timeout monitoring by setting TMOUTTIME to a non-zero value.
- Disable timeout monitoring by setting TMOUTTIME to zero.

Timeout processing

If mode is CAScade and SP times out, the block does the following:

- Sets the input timeout flag (TMOUTFL)
- Holds SP at its last good value.
- Sheds to the user-specified **timeout mode** (MODE = TMOUTMODE).
- Requests the primary to initialize.

The block sets its cascade request flag (CASREQFL), if SP times out and sheds to AUTOmatic mode. This indicates that the block is waiting to return to the CAScade mode, and it will as soon as it brings a good SP value. When it receives a good SP value, the block does the following:

- Changes the mode back to CAScade.
- Updates the SP.

You **cannot** set the CASREQFL. However, it can be cleared by setting the block's MODE to MANUAL. This allows you to disable the automatic return to Cascade mode.

- The block only sets CASREQFL if the original mode was Cascade, the SP input times-out, and TMOUTMODE = AUTO.

ATTENTION

If the input is from a connection in another controller in a peer-to-peer architecture, the actual timeout time equals the configured TMOUTTIME plus the CDA timeout time. The CDA timeout time equals four times the configured CEE subscription rate. For example, if the CEE subscription rate is 100 milliseconds and the TMOUTTIME is 5 seconds, the actual timeout time for the block is 4 times 100ms plus 5s or 5.4 seconds.

Windup handling

When a windup condition is reached, the block stops calculating the integral term, but continues to calculate the proportional and derivative term.

- A windup condition exists if:
  - PID block has a secondary and the secondary is in windup.
  - PID block's output exceeds one of the user-specified output limits (OPHILM, OPLOLM).

Windup processing

Every regulatory control type block maintains anti-reset windup status for its output (ARWOP) and each of its initializable inputs (ARWNET). The following table lists the posssible values for ARWOP and ARWNET parameters.

If the Value is . . .	Then, the Associated Parameter . . .
Normal	is free to move in either direction.
Hi	is at its high limit and it may only be lowered.
Lo	is at its low limit and it may only be raised.
HiLo	may not move in either direction.

Manual Mode Interaction

When the MODE of a regulatory control block is changed to Manual (Man), the block sets its windup status (ARWNET) to HiLo. This means that every block upstream in a cascade strategy will set its windup status (ARWNET and ARWOP) to HiLo.

ARWOP computation

The ARWOP indicates if the output (OP) can be raised or lowered. The PID-type function blocks use ARWOP to restrict integral control. When ARWOP contains a value other than **Normal**, the PID block stops integral control in the windup direction. Integral control continues in the other direction, as does proportional and derivative control. But, windup status has no impact on proportional and derivative control.

If a function block has a secondary, it fetches the secondary's windup status and recomputes its ARWOP. The conditions within the function block, such as output being

at its high limit, also affect the ARWOP. The ARWOP is computed as follows, assuming the block has only one output or that it is not a FANOUT block.

If Any of the Following are True . . .	Then, ARWOP Equals . . .
This block is inactive.	**HiLo**
A secondary exists but this block cannot fetch secondary data from it (communications or configuration error).	
A secondary exists and its windup state equals HiLo	
This block is in initialization (INITMAN = On).	
A secondary exists and it is requesting this block to initialize.	
A secondary exists and its windup state equals Hi.	**Hi**
This block's output is at its high limit (OPHIFL = On).	
A secondary exists and its windup state equals Lo.	**Lo**
This block's output is at its low limit (OPLOFL = On).	

ARWNET computation

When ARWNET is HiLo, stores to SP are not limited, rather this is the status propagated to the primary. The only *limiting* anti-reset windup status ever does is to stop integral action in one or both directions on PID blocks. For any other regulatory control type block, ARWNET is not used for any kind of limiting. The ARWNET is computed as follows, assuming the block has only one output or that it is not a FANOUT block.

If Any of the Following are True . . .	Then, ARWNET Equals . . .
This block is inactive.	**HiLo**
The ARWOP equals HiLo.	

If Any of the Following are True . . .	Then, ARWNET Equals . . .
This block is in Manual mode (MODE = Man)	
The calculated value (CV) range (CVEUHI / CVEULO) is NaN.	
The CV is NaN	
This block is connected to a non-initializable primary	
The ARWOP equals Hi (Pid function blocks have a configurable Control Action option (CTLACTN). If CTLACTN = Reverse , ARWNET will track ARWOP; but if CTLACTN = Direct , ARWNET will be the opposite of ARWOP.)	**Hi**
The input from the primary is at a high limit. For example, SPHIFL = On.	
This block's output has reached its positive rate-of-change limit (OPROCPOSFL = On)	
The ARWOP equals Lo (Pid function blocks have a configurable Control Action option (CTLACTN). If CTLACTN = Reverse , ARWNET will track ARWOP; but if CTLACTN = Direct , ARWNET will be the opposite of ARWOP.)	**LO**
Input from the primary is at a low limit. For example, SPLO.FL = On.	
This block's output has reached its negative rate-of-change limit (OPROCNEGFL = On)	

Anti-Reset Windup Status

The anti-reset windup network in (ARWNETIN) and anti-reset windup output in (ARWOPIN) parameters are added in the standard anti-reset windup (ARW) computation logic. They are user configurable and allow stores from Sequential Control Modules (SCMs) and control algorithm block (CAB) programs.

The ARWNETIN and ARWOPIN parameters would be *ORed* into the existing standard logic so it is not lost. The following table summarizes the influence the ARWNETIN and ARWOPIN parameters have on the ARWNET and ARWOP parameters, which are not user configurable.

ARWNETIN or ARWOPIN Parameter Is. . .	Standard Computation Logic Is . . .	ARWNET or ARWOP Parameter Is . . .
NORMAL	NORMAL	NORMAL
NORMAL	HI	HI
NORMAL	LO	LO
NORMAL	HILO	HILO
HI	NORMAL	HI
HI	HI	HI
HI	LO	HILO
HI	HILO	HILO
LO	NORMAL	LO
LO	HI	HILO
LO	LO	LO
LO	HILO	HILO
HILO	NORMAL	HILO
HILO	HI	HILO
HILO	LO	HILO
HILO	HILO	HILO

Override feedback processing

The PIDER block does not support override feedback processing.

Error handling for RFB and TRFB inputs

- If the tracking control switch (S1) is **Off** and the reset feedback value (RFB) is bad, this block sets CV to NaN (Bad).
  - When the RFB returns to good, CV is initialized, and the dynamic PID terms are returned to steady-state.
- If S1 is **On** and the tracking value (TRFB) is bad, the block sets CV to NaN (Bad).
  - When the TRFB returns to good, CV is initialized, and the dynamic PID terms are returned to steady- state. If configured, an initialization request is sent to the primary.

Restart or point activation

Initialization takes place when the block is activated or inactivated. Initialization also takes place when the ACE controller node is repowered.

PIDER parameters

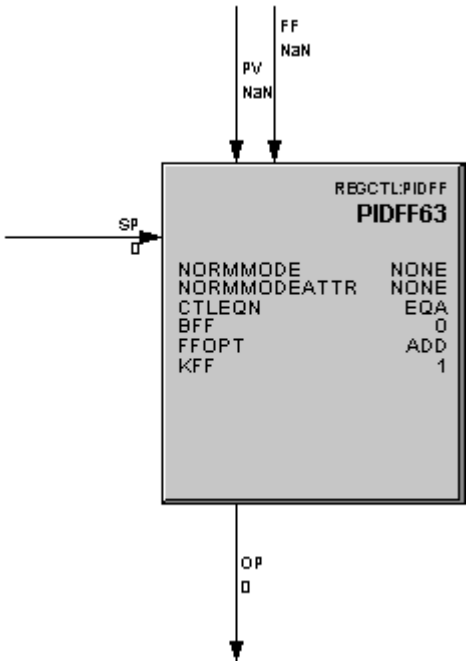
REFERENCE - INTERNAL

Refer to the *Control Builder Components Reference* for a complete list of the parameters used with the PIDER block.

PIDFF (PID with Feedforward) Block

Description

The PIDFF block is like the PID block but it accepts a feedforward signal as an additional input. You can configure the PIDFF block so the feedforward signal is added to or multiplied by the normal PID algorithm's incremental output to meet your particular control requirements. This lets you implement a feedforward control function through a single function block. The PIDFF block looks like this graphically:



The PIDFF block has three analog inputs – a process variable (PV), a set point (SP), and a feedforward signal (FF). The difference between PV and SP is the error and this block calculates a control output (OP) that should drive the error to zero. The feedforward signal (FF) is included in the calculation of the PID's incremental output before the full value output is accumulated.

The following equations are supported:

- Proportional, Integral, and Derivative (PID) on the error
- Proportional and Integral (PI) on the error and Derivative (D) on changes in PV
- Integral (I) on the error and Proportional and Derivative (PD) on changes in PV
- Integral (I) only
- Proportional (P) only

The PIDFF block may be used to provide feedforward response in a typical PID control loop application. The following figure shows a PID with feedforward controller being used with a lead/lag relay to provide dynamic feedforward control for a feed flow application. In this case, the basic idea is to measure the feed flow variations and feedforward this information to the appropriate control valve before the closed-loop system senses that the disturbance has arrived.

The lead/lag relay adds a dynamic or time variable in the feedforward circuit. It can either advance or delay a signal going through it. The “leads” and “lags” are adjustable so that a signal going in comes out varying in time over a broad range of shapes.

You can easily configure this control strategy in Control Builder using the PIDFF block in conjunction with IOCHANNEL and Auxiliary type function blocks, which include DEADTIME and LEADLAG blocks.

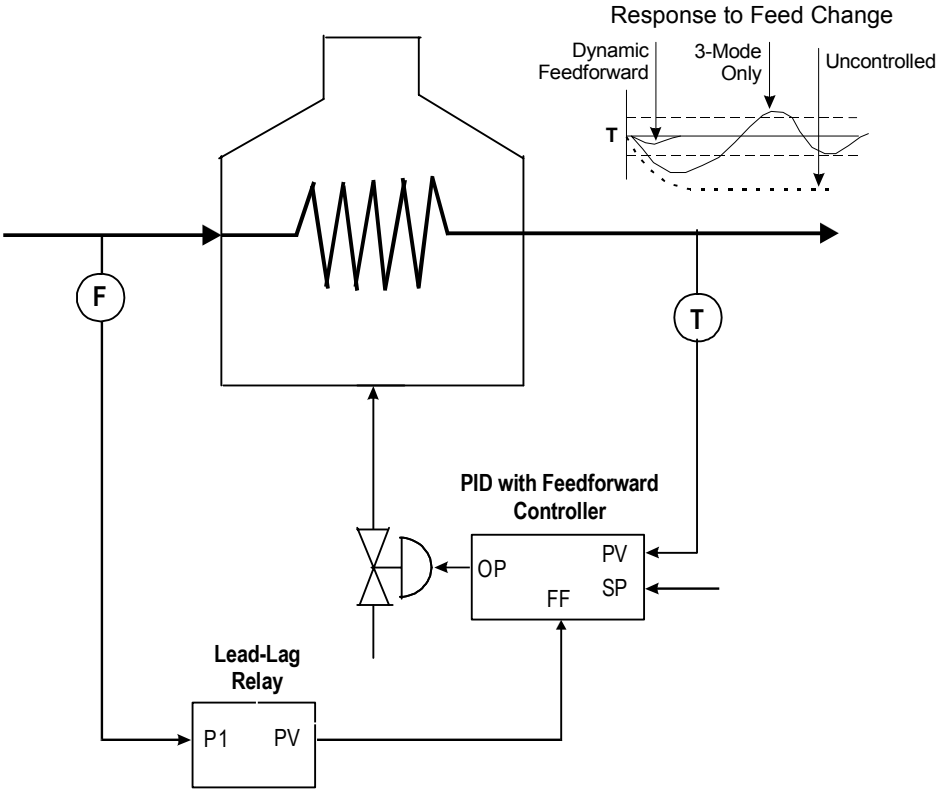


Figure 27 Simple PID with feedforward control loop example.

Each PIDFF block supports the following user configurable attributes. The following table lists the given name of the "Tab" in the parameter configuration form and then briefly describes the attributes associated with that Tab. This data is only provided as a quick document reference, since this same information is included in the on-line context sensitive Help.

Configuration Tab	Description
Main	• Name – Block (Tag) name of up to 16 characters long. Must be unique within the CM block containing it. • Description (DESC) – Block descriptor of up to 24 characters long. • Engineering Units (EUDESC) – Lets you specify a text string of up to 16 characters to identify the variable values associated with this block. For example, you could specify DEGF for temperature values in degrees Fahrenheit. This name is used on any associated displays and generated reports. • Execution Order in CM (ORDERINCM) – Specifies the execution order of the block in the CM relative to other blocks contained in this CM. Enter as a number between 1 to 32767. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. • PVEU Range High (PVEUHI) – Lets you specify the high input range value in engineering units that represents 100% full scale PV input for the block. The default value is 100. • PVEU Range Low (PVEULO) – Lets you specify the low input range value in engineering units that represents the 0% full scale PV input for the block. The default value is 0 (zero). • Manual PV Option (PVMANOPT) – Lets you specify the mode and output the block is to assume when PVSTS changes to MANual. The selections are: – NOSHED - No changes. – SHEDHOLD - Sets MODE to MANual and MODEATTR to Operator, disables external mode switching (ESWPERM), and holds output at the last good value. – SHEDLOW - Sets MODE to MANual and MODEATTR to Operator, disables external mode switching (ESWPERM), and sets output to its extended low limit (OPEXLOLM) value. – SHEDHIGH - Sets MODE to MANual and

Configuration Tab	Description
	MODEATTR to Operator, disables external mode switching (ESWPERM), and sets output to its extended high limit (OPEXHILM) value. – SHEDSAFE - Sets MODE to MANual and MODEATTR to Operator, disables external mode switching (ESWPERM), and sets output to the configured safe output (SAFEOP) value. The default selection is SHEDHOLD. The block requests its primary to initialize after a mode shed or lets the primary know that it is woundup, if it does shed its mode. An operator can change the block's mode after it is shed, but, the operator must first set PVMANOPT to NOSHED, so the mode doesn't shed again. When PVSTS returns to normal, the block clears its primary initialization request but remains in MANual mode after a mode shed. An operator must return the block to its normal mode. If mode was not shed, the block clears its windup condition and does a oneshot initialization. It also requests the primary to do a oneshot initialization. • Normal Mode (NORMMODE) – Lets you specify the MODE the block is to assume when the Control to Normal function is initiated through the Station display. Selections are MANual, AUTOmatic, CAScade, BackupCAScade and NONE. All selections are not valid for a given block. The default selection is NONE. • Normal Mode Attribute (NORMMODEATTR) – Lets you specify the mode attribute (MODEATTR) the block is to assume when the Control to Normal function is initiated through the Station display. Selections are NONE, OPERATOR and PROGRAM. The default selection is NONE. • Mode (MODE) – Lets you set the block's current MODE. The selections are MANual, AUTOmatic, CAScade, BackupCAScade, NONE, and NORMAL. All selections are not valid for a given block. The default selection is MANual. MODE identifies who may store values to the block's initializable inputs or output. Blocks strictly enforce the MODE assignment. For

Configuration Tab	Description
	example, if the MODE is CAScade and the block is getting its SP from another block, an operator is prevented from overwriting the SP value. However, If there is a breakdown in the cascade loop, the MODE can be changed so an operator can write a value to the SP. Some function blocks perform automatic mode switching (or mode shedding), while others require manual intervention. The block's MODE is derived at "runtime" based on current conditions. MODE processing checks for the following conditions, and changes the block's MODE as appropriate. – External request for MODE switching. – Safety interlock request. • Mode Attribute (MODEATTR) – Lets you set the block's mode attribute. The selections are NONE, OPERATOR, PROGRAM, and NORMAL. The default selection is OPERATOR. MODEATTR identifies who may store values to the output (OP), when the block's MODE is MANual. • Permit Operator Mode Changes (MODEPERM) – Lets you specify if operators are permitted to make MODE changes or not. The default is Enabled (checked). A store to MODE does not change the NORMMODE. • Permit External Mode Switching (ESWPERM) – Lets you specify if external MODE switching through user configured interlocks is permitted or not, if you have at least an Engineering access level. The default is Disabled (unchecked). • Enable External Mode Switching (ESWENB) – Lets you specify if external MODE switching through user configured interlocks is enabled or not, if ESWPERM is checked (Permitted). The default is Disabled (unchecked). • Enable Secondary Initialization Option (SECINITOPT) – Lets you specify if the block is to ignore initialization and override requests from the secondary or not. The default selection is Enabled (checked, do not ignore). • Safety Interlock Option (SIOPT) – Lets you specify MODE and OP block is to assume upon a safety

Configuration Tab	Description
	interlock alarm. The selections are NO_SHED, SHEDHOLD, SHEDLOW, SHEDHIGH, and SHEDSAFE. The default selection is SHEDHOLD. • Bad Control Option (BADCTLOPT) – Lets you specify MODE and OP block is to assume if CV goes BAD. The selections are NO_SHED, SHEDHOLD, SHEDLOW, SHEDHIGH, and SHEDSAFE. The default selection is NOSHED.
Algorithm	• Control Equation Type – Lets you select the control equation the block is to use. The selections are EQA, EQB, EQC, EQD, and EQE. See the PID Equations section for this block for details. The default selection is EQA. • Control Action – Lets you specify if the block is to provide direct or reverse control action. The default selection is REVERSE, which means output decreases as input increases. • T1 (minutes) (T1) – Lets you set integral time to be used for the integral term in the control equation. • T1 High Limit (minutes) (T1HILM) – Lets you define the high limit value in minutes for the integral time setting. The default limit is 1440. • T1 Low Limit (minutes) (T1LOLM) – Lets you define the low limit value in minutes for the integral time setting. The default limit is 0. • T2 (minutes) (T2) – Lets you set the derivative time to be used for the derivative term in the control equation. • T2 High Limit (minutes) (T2HILM) – Lets you define the high limit value in minutes for the derivative time setting. The default limit is 1440. • T2 Low Limit (minutes) (T2LOLM) – Lets you define the low limit value in minutes for the derivative time setting. The default limit is 0. • High Gain Limit (GAINHILM) – Lets you set a high limit for the gain (K) value. If this value is exceeded, K is clamped to this limit. The default value is 240. • Low Gain Limit (GAINLOLM) – Lets you set a low limit

Configuration Tab	Description
	for the gain (K) value. If K is less than this value, it is clamped to this limit. The default value is 0. • Gain Options (GAINOPT) – Lets you select the type of gain term to be used in the PID equation. The default value is LIN. The selections are: – LIN - The LINear gain option provides a proportional control action that is equal to a constant (K) time the error (PV -SP). This applies to equations A, B, and C. – GAP - The GAP gain option reduces the sensitivity of the control action when the PV is in a user-specified band (gap) around the set point. This applies to equations A, B, and C. – NONLIN - The NONLIN gain option provides a proportional control action that is equal to the square of the error, rather than the error itself. This applies to equations A, B, and C. – EXT - The EXTernal gain option modifies the gain (K) by an input value from either the process, another function block, or a user program. You can use this option to compensate for nonlinear process gain - lets you tune the PID gain independent of the normal operating point of the process. For example, If you are controlling the level of tank whose cross-section is not constant, you can use the EXT option to modify the gain to compensate for the nonlinear rate of level change, which is caused by the changing shape of the tank. This applies to equations A, B, and C. • Overall Gain (K) – Lets you set the overall gain value used to calculate the proportional term in the PID equation. The default value is 1. • Gap High Limit (GAPHILM) – Lets you define the high limit value in PV engineering units to be used when calculating GAP gain. • Gap Low Limit (GAPLOLM) – Lets you define the low limit value in PV engineering units to be used when calculating GAP gain.

Configuration Tab	Description
	• Gap Gain Factor (KMODIFGAP)– Lets you specify the value to be used for calculating overall gain (K) when the PV input is within the user specified band (GAPLOLM - GAPHILM) around the SP. The value range is 0.0 to 1.0. • Linear Gain Factor (KLIN) – Lets you specify the value to be used for calculating the overall gain (K) in association with GAP, NONLIN, or EXT gain option. • Non-Linearity Form (NLFORM) –Lets you specify the non-linearity form (0 or 1) to be used for calculating gain (K) based on the formula shown for Non Linear Gain in the Gain Options section. The default value is 1. • Non-Linear Gain Factor (NLGAIN) –Lets you specify the non-linear gain value to be used for calculating gain (K) based on the formula shown for Non Linear Gain in the Gain Options section. The default value is 0. • External Gain Factor (KMODIFEXT) – Lets you specify an input value from either the process, another function block, or a user program to be used to modify the gain (K) calculation per this formula: K = KLIN * KMODIFEXT See External Gain in the Gain Options section for more details. The default value is 1. • Legacy Gap: An option which allows the user to revert back to older GAP and Nonlinear gain calculations so that there is no change in behavior after migration.from TPS to Experion The option when enabled will allow the Experion controllers to have the same behavior as the TPS xPM controller See LEGACYGAP Option in R300. • Feedforward Type (FFOPT) – Lets you specify whether the feedforward signal is to be added to (ADD) or multiplied by (MULTIPLY) the incremental PID output. The default value is ADD. • Gain (KFF) – Lets you specify the desired gain for the feedforward input. The default setting is 1. • Bias (BFF) – Lets you specify the desired bias value

Configuration Tab	Description
	for the feedforward input. The default setting is 0.
SetPoint	• SP (SP) – Lets you specify an initial set point value. The default value is 0. • High Limit (SPHILM) – Lets you specify a high limit value for the SP. If the SP value exceeds this limit, the block clamps the SP to the limit value and sets the SP high flag (SPHIFL). The default value is 100. • Low Limit (SPLOLM) – Lets you specify a low limit value for the SP. If the SP value falls below this limit, the block clamps the SP to the limit value and sets the SP low flag (SPLOFL). The default value is 0. • Mode (TMOUTMODE) – Lets you select the desired MODE the block is to assume, if an initializable input times out, which means the input has not been updated within a designated timeout time. The selections are AUTOmatic, BCAScade, CAScade, MANual, NONE, and NORMAL. The default selection is MANual. • Time (TMOUTTIME) – Lets you specify a time in seconds that must expire before the block assumes that its input update has timed out. The block must be in CAScade mode for it to monitor its primary input for timeout. The default setting is 0, which means the timeout function is disabled. If the input is from a connection in another controller in a peer-to-peer architecture, the actual timeout time equals the configured TMOUTTIME plus the CDA timeout time. The CDA timeout time equals four times the configured CEE subscription rate. For example, if the CEE subscription rate is 100 milliseconds and the TMOUTTIME is 5 seconds, the actual timeout time for the block is 4 times 100ms plus 5s or 5.4 seconds. • Enable Advisory SP Processing (ADVDEVOPT) – Lets you specify whether or not the block is to generate a deviation alarm when the PV deviates from a user specified “advisory” SP value. The default selection is unchecked (Disabled). • Advisory SP Value (ADVSP) – Lets you set an advisory SP value in PV engineering units, when Advisory SP Processing is enabled. When PV exceeds

Configuration Tab	Description
	or deviates from this value, the block generates an advisory deviation alarm. • Enable PV Tracking (PVTRAKOPT) – Lets you specify if PV tracking is to be applied to this block or not. When PV tracking is enabled, this option sets the SP equal to PV when the operation of a cascade loop is interrupted by either initialization, operator or program operation (such as, setting the MODE to MANual). This option is normally enabled for PIDs in a cascade loop. The default selection is unchecked (disabled). See the PV tracking section for this block for more details. • Enable PV Tracking in Auto/Init (PVTRAKOPTAI) – Lets you specify if PV Tracking option for the specific case when the block is in Auto mode and it is undergoing initialization (INITMAN is ON) is be enabled or not. The default selection is unchecked (disabled). • Enable SP Ramping (SPTVOPT) – Lets you specify if an operator can initiate a set point ramp action or not. It provides a smooth transition from the current set point value to a new one. The default selection is box unchecked (disabled). See the Set point ramping section for this block for more details. • Normal Ramp Rate (SPTVNORMRATE) – Lets you specify a ramp rate in engineering units per minute for the SP ramping function, when it is enabled. This lets an operator start the SP ramping function without specifying a ramp time. The default selection is Not a Number (NaN). See the Set point ramping section for this block for more details. • Max. Ramp Deviation (SPTVDEVMAX) – Lets you specify a maximum ramp deviation value in engineering units per minute for the SP ramping function, when it is enabled. Keeps PV within the specified deviation range for a ramping SP by stopping the SP ramp until the PV input catches up with the SP value. The default value is NaN, which means no ramp deviation check is made. See the Set point ramping section for this block for more details. • Enable SP Push: (PUSHSP) – Lets you specify that

Configuration Tab	Description
	the PIDFF SP will be pushed from an Inter Cluster Gateway when the PIDFF is the secondary of a cascade that extends over two Experion clusters.
Output	• High Limit (%) (OPHILM) – Lets you specify the output high limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you enter a High Limit of 90%, the high limit in engineering units is 90% times 450 or 405 + 50 (CVEULO) equals 455. This check is not applied for a function block that is in the MANual mode. The default value is 105%. • Low Limit (%) (OPLOLM) – Lets you specify the output low limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, if the CV range is 50 to 500 and you enter a Low Limit of 10%, the low limit in engineering units is 10% times 450 or 45 + 50 (CVEULO) equals 95. This check is not applied for a function block that is in the MANual mode. The default value is -5%. • Extended High Limit (%) (OPEXHILM) – Lets you specify the output extended high limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you use the default value of 106.9%, the extended high limit in engineering units is 106.9% times 450 or 481.05 + 50 (CVEULO) equals 531.05. This check is not applied for a function block that is in the MANual mode. The default value is 106.9%. • Extended Low Limit (%) (OPEXLOLM) – Lets you specify the output extended low limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 50 to 500 and you use the default value of -6.9%, the extended low limit in engineering units is -6.9% times 450 or -31.05 + 50 (CVEULO) equals 18.95. This check is not applied for a function block that is in the MANual mode. The default value is -6.9%. • Rate of Change Limit (%) (OPROCLM) – Lets you specify a maximum output rate of change limit for both the positive and negative directions of the output in percent per minute. This lets you prevent an excessive

Configuration Tab	Description
	rate of change in the output so you can match the slew rate of the control element to the control dynamics. We recommend that you configure this value before you tune the loop, so tuning can accommodate any slow down in response time caused by this rate limiting. This check is not applied for a function block that is in the MANual mode. The default value is Not a Number (NaN), which means no rate limiting is applied. • Minimum Change (%) (OPMINCHG) – Lets you specify an output minimum change limit as a percent of the Calculated Variable range (CVEUHI – CVEULO). This lets you define how much the OP must change before the function block outputs a new value. It filters out changes that are too small for the final control element to respond to. This check is not applied for a function block that is in the MANual mode. The default value is 0, which means no change limiting is applied. • Safe OP (%) (SAFEOP) – Lets you specify the safe output value as a percent of the Calculated Variable range (CVEUHI – CVEULO). For example, If the CV range is 0 to 500 and you enter a Safe OP of 50%, the safe output value in engineering units is 50% times 500 or 250. The default value is Not a Number (NaN), which means the OP is held at its last good value. ' • CVEU Range High (CVEUHI) –Lets you specify the high output range value in engineering units that represents 100% full scale CV output for the block, if the block has no secondary. The default value is 100. • CVEU Range Low (CVEULO) – Lets you specify the low output range value in engineering units that represents the 0 full scale CV output for the block, if the block has no secondary. The default value is 0 (zero). • Output Bias (OPBIAS.FIX) – Lets you specify a fixed bias value in engineering units that is added to the Calculated Variable (CV) output value. See the Output Bias section for this function block for details. The default value is 0, which means no value is added. • Output Bias Rate (OPBIAS.RATE) – Lets you specify an output floating bias ramp rate in engineering units per minute. This bias rate is only applied when the

Configuration Tab	Description
	floating bias in non-zero. See the Output Bias section for this function block for details. The default value is Not a Number (NaN), which means no floating bias is calculated. As a result, if the primary does not accept the block's initialization value, a bump in OP occurs.
Alarms	• Type – Identifies the types of alarms this block supports. Of course, these alarms also interact with other block configuration options such as the Safety Interlock Option (SIOPT) and Bad Control Option (BADCTLOPT). The types are: – OP High (OPHIALM.FL) – OP Low (OPLOALM.FL) – Deviation High (DEVHIALM.FL) – Deviation Low (DEVLOALM.FL) – Advisory Deviation (ADVDEVALM.FL) – Safety Interlock (SIALM.FL) – Bad Control (BADCTLALM.FL) • Enable Alarm (ADVDEVOPT and SIALM.OPT) – Lets you enable or disable Advisory Deviation and/or Safety Interlock alarm types. A check in the box means the alarm is enabled. The default selections are unchecked or Disabled for Advisory Deviation and checked or Yes (enabled) for Safety Interlock. You can also configure the ADVDEVOPT and SIALM.OPT parameters as a block pins, configuration and/or monitoring parameters so they appear on the block in the Project and Monitoring tree views, respectively. • Trip Point – Lets you specify the following trip points for the given alarm. The default value is NaN, which disables the trip point. • OPHIALM.TP (Output High Alarm Trip Point) • OPLOALM.TP (Output Low Alarm Trip Point • DEVHIALM.TP (Deviation High Alarm Trip Point)

Configuration Tab	Description
	• DEVLOALM.TP (Deviation Low Alarm Trip Point) • ADVDEVALM.TP (Advisory Deviation Alarm Trip Point) • Priority – Lets you set the desired priority level individually for each alarm type (OPHIALM.PR, OPLOALM.PR, DEVHIALM.PR, DEVLOALM.PR, ADVDEVALM.PR, SIALM.PR, BADCTLALM.PR,). The default value is LOW. The levels are: – NONE - Alarm is neither reported nor annunciated. – JOURNAL - Alarm is logged but it does not appear on the Alarm Summary display. – LOW, HIGH, URGENT - Alarm is annunciated and appears on the Alarm Summary display. • Severity – Lets you assign a relative severity individually for each alarm type (OPHIALM.SV, OPLOALM.SV, DEVHIALM.SV, DEVLOALM.SV, ADVDEVALM.SV, SIALM.SV, BADCTLALM.SV) as a number between 0 to 15, with 15 being the most severe. This determines the alarm processing order relative to other alarms. The default value is 0. • Deadband Value (ALMDB) – Lets you specify a deadband value that applies to all analog alarms to prevent nuisance alarms due to noise at values near the trip point. The default value is 1. Note that this value is loaded to the individual alarm deadband parameters (for example, OPHIALM.DB and OPLOALM.DB) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB. • Filter Time (ALMTM) – Lets you specify a time in seconds to define how long an analog alarm must exist before it is set true. The default value is 0, which means the alarm is set true as soon as the value exceeds the deadband value. Note that this value is loaded to the individual alarm filter time parameters (for example, OPHIALM.TM and

Configuration Tab	Description
	OPLOALM.TM) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB. • Deadband Units (ALMDBU) – Lets you specify if the deadband value represents percent or engineering units. The default value is percent. Note that this value is loaded to the individual alarm deadband unit parameters (for example, OPHIALM.DBU and OPLOALM.DBU) when the CM is loaded. If you configure the individual alarm parameters as Monitoring Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB.
SCM	• SCM Mode Tracking Option (MODETRACK) – Lets you select the desired Mode Tracking function for the SCM associated with this block's Control Module. It defines how the FB will set the state of the MODEATTR based upon the MODE of the SCM. See the *Sequential Control Module User Guide* for more information on this function. The default selection is ONESHOT. The selections are: – None – ONESHOT – SEMICONT – CONTRTN – CONT • Option Type – Lets you specify the action the function block is to take when the SCM goes into an abnormal state. The Starting State Option (STARTOPT) applies when the SCM state is Checking, Idle, or Complete. The Stop/Abort State Option (STOPOPT) applies when the SCM state is Stopping or Stopped, Aborting or Aborted. The Hold State Option (HOLDOPT) applies when the SCM state is Holding or Hold. The Restart State Option (RESTARTOPT) applies when the SCM state is Resume or Run. The NONE and LASTREQ are the only selections for the Restart State Option.

Configuration Tab	Description
	You can select from one of these types for the other options as applicable for the given regulatory control function block: – NONE - No changes. – MAN - Set MODEREQ = MANUAL. – AUTO - Set MODEREQ = AUTOMATIC. – CAS - Set MODEREQ = CASCADE. – FIXEDOP - Set OPREQ = Configured Value. – HOLDPV - Set SPREQ = PV. – FIXED SP - Set SPREQ = Configured Value and SPRATEREQ = NaN. – RAMPEDSP - Set SPTVREQ = Configured Value and SPRATEREQ = Configured Rate. • Value (STARTVAL, STOPVAL, HOLDVAL) – Depending upon Option Type selection, lets you specify an output or set point value within the respective range. For output, within OPEXLOLM to OPEXHILM and within SPLOLM to SPHILM, for set point. The default value is NaN (Not a Number). • Rate (STARTRATE, STOPRATE, HOLDRATE) – When the RAMPEDSP option is selected, lets you specify a rate value (STARTRATE, STOPRATE, HOLDRATE) for setting the SPRATEREQ for an SP ramping function.
Block Pins	Lets you select the available parameters that you want to expose as input/output pins on the function block graphic in Control Builder.
Configuration Parameters	Lets you select the available parameters that you want to appear on the face of the function block in the Project tab in Control Builder.
Monitoring Parameters	Lets you select the available parameters that you want to appear on the face of the function block in the Monitoring tab in Control Builder.
Block Preferences	Lets you change several block-viewing preferences

Configuration Tab	Description
	including the color of the block's faceplate.
Insertion Type	Lets you include an insertion type from a CAB instances in the block. See *CAB insertion configuration considerations for regulatory control blocks* for more information

Function

A PIDFF requires up to three inputs – a process variable (PV), a set point (SP), and a feedforward (FF):

- PV is pulled from another function block.
  PV is typically pulled from a Data Acquisition (DATAACQ) function block, which performs PV limit checking and alarming. The PV is non-initializable input.
- SP is pulled from another function block, or you can store a value to it or use a value from a user program.
  If SP is pulled from a primary, the PIDFF's Mode must be Cascade; and if you store a value or use a user program, Mode must be Manual or Automatic. If Mode is Cascade, the PIDFF must perform timeout checking on SP (to make sure the primary is periodically updating it). The SP is an initializable input.
- FF is pulled from another function block.
  FF is typically pulled from a LEADLAG function block, which provides dynamic signal adjustments. The FF is a non-initializable input.

A PIDFF also has the following optional inputs. Typically, these are flags, which may be stored by the operator or user program to change the normal operation of the PID.

- ESWAUTO, ESWCAS, ESWMAN and SI – Indicates if an external source, such as a user program, wants to change the PID's Mode:
  - If ESWAUTO = On, the external source wants to change the Mode to Auto.
  - If ESWCAS = On, the external source wants to change the Mode to Cascade.
  - If ESWMAN = On, the external source wants to change the Mode to Manual.
  - If SI = On, the external source wants to invoke the PIDFF's safety interlock logic.

If a BACKCALC connection is made to the secondary, the PIDFF reads BACKCALCIN from the secondary before calculating its OP:

- BACKCALCIN is a “data container”, which means it contains many pieces of information but is accessed by a single read. Among other things, the information in BACKCALCIN indicates if the secondary is wound-up or if it wants the PIDFF to initialize.
- The BACKCALCIN/BACKCALCOUT connection for each secondary used is automatically built by Control Builder as implicit/hidden connections. This means you do not have to manually wire BACKCALC connections in Control Builder.
- The secondary builds BACKCALCIN when it receives a read request from the primary. This way, BACKCALCIN is guaranteed to contain the most current status.

Functional scenario

This scenario is based on the functional block diagram of a typical feedforward loop shown in the following figure and it assumes the following:

- The PIDFF’s Mode is AUTOmatic. As a result, SP is set by the operator or a user program.
- PIDFF pulls PV from Data Acquisition (DATAACQ) function block as shown in the following figure.
- PIDFF pulls FF from the LEADLAG function block as shown in the following figure.
- The PV, FF, and OP connections are all good which means there are no communication errors or timeouts.

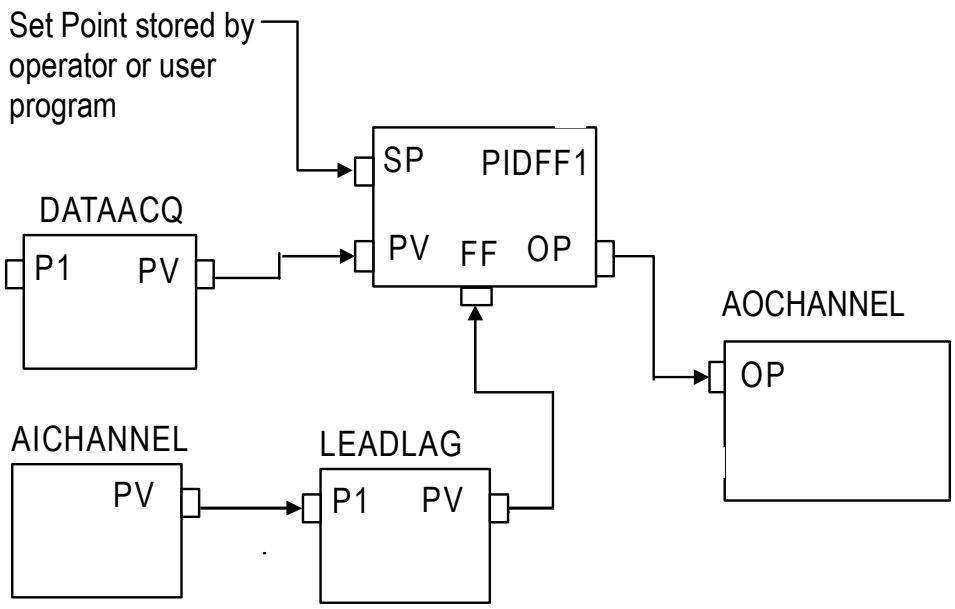


Figure 22 Functional block diagram of typical PID feedforward operation.

The functional steps associated with this PIDFF operating scenario are listed in the following table.

Step	Action
1	The Operator provides a value to the PIDFF1 SP variable (before the PIDFF1 executes).
2	The PIDFF1's "Execute" method is called by the CEE (Control Execution Environment). The PID execution period is configurable.
3	The PIDFF1 checks PVSOURCE and decides whether or not to fetch PV. If PVSOURCE = Auto, it brings PV from the DATAACQ; otherwise, it simply uses the current value of PV.
4	The PIDFF pulls the FF from the LEADLAG block.
5	The PIDFF1 checks SI, ESWAUTO, ESWCAS and ESWMAN to see if an external source wants to invoke Safety Interlock processing or change the Mode.

Step	Action
6	The PIDFF1 performs SP processing. (SP processing options are specified at configuration time.)
7	The PIDFF1 calculates an output, based on PV and SP values and the configured algorithm plus FF (FF processing option is specified at configuration time).
8	The PIDFF1 performs limit checking and alarming (if required) on OP.
9	The PIDFF1 stores OP to the AOCHANNEL.
10	The PIDFF1's "Execute" method completes.

Operating modes and mode handling

The PIDFF block operates in the following modes:

- MAN (MANual)
  - If mode is MANual, OP may be stored by the operator or a user program; PV, FF, and SP are ignored – if a primary exists, it goes to the initialized state.
- AUTO (AUTOmatic)
  - If mode is AUTOmatic, SP (or SPP) may be stored by the operator or a user program; if a primary exists, it goes to the initialized state. SP contains set point value in engineering units and SPP contains the value in percent.
- CAS (CAScade)
  - If mode is CAScade, SP is pulled from a primary; if the primary is off-control (that is, inactive or initializing) or the connection is bad, the PIDFF block invokes timeout processing.

Required inputs

The PIDFF block requires both PV and FF inputs to provide its feedforward function. The PV and FF inputs must be pulled from other blocks; you cannot store to them. Typically, they are connected to the output of an auxiliary or data acquisition (DATAACQ) block.

The SP input is not required, since it does not have to be pulled from another function block.

- If Mode is CAScade and the SP is pulled from another function block, it receives its value from an upstream primary and it is an initializable input.
- If Mode is CAScade and the SP is **not** connected to another function block, the value of the SP is frozen at the last acquired value.
- If Mode is AUTOmatic, the SP value may be stored by the operator or a user program.

The PIDFF block may have one primary or none, depending on whether SP is pulled from another block or not; there is one primary per initializable input.

Input ranges and limits

- You must specify a PV engineering unit range, PVEUHI and PVEULO.
  - PVEUHI and PVEULO define the full range of PV in engineering units. PVEUHI represents the 100% of full scale value. PVEULO represents the 0% of full scale value.
  - PVEUHI and PVEULO also define the engineering unit range of SP – PV and SP are assumed to have the same range.
- The PIDFF block assumes PV is within PVEUHI and PVEULO – it applies no range check – however, PV typically comes from a data acquisition (DATAACQ) block which applies its own limit and range check.
- SPHILM and SPLOLM define set point operating limits in engineering units.
  - The operator is prevented from storing a set point value that is outside these limits; if the primary or a user program attempts to store a value outside of the limits, the PID block clamps it to the appropriate limit and sets the primary's windup status.
- SP contains set point value in engineering units and SPP contains the value in percent.
  - If Mode is AUTOmatic, the operator or a user program may store to either SP or SPP.

Initializable outputs

"Initializable output" and "initializable input" are variable attributes, similar to data type or access level. A variable with the "initializable" attribute has an associated BACKCALC variable, and when a connection is created between an initializable input and initializable output, you can also create a BACKCALC connection. Control Builder automatically builds the required BACKCALC connections, so you don't have to create them manually. These "implicit" build connections are "hidden" from view and the related parameter pins are not exposed on the control chart.

For example, if you connect OP from a PIDFF block to a PID block or an AOCHANNEL block, Control Builder automatically creates the BACKCALCOUT to BACKCALCIN connection.

- OP = Calculated output, in percent.
- OPEU = Calculated output, in engineering units.

You may create a connection to OP or OPEU but not both. Therefore, this block may have only one secondary. If you do not create a connection to OP or OPEU, then the block does not have a secondary. Alternately, if you connect OP or OPEU to a non-initializable input, then this block does not have a secondary. (Note that the default OP connection pin is exposed on the blocks and the implicit/hidden connection function automatically makes the appropriate value/status parameter (OPX/OPEUX) connection when required. For example, if you connect the output from a primary PID block (PIDA.OP) to the set point of a secondary PID block (PIDB.SP), the implicit/hidden connection is made to PIDA.OPX to provide value/status data.)

ATTENTION

Be sure you use a FANOUT block to make multiple output connections. We recommend that you do not make multiple connections from a single PID output.

Control initialization

The PIDFF block brings initialization requests from its secondary through BACKCALC. In addition, the secondary may propagate oneshot initialization requests to this block.

- Note that SECINITOPT may be used to ignore initialization requests from the secondary.
- If the secondary is requesting initialization, the PIDFF block:
  - initializes its output
    CV = initialization value from the secondary
  - sets initialization request parameters for its primary
    INITREQ = On
    INITVAL = SP
    INITMAN = On

Output bias

If the PIDFF block algorithm is configured as Equation E, the output bias (OPBIAS) is added to the algorithm's Calculated Value (CV) and the result is stored in CV. CV is later checked against OP limits and, if no limits are exceeded, copied to the output.

If the PIDFF block algorithm is configured as Equation A, B, C, or D, no output bias (OPBIAS) is applied.

The OPBIAS is the sum of the user-specified fixed bias (OPBIAS.FIX) and a calculated floating bias (OPBIAS.FLOAT). The purpose of the floating bias is to provide a bumpless transfer when the function block initializes or changes mode as long as the PIDFF block is the first initializable block.

- OPBIAS is recomputed under the following conditions to avoid a bump in the output. (Note that the PIDFF block only applies OPBIAS.FLOAT to the output for the latter two conditions, when it is the first initializable block.)
  - When the function block starts up (that is, goes Active).
  - When the function block initializes (for example, the secondary requests initialization).
  - When the mode changes to Auto or Cascade.

ATTENTION

When the mode is Manual, OPBIAS is not used, since the output is not calculated. This means that the OPBIAS is not recomputed, when the mode changes to Manual.

- You may set the OPBIAS value only if the function block is Inactive or Mode equals Manual. This is done to prevent a bump in the output when the bias is changed. The following occurs when you set the OPBIAS value.
  - The total bias (OPBIAS) and fixed bias (OPBIAS.FIX) are both set to the entered value.
  - The floating bias (OPBIAS.FLOAT) is set to zero.

ATTENTION

When the function block goes Active or the Mode changes to Auto or Cascade, OPBIAS and OPBIAS.FLOAT are recomputed.

- There are no limit checks applied when you set an OPBIAS value. However, after the bias is added to CV, the result is compared against the output limits and clamped, if necessary.
- You configure the value for the fixed bias (OPBIAS.FIX) and it is never overwritten by the floating bias (OPBIAS.FLOAT). This means the total bias will eventually equal the OPBIAS.FIX , if you configure OPBIAS.RATE to ramp down OPBIAS.FLOAT.
- The OPBIAS.FLOAT is calculated as follows.

$$\text{OPBIAS.FLOAT} = CV_{INIT} - (CV_{UNBIASED} + \text{OPBIAS.FIX})$$

Where:

CV_{INIT} = initialization value received from the secondary

$CV_{UNBIASED}$ = unbiased calculated value (based on input from the primary)

OPBIAS.FIX = fixed bias (user-specified)

- If the primary accepts this block's initialization request, then CV + OPBIAS.FIX should be the same as CVINIT and OPBIAS.FLOAT will be zero. In most cases,

OPBIAS.FLOAT will be zero. However, if the primary does not accept this block's initialization request because the primary is a FANOUT block or it was configured to ignore initialization, then OPBIAS.FLOAT value will not be zero.

If OPBIAS.FLOAT is not zero, you can configure it to ramp down to zero through the OPBIAS.RATE parameter.

- You configure the OPBIAS.RATE to apply a ramprate to the OPBIAS.FLOAT. It is only used when the OPBIAS.FLOAT is not zero. The OPBIAS.RATE is expressed in Engineering Units per minute and may have the following values.

  – Zero:
  If OPBIAS.RATE is zero, a OPBIAS.FLOAT is calculated and bumpless transfer is guaranteed. However, if OPBIAS.FLOAT is not zero, it will never ramp down.

  – Non-zero:
  If OPBIAS.RATE is not zero, an OPBIAS.FLOAT is calculated and bumpless transfer is guaranteed. If the OPBIAS.FLOAT is not zero, it is ramped to zero at the rate you configured for the OPBIAS.RATE parameter.

  The function block ramps the OPBIAS.FLOAT to zero by applying the following calculation each time it executes.

  OPBIAS.FLOAT = OPBIAS.FLOAT – (OPBIAS.RATE / cycles_per_Min)

Where:

cycles_per_min = number of times the function block executes per minute (calculated)

  – NaN:
  When the OPBIAS.RATE is Not a Number (NaN), no OPBIAS.FLOAT is calculated. This means a bump in the output will occur, if the primary does not accept this block's initialization value.

Output ranges and limits

- CVEUHI and CVEULO define the full range of CV in engineering units.

  – If the PIDFF block has a secondary, its CV range must be the same as the secondary's input range – if this PIDFF function has a secondary, it brings the secondary's input range through BACKCALC and sets its CV range to that.

  - If the PIDFF block has no secondary, you can configure the CVEUHI and CVEULO values. The default values are 100 and 0, respectively.
  - Note that this PIDFF block brings the secondary's input range regardless of SECINITOPT (that is, regardless of whether the secondary's initialization and override data are used).

- OPHILM and OPLOLM define the normal high and low limits for OP as a percent of the CV range – these are user-specified values.
  - OP is clamped to these limits if the algorithm's calculated result (CV) exceeds them, or another block or user program attempts to store an OP value that exceeds them, however, the operator may store an OP value that is outside of these limits.
- OPEXHILM and OPEXLOLM define the extended high and low limits for OP as a percent of the CV range – these are user-specified values.
  - The operator is prevented from storing an OP value that exceeds these limits.

Output limit behavior for PIDFF block recovering from Bad Control

If PIDFF block is in Manual mode or has shed to Manual mode and is recovering from a Bad Control condition, its output clamps to the normal high and low limits (OPHILM/OPLOLM) rather than the extended high and low limits (OPEXHILM/OPEXLOLM) when the secondary block comes out of initialization.

Since the PIDFF block is in Manual mode, the output (OP) can be set to the extended limit from the Limit value, when the secondary block comes out of initialization with an initialization value (INITVAL[1..8]) higher/lower than normal output limits, respectively.

TIP

Be sure the output limits (OPHILM/OPLOLM) of the PIDFF block are the same or within the secondary's limits.

Direct or reverse control

A PIDFF block may be configured for direct-control action or reverse-control action.

Changing the control action effectively changes the sign of the gain.

- With direct-control action, an increase in the error (PV - SP) increases the PID output (CV).

- With reverse-control action, an increase in the error (PV - SP) decreases the PID output (CV).

For example, if SPP = 50% and PVP = 51%, then the error is 1%.

- With direct-control action, if PVP changes to 52%, the error increases causing CV to increase.
- With reverse-control action, if PVP changes to 52%, the error increases causing CV to decrease. .

Set Point Ramping

The Set Point Ramping option lets you ramp from the current set point value to a target set point value. You enable this option by selecting the Enable SP Ramping check box on the block's parameter configuration form. This is equivalent to setting the SPTVOPT parameter to Enable. You can also configure the following related parameters through the configuration form or the equivalent parameters.

Parameter	Description
Normal Ramp Rate (SPTVNORMRATE)	Normal ramp rate value in engineering units that you enter. The value can be Not a Number (NaN) or greater than zero. If value is NaN, it means a “step change” in the SP, which is the same as a ramp time of zero. This parameter lets you start SP ramping without specifying a ramp time. This function block calculates a ramp time (SPTVTIME) and ramp rate (SPTVRATE) as follows, when SP ramping is enabled: • If SPTVNORMRATE is a value other than zero or NaN: SPTVRATE = SPTVNORMRATE SPTVTIME = |(SPTV – SP)| / SPTVRATE • Otherwise: SPTVRATE = NaN SPTVTIME = 0 (That is, do a step change.)
Max. Ramp Deviation (SPTVDEVMAX)	Lets you specify a maximum deviation in engineering units per minute allowed between PV and SP during ramping. The value can be NaN or greater than zero. If value is NaN, it means no ramp deviation checking is done. If the maximum ramp deviation value is other than NaN, SP ramping stops when the absolute value of the deviation (|PV – SP|) exceeds the maximum deviation. The deviation flag (SPTVDEVFL) is set, and SP ramping state (SPTVSTATE) remains in Run. Ramping resumes as soon as the absolute value of the deviation returns within the maximum deviation limit. This also resets the deviation flag (SPTVDEVFL). If you have entered a ramp time (SPTVTIME) and ramping is interrupted by maximum ramp deviation, the actual ramp time (SPTVTIME) will be greater than the time you specified.

You can configure these other SP ramping related parameters to appear as block pins or monitoring parameters that can be viewed on the block during Control Builder monitoring, as shown in the following figure. You can access these parameters to invoke and monitor SP ramping while monitoring the control strategy through Control Builder or the PID Loop Point Detail display in Station.

Parameter	Description
SPTV	SP target value that you enter. You can only set SPTV when the SPTVOPT is Enabled, the SPTVSTATE is Off or Preset, and the block's mode is Auto or Manual. When you set SPTV with the block's Control Module active, this occurs: • The block calculates a ramp time (SPTVTIME). • The SPTVSTATE goes to Preset. • A "P" modifier appears next to the SP value on the PID detail display in Station.
SPTVDEVFL	SP target value deviation flag indicates when deviation exceeds the maximum ramp deviation limit.
SPTVRATE	SP target value ramp rate. This rate is calculated as shown above for the SPTVNORMRATE and as follows: • If you specify a ramp time (SPTVTIME) value other than zero: SPTVRATE = |(SPTV – SP)| / SPTVTIME Otherwise: SPTVRATE = NaN • If you change the SPTVNORMRATE, this block recalculates the ramp time (SPTVTIME) and ramp rate (SPTVRATE) as follows: If ramp time (SPTVTIME) is a value other zero: SPTVRATE = SPTVNORMRATE SPTVTIME = |(SPTV – SP)| / SPTVRATE Otherwise: SPTVRATE = NaN SPTVTIME = 0 (That is, do a step change.)
SPTVTIME	SP target value time. This time is calculated in conjunction with SPTVRATE as described above or is entered by you. You can only set SPTVTIME when the SPTVOPT is Enabled, the SPTVSTATE is Off or Preset, and the block's mode is Auto or Manual.

Parameter	Description
SPTVSTATE	SP target value state. The possible states are: • Off, • Preset, or • Run You can only set the SPTVSTATE when the Control Module containing this block is active and the block's mode is Auto. When you set SPTVSTATE to Run from Preset, this occurs: • An "R" modifier appears next to the SP value on the PID detail display in Station. • SP begins to ramp toward SPTV and SPTVTIME decreases. When SPTVTIME reaches zero, SP equals SPTV and the SPTVSTATE goes to Off.

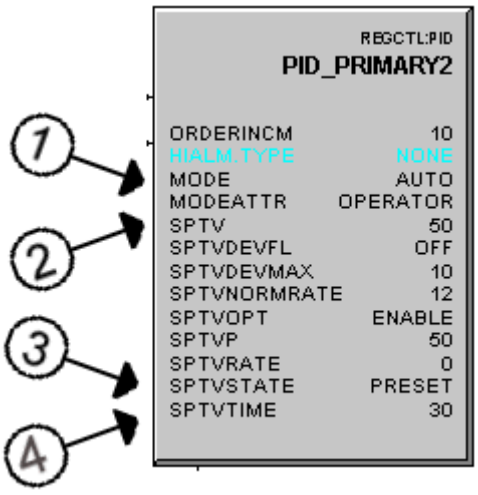


Figure 28 PID block with SP ramping parameters configured for monitoring.

The following table includes descriptions of the callouts in the figure above.

Callout	Description
1	Block's mode must be Auto and SPTVSTATE must be Preset, before you can start SP ramping by setting SPTVSTATE to Run with SPTV set to desired value.
2	The SPTVSTATE automatically goes to Preset, when: • You set a value for SPTV or SPTVTIME. • Mode changes to Manual while SPTVSTATE is Run. • Block is initialized (INITMAN = ON) while SPTVSTATE is Run. However, a oneshot initialization does not cause a change in SPTVSTATE. The SPTVSTATE automatically goes to Off, when: • SP is set by you, a program or another function block. • Mode changes to Cascade or Backup Cascade. • Control Module goes Inactive.
3	You can only set a value for SPTV and SPTVTIME, when: • SPTVSTATE is Off or Preset, and • Mode is Auto or Manual.

ATTENTION

- When SP ramping is Enabled, the SPTVSTATE must be Off before you can make changes to the SP limits (SPHILM and SPLOLM).
- If the anti-reset windup status (ARWNET) indicates that SP is woundup (Hi, Lo or HiLo), SP ramping stops. When ARWNET indicates that SP has returned to normal, SP ramping continues from where it stopped.

PV tracking

The PV tracking option sets SP equal to PV when a cascade is broken due either to function block initialization or operator or program action (such as, setting the mode to Manual).

PV tracking is configured by setting PVTRAKOPT = Track.

Typically, PV tracking is configured for PIDFF blocks in a cascade configuration strategy. This allows the PIDs to resume control with no error after initialization or when they are taken out of Manual mode.

If PV tracking is configured, the PIDFF block sets SP equal to PV (subject to SP limits) when either of the following conditions exist:

- PIDFF block is in Manual mode
- PIDFF block is initializing and not in Auto mode.

ATTENTION

- PV tracking does not occur on recovery from a bad PV.
- PV tracking does not occur if PID block is in Auto mode.
  a) If PIDFF block is in Auto mode, it means SP is normally stored by the user.
  b) If PV tracking is initiated, the stored SP value is lost.
- The PVTRAKOPTAI parameter provides a PV Tracking option for the specific case when the block is in Auto mode and it is undergoing initialization (INITMAN is On). The PV Tracking option parameter (PVTRAKOPT) does not include the Auto/Init case, but does cover several other situations as noted above.

Feedforward add or multiply action and equations

The PidFF block supports the same equations as the normal PID. In addition, the feedforward signal is applied to the PID's incremental output (DELCV), when the control equation A, B, C, or D is configured; and it is applied to the PID's full value output (CV), when the control equation E is configured.

- If the configured control equation (CTLEQN) is A, B, C, or D and the feedforward action is Additive, then:
  - If the feedforward value (FF_n) is good (symbols are defined below):

    $$CV_n = Cv_{n-1} + DELCV + KFF * (FF_n - FF_{n-1})$$

  - If the status of FF_n or FF_{n-1} is Bad:

    $$CV_n = Cv_{n-1} + DELCV$$

- If the configured control equation (CTLEQN) is E and the feedforward action is Additive, then:
  - If the feedforward value (FF_n) is good:

    $$CV_n = CV_n + KFF * FF_n$$

  (The CV_n is computed based on equation E using SP and PV and includes the OPBIAS terms.)

  - If the status of FF_n is Bad:

    $$CV_n = CV_n + KFF * FF_{LGV}$$

  (The CV_n is computed based on equation E using SP and PV and includes the OPBIAS terms. This ensures that there is no "bump" in the output, when the feedforward input goes from good to bad.)

- If the configured control equation (CTLEQN) is A, B, C, or D and the feedforward action is Multiplicative, then:
  - If the feedforward value (FF_n) is good:

    $CV_{PID} = CV_{PID} + DELCV$

    $CV = CV_{PID} * (KFF * FF_n + BFF)$

  - If the status of FF_n is Bad:

    $CV = CV_{PID} * (KFF * FF_{LGV} + BFF)$

  (**Note:** FF_{LGV} is initialized to 1.0. Therefore, if FF_n is Bad from the start, then:

  $CV = CV_{PID} * (KFF + BFF)$

  Do **not** change the KFF and BFF parameters online.)

  - If FF_n is okay but the status of FF_{n-1} is Bad, then CV is kept as is (to prevent a bump) and CV_{PID} is back-calculated as follows:

    $CV_{PID} = CV / (KFF * FF_n + BFF)$

  (Do **not** change the KFF and BFF parameters online.)

- If the configured control equation (CTLEQN) is E and the feedforward action is Multiplicative, then:
  - If the feedforward value (FF_n) is good:

    $CV = CV_{PID} * (KFF * FF_n + BFF)$

  (Where CV_{PID} is computed based on equation E using SP and PV and includes the OPBIAS terms.)

  - If the status of FF_n is Bad:

    $CV = CV_{PID} * (KFF * FF_{LGV} + BFF)$

  (**Note:** FF_{LGV} is initialized to 1.0. Therefore, if FF_n is Bad from the start, then:

  $CV = CV_{PID} * (KFF + BFF)$)

- If FF_n is ok but the status of FF_{n-1} is Bad, then CV is kept as is (to prevent a bump) and CV_{PID} is back-calculated as follows:

$$CV_{PID} = CV / (KFF * FF_n + BFF)$$

Where:

BFF = Bias of the feedforward input

CV = Full-value output with FF term included

CV_{PID} = full-value output of the PID block without the FF term (This is a calculated value and **not** a user-visible parameter.)

DELCV = Incremental output of the PID function

KFF = Gain of the feedforward input

FF = Feedforward input value

FF_{LGV} = Last good value of FF (This is a calculated value and **not** a user-visible parameter.)

$_n$ = Notation to indicate value of this pass

$_{n-1}$ = Notation to indicate value of preceding pass

Feedforward value status

If the value status of the feedforward signal goes bad, the multiplicative feedforward component of the output value is frozen at the last good value or the additive feedforward component of the output is left out, and normal PID processing continues.

When the value status of the feedforward signal returns to normal, the feedforward action resumes. This does not cause a bump in the output because any change from the last good value is internally absorbed and the PID dynamics are not affected. The floating, full-value output continues as if there were no feedforward change, but the contribution of the feedforward action continues from that point.

PID equations

The PIDFF block provides five different equations for calculating the PID – the CTLEQN parameter is used to specify the desired equation.

ATTENTION

The CV term used in the following PID equations is the same as the CV_{pid} term used in the previous feedforward equations. It represents the full value output of the PID function without the FF term added.

- Equation A – all three terms (Proportional, Integral, Derivative) act on the error (PV - SP) as follows:

$$CV = K * L^{-1}\left[\left(1 + \frac{1}{T1_S} + \frac{T2_S}{1 + a * T2_S}\right) * \left(PVP_S - SPP_S\right)\right]$$

- Equation B – the proportional and integral terms act on the error (PV - SP) and the derivative term acts on changes in PV as follows:

$$CV = K * L^{-1}\left[\left(1 + \frac{1}{T1_S} + \frac{T2_S}{1 + a * T2_S}\right) * PVP_S - \left(1 + \frac{1}{T1_S}\right) * SPP_S\right]$$

- This equation is used to eliminate derivative spikes in the control action as a result of quick changes in SP.
- Equation C – the integral term acts on the error (PV - SP) and the proportional and derivative terms act on changes in PV as follows:

$$CV = K * L^{-1}\left[\left(1 + \frac{1}{T1_S} + \frac{T2_S}{1 + a * T2_S}\right) * PVP_S - \left(\frac{1}{T1_S}\right) * SPP_S\right]$$

- This equation provides the smoothest and slowest response to SP changes.
- Equation D – integral control only as follows:

$$CV = L^{-1}\left[\frac{1}{T1_S} * \left(PVP_S - SPP_S\right)\right]$$

- Equation E – proportional only as follows:

CV = K * (PV – SP) + OPBIAS.FIX + OPBIAS.FLOAT

  - Output bias processing adds a fixed bias (user specified) and floating bias (calculated to provide bumpless transfer after initialization or mode change) to the unbiased CV.

ATTENTION

To prevent a bump in the output, you must configure the OPBIAS.RATE parameter for a value (in Engineering Units per minute) other than 0.0 (zero) or NaN (Not a Number) to enable the ramping function for the floating bias.

  - Reverse-control action causes the sign of the unbiased CV to be reversed.
  - If both options are selected, the unbiased CV is reversed first, and then the fixed and floating bias are added – neither the bias nor the final CV are reversed

Where:

CV	=	output of PID (Equations A, B, C, D) in percent or output of P-controller (Equation E only) in engineering units
K	=	gain (proportional term)
L^{-1}	=	inverse of the LaPlace transform
PV	=	process input value in engineering units
PVP	=	PV in percent
a	=	1/16 fixed rate amplitude
s	=	La Place operator
SP	=	set point value in engineering units
SPP	=	SP in percent
T1	=	integral time constant in minutes
T2	=	derivative time constant in minutes
OPBIAS.FIX	=	fixed bias (Equation E only)

OPBIAS.FLOAT = floating bias (Equation E only)

Gain options

If PID equation A, B, or C is selected, any of the following gain equations may be chosen:

- Linear Gain – provides a proportional control action that is equal to a constant (K) times the error.
  - This is the most commonly-used gain option – K is a user-specified constant and has a default value of 1.0.
- Gap Gain – used to reduce the sensitivity of the control action when PV is in a user-specified band (gap) around the set point.
  - Gap size and control action are specified at configuration time through the following parameters:

KLIN	Linear (normal) gain – to be used when PV is outside the gap.
KMODIFGAP	Gain-modification factor – to be used when PV is inside the gap. Range of KMODIFGAP = 0.0 to 1.0.
GAPLOLM	Lower limit of gap – in same engineering units as PV.
GAPHILM	Upper limit of gap – in same engineering units as PV.

  - Gain (K) is derived as follows:

When PV is outside the gap:

$$K = KLIN$$

When PV is inside the gap (SP - GAPLOLM <= PV <= SP + GAPHILM):

$$K = KLIN * KMODIFGAP$$

- Nonlinear Gain – provides control action that is proportional to the square of the error, rather than the error itself.
  - Gain (K) is derived as follows:

$$K = KLIN * \left[NLFORM + NLGAIN * \frac{|PV - SP|}{(PVEUHI - PVEULO)} \right]$$

Where:

KLIN = linear (normal) gain (user-configured)

NLFORM = nonlinear gain form (user-configured; may be 0 or 1)

NLGAIN = nonlinear gain (user-configured)

- External Gain – where, when gain (K) is selected, it is modified by an input value that can come from either the process, another function block, or a user program.
  - The main use of this option is to compensate for nonlinear process gain – you can tune the PID gain independently of the normal operating point of the process.
  - For example, in controlling the level of a tank whose cross-section is not constant, the gain could be modified to compensate for the nonlinear rate of level change that is caused by the changing shape of the tank.
  - Gain (K) is derived as follows:

$$K = KLIN * KMODIFEXT$$

Where:

KLIN = linear (normal) gain (user-configured)

KMODIFEXT = external gain modifier (such as from a user program)

Tuning Constant Change Considerations

WARNING

You cannot always reverse output (OP) resulting from changes you make to a tuning constant gain (K), integral time (T1) or derivative time (T2) in an online control loop.

You cannot undo a change in a tuning constant in an online control loop by simply changing the constant back to its original value. The output (OP) does not jump back to its original prior value just because you return the constant to its prior value. In this case, you must put the loop in MANUAL mode and set the output (OP) to the desired value before returning the loop to AUTO mode.

Timeout monitoring

If mode is CAScade, the PIDFF block performs timeout monitoring on SP – if a good SP value is not received within a predefined time (TMOUTTIME), the PIDFF block invokes timeout processing.

The maximum time between updates is specified by TMOUTTIME (in seconds)

- Enable timeout monitoring by setting TMOUTTIME to a non-zero value.
- Disable timeout monitoring by setting TMOUTTIME to zero.

Timeout processing

If mode is CAScade and SP times out, the PIDFF block does the following:

- Sets the input timeout flag (TMOUTFL)
- Keeps SP at its last good value.
- Changes the mode to a user-specified TMOUTMODE.
- Requests the primary to initialize.

The PIDFF block sets its cascade request flag (CASREQFL), if SP times out and sheds to AUTOmatic mode. This indicates that the block is waiting to return to the CAScade mode, and it will as soon as it brings a good SP value. When it receives a good SP value, the block does the following:

- Changes the mode back to CAScade.

- Updates the SP.
- Clears CASREQFL.

You **cannot** set the CASREQFL. However, it can be cleared by setting the block's MODE to MANUAL.

ATTENTION

If the input is from a connection in another controller in a peer-to-peer architecture, the actual timeout time equals the configured TMOUTTIME plus the CDA timeout time. The CDA timeout time equals four times the configured CEE subscription rate. For example, if the CEE subscription rate is 100 milliseconds and the TMOUTTIME is 5 seconds, the actual timeout time for the block is 4 times 100ms plus 5s or 5.4 seconds.

Windup handling

When a windup condition is reached, the PIDFF block stops calculating the integral term, but continues to calculate the proportional and derivative term.

- A windup condition exists if:
  - PIDFF block has a secondary and the secondary is in windup.
  - PIDFF block's output exceeds one of the user-specified output limits (OPHILM, OPLOLM).

Windup processing

Every regulatory control type block maintains anti-reset windup status for its output (ARWOP) and each of its initializable inputs (ARWNET). The following table lists the posssible values for ARWOP and ARWNET parameters.

If the Value is . . .	Then, the Associated Parameter . . .
Normal	is free to move in either direction.
Hi	is at its high limit and it may only be lowered.
Lo	is at its low limit and it may only be raised.
HiLo	may not move in either direction.

Manual Mode Interaction

When the MODE of a regulatory control block is changed to Manual (Man), the block sets its windup status (ARWNET) to HiLo. This means that every block upstream in a cascade strategy will set its windup status (ARWNET and ARWOP) to HiLo.

ARWOP computation

The ARWOP indicates if the output (OP) can be raised or lowered. The PID-type function blocks use ARWOP to restrict integral control. When ARWOP contains a value other than **Normal**, the PID block stops integral control in the windup direction. Integral control continues in the other direction, as does proportional and derivative control. But, windup status has no impact on proportional and derivative control.

If a function block has a secondary, it fetches the secondary's windup status and recomputes its ARWOP. The conditions within the function block, such as output being at its high limit, also affect the ARWOP. The ARWOP is computed as follows, assuming the block has only one output or that it is not a FANOUT block.

If Any of the Following are True . . .	Then, ARWOP Equals . . .
This block is inactive.	**HiLo**
A secondary exists but this block cannot fetch secondary data from it (communications or configuration error).	
A secondary exists and its windup state equals HiLo	
This block is in initialization (INITMAN = On).	
A secondary exists and it is requesting this block to initialize.	
A secondary exists and its windup state equals Hi.	**Hi**
This block's output is at its high limit (OPHIFL = On).	
A secondary exists and its windup state equals Lo.	**Lo**
This block's output is at its low limit (OPLOFL = On).	

ARWNET computation

When ARWNET is HiLo, stores to SP are not limited, rather this is the status propagated to the primary. The only *limiting* anti-reset windup status ever does is to stop integral action in one or both directions on PID blocks. For any other regulatory control type block, ARWNET is not used for any kind of limiting. The ARWNET is computed as follows, assuming the block has only one output or that it is not a FANOUT block.

If Any of the Following are True . . .	Then, ARWNET Equals . . .
This block is inactive.	**HiLo**
The ARWOP equals HiLo.	
This block is in Manual mode (MODE = Man)	
The calculated value (CV) range (CVEUHI / CVEULO) is NaN.	
The CV is NaN	
This block is connected to a non-initializable primary	
The ARWOP equals Hi (Pid function blocks have a configurable Control Action option (CTLACTN). If CTLACTN = Reverse , ARWNET will track ARWOP; but if CTLACTN = Direct , ARWNET will be the opposite of ARWOP.)	**Hi**
The input from the primary is at a high limit. For example, SPHIFL = On.	
This block's output has reached its positive rate-of-change limit (OPROCPOSFL = On)	

If Any of the Following are True . . .	Then, ARWNET Equals . . .
The ARWOP equals Lo (Pid function blocks have a configurable Control Action option (CTLACTN). If CTLACTN = Reverse , ARWNET will track ARWOP; but if CTLACTN = Direct , ARWNET will be the opposite of ARWOP.)	**LO**
Input from the primary is at a low limit. For example, SPLO.FL = On.	
This block's output has reached its negative rate-of-change limit (OPROCNEGFL = On)	

Anti-Reset Windup Status

The anti-reset windup network in (ARWNETIN) and anti-reset windup output in (ARWOPIN) parameters are added in the standard anti-reset windup (ARW) computation logic. They are user configurable and allow stores from Sequential Control Modules (SCMs) and control algorithm block (CAB) programs.

The ARWNETIN and ARWOPIN parameters would be *ORed* into the existing standard logic so it is not lost. The following table summarizes the influence the ARWNETIN and ARWOPIN parameters have on the ARWNET and ARWOP parameters, which are not user configurable.

ARWNETIN or ARWOPIN Parameter Is. . .	Standard Computation Logic Is . . .	ARWNET or ARWOP Parameter Is . . .
NORMAL	NORMAL	NORMAL
NORMAL	HI	HI
NORMAL	LO	LO
NORMAL	HILO	HILO
HI	NORMAL	HI
HI	HI	HI
HI	LO	HILO
HI	HILO	HILO

ARWNETIN or ARWOPIN Parameter Is. . .	Standard Computation Logic Is . . .	ARWNET or ARWOP Parameter Is . . .
LO	NORMAL	LO
LO	HI	HILO
LO	LO	LO
LO	HILO	HILO
HILO	NORMAL	HILO
HILO	HI	HILO
HILO	LO	HILO
HILO	HILO	HILO

Bypassing feedforward control action

An operator or a user program may bypass the feedforward action as follows.

- If the feedforward signal comes from an Auxiliary function block (Data Acquisition or Auxiliary Calculator), set the Auxiliary block's PVSOURCE parameter to MANual and do **not** manually change the PV. To resume feedforward action, change the PVSOURCE to AUTOmatic.
- If the feedforward signal comes from another Regulatory Control block, set the mode of the other block to MANual and do **not** manually change the output (OP), which is the FF input to the PIDFF block. To resume feedforward action, switch the other block's mode to AUTOmatic or CAScade.

Override feedback processing

Override feedback processing for the PIDFF block is the same as the normal PID block except, if multiplicative feedforward action is configured, a feedforward term is added to the output calculation as follows. Please see the Override feedback processing section for the PID Block for more information.

- If the override offset flag is Off and the PID is using either direct or reverse control action, then:

  CV = (override feedback value from secondary)

  CV_{PID} = (override feedback value from secondary) / (KFF * FF + BFF)

- If the override offset flag is On and the PID is using direct control action, then:

  CV = (override feedback value from secondary) + K * (PVP - SPP) * (KFF * FF + BFF)

  CV_{PID} = (override feedback value from secondary) / (KFF * FF + BFF) + K * (PVP - SPP)

- If the override offset flag is On and the PID is using reverse control action, then:

  CV = (override feedback value from secondary) – K * (PVP - SPP) * (KFF * FF + BFF)

  CV_{PID} = (override feedback value from secondary) / (KFF * FF + BFF) – K * (PVP - SPP)

Where:

CV = Full-value output of this function block with FF term included

CV_{PID} = full-value output of the PID block without the FF term (This is a calculated value and **not** a user-visible parameter.)

BFF = Bias of the feedforward input

K = Overall gain

KFF = Gain of the feedforward input

FF = Feedforward input value

PVP = PV in percent

SPP = SP in percent

ATTENTION

You can use SECINITOPT to ignore override requests from the secondary.

PIDFF parameters

REFERENCE - INTERNAL

Refer to Control *Builder Components Reference* for a complete list of the parameters used with the PIDFF block.

Honeywell

Honeywell International
Process Solutions
2500 West Union Hills
Phoenix, AZ 85027