

Honeywell

Experion PKS[®] Theory

Doc. No.:	EP-DCXX81
Release:	100
Last Revision Date:	6/02

PP30500A

Notices and Trademarks

**Copyright 2002 by Honeywell International, Inc.
Release 100 – June 2002**

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, and **TotalPlant** are U.S. registered trademarks.

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

Honeywell
ACS – Industry Solutions
2500 West Union Hills Dr.
Phoenix, AZ 85027
800-343-0228

About This Document

Contacts

Wide Web

The following lists Honeywell's World Wide Web sites that will be of interest to our industrial automation and control customers.

Honeywell Organization	WWW Address (URL)
Corporate	http://www.honeywell.com
Industrial Automation and Control	http://www.iac.honeywell.com

Telephone

Contact us by telephone at the numbers listed below.

Organization		Phone Number	
United States and Canada	Honeywell	1-800-343-0228	Solution Support Center
	ACS – Industry	1-800-525-7439	On-Site Service Information
	Solutions	1-800-852-3211	Training Registrar
		Experion PKS Support Program Information: 1-800-288-7491	
Asia Pacific	Honeywell Asia Pacific Hong Kong	(852) 28298298	
Pacific	Honeywell Limited Australia	(612) 9353 7000	
Korea	Honeywell Seoul, Korea	82-2-799-6318	
China	Honeywell China	86-10-65610371	
Europe	Honeywell PACE Brussels, Belgium	[32-2] 728-2111	

Latin America Honeywell (954) 845-2600
Sunrise, Florida U.S.A.

Symbol definitions

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

Symbol	Definition
	ATTENTION Used to identify information that requires special consideration.
	TIP Used to identify information that can not be classified as requiring special consideration, but as "nice-to-know."
	REFERENCE - INTERNAL Used to identify an information reference source internal to the document set.
	REFERENCE - EXTERNAL Used to identify an information reference source external to the document set.
	CAUTION Refers to the Product Manual for additional information.
	WARNING Refers the to the Product Manual for additional information.
	SHOCK HAZARD WARNING: risk of electrical shock. This symbol warns the user of a potential shock hazard where HAZARDOUS LIVE voltages greater than 30 Vrms, 42.4 Vpeak, or 60 VDC may be accessible.
	ESD HAZARD Electrostatic Discharge (ESD) hazards. 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.

Symbol**Definition**

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 Protectiveearth 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

Part I – Control Builder Components Theory

Control Builder Components	1
Some Underlying Concepts.....	1
Introduction.....	1
In the beginning or the Single-Loop Controller Reference.....	1
Experion PKS 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 Execution Environments	16
5 ms versus 50 ms	16
CEE Communication Performance.....	19
CEE/CPM Processing Resources	22
CEE/CPM Memory Resources	22
Function Block Execution Schedules	23
Schedule consideration differences.....	23
Control Module and Sequential Control Module FBs schedule.....	23
Component Function Block schedule	28
IOM FB schedule.....	28
CPM and CEE FBs schedule.....	28
Cycle overruns.....	29
Block Configuration Load Considerations	30
About load considerations	30
Data categories	30
Container and self-standing blocks load versus states.....	31
Load error messages.....	32
RAM Retention Start Up (RRSU).....	33
Memory Usage for CEE on CPM	34

Typical Requirements	34
CPU Utilization for CEE on CPM	37
CPU load categories	37
CPU utilization limits and estimates	37
 Peer-to-Peer Functionality	 39
Basic Peer-to-Peer Design Concepts	39
About Peer-to-Peer	39
Data flow models	40
Peer environments and subscription periods	41
Software Architecture for CPM/CEE	41
A word about ACE/CEE architecture	43
Implications for Control Builder Configuration	44
Multiple CPMs and ACEs	44
CEE execution and subscription rates	44
Function block support	45
Peer-to-peer connections and DEF and REF blocks	47
Peer-to-peer configuration example	47
Peer-to-peer configuration guidelines	51
 External OPC Server Support	 53
OPC Data Access	53
OPC Server function block	53
OPC client/server data flow	53
OPC Data References	55
OPC Data Name Syntax	55
Parameter connectors only	55
OPC references in expressions	56
OPC references in SCM Alias table	57
OPC Data Type Conversions	58
About data type	58
Gets conversions	58
General data conversion considerations	60
Stores conversions	62
ACE interface to TPS system as OPC server	63
ACE recognizes HCI	63
 Controller Redundancy Functionality	 65
Basic Redundancy Design Concepts	65
About Controller redundancy	65

Contents

Switchover and Secondary readiness	66
Role of the Redundancy Module	67
Role of the C200 Control Processor	68
Control Module Independence	71
CM Independence Background	71
CPU Utilization	72
Communication Bandwidth Utilization	73
Data Reference	74
Implications for Control Builder Functions.....	75
Redundant C200 CPM configuration	75
RM configuration	75
RM monitoring	76
RM/RCP dialog box	78
Main tab.....	79
Summary tab	80
RM Profiles tab	83
Configuration tab	85
Synchronization tab	87
Chassis Profiles tab	91
Display tab	93
Sever History tab	94
Sever Displays tab	96
Auto-Synchronization events	98
Control Mode Shed on Loss of I/O Functionality	99
Basic Control Mode Shed Design Concepts	99
About Control Mode shed on loss of I/O	99
How it works	99
Implications for Operation	101
Resetting mode after I/O communications are restored.	101
Allowing Redundancy synchronization with lost I/O communications.....	101
Control Builder Export and Import Functionality	103
Basic Export/Import Design Concepts	103
About Control Builder Export and Import functions.....	103
Export functionality	103
Import functionality	105
Export/Import functional overview.....	106
Some Operation Considerations	107
Export/Import usage notes	107

SCM and CM Chart Visualization Functionality	109
Basic Chart Visualization Design Concepts	109
About Chart Visualization	109
How SCM chart visualization works	109
Some SCM and CM Chart Operation Considerations	114
Detail display interaction notes	114
Process Manager Input/Output Functionality	115
I/O Link Interface	115
Seamless integration	115
I/O Functions	115
A word about Point form	115
IOP validation	115
High and Low Level Analog Input Points	116
Function	116
PV Characterization	120
Linear Conversions	120
Square root conversion	121
Thermal conversion	122
Smart Transmitter Interface Point	124
Smart Transmitter support	124
Multivariable transmitter support	124
Transmitter parameters and database access	125
STI parameter comparisons	126
Transmitter communication mode	127
STI IOP commands	128
Point states	129
STI IOP functions	130
STI PV characterization	131
STI linear conversion	132
STI square root conversion	133
STI thermal conversion	133
STI PV range checking and filtering	133
Analog Output Point	135
AO functions	135
AO direct/reverse output	136
AO output characterization	136
AO calibration compensation	137
Digital Input Point	138
DI functions	138
DI status point	140
DI PV source selection	140

Contents

DI off-normal alarming	140
Alarm delay.....	141
Event reporting	141
DI latched input point.....	141
DI sequence of events point	142
SOE definitions.....	142
SOE resolution considerations	143
DI SOE configuration considerations.....	144
Digital Output Point	147
DO functions.....	147
Pulse Width Modulated (PWM) Output Type.....	149
Status Output Type.....	150
Initialization request flag	150
 Component Categories and Types	 151
Overview.....	151
About categories.....	151
Function block types and Data Organization	151
 Regulatory Control	 153
Regulatory Control Blocks.....	153
Functional overview.....	153
Common regulatory control functions	156
AUTOMAN (Auto Manual) Block	158
Description.....	158
Function.....	167
Configuration example.....	168
Inputs.....	170
Output.....	170
Initializable inputs and outputs.....	171
Output ranges.....	171
Output bias	172
Mode Handling	174
Timeout Monitoring	174
Control Initialization	175
Secondary initialization option	176
Override feedback processing.....	176
AUTOMAN parameters	176
FANOUT Block	177
Description.....	177
Function.....	186
Configuration example.....	186
Inputs.....	187

Outputs	187
Initializable inputs and outputs	187
Output ranges	188
Output bias	189
Mode handling	191
Timeout monitoring	191
Control initialization	192
Secondary initialization option	194
Override feedback processing	194
BACKCALC processing	194
Windup processing	195
FANOUT parameters	195
OVRDSEL (Override Selector) Block	196
Description	196
Function	206
Configuration example	209
Configuration considerations	212
Inputs	213
Input ranges	213
Input descriptors	213
Initializable outputs	213
Output ranges and limits	214
Mode handling	215
Timeout monitoring	215
Timeout processing	215
Bypass processing	216
Bad input option	216
Equations	217
Input switching	217
Output bias	217
Bad CV processing	218
Control initialization	218
Restart or function block activation	219
Override feedback propagation	219
Recommendations on configuring override strategies	220
OVRDSEL parameters	220
PID Block	221
Description	221
Function	237
Functional scenario	238
Configuration examples	240
Operating modes and mode handling	244
Required inputs	245
Input ranges and limits	245
Initializable outputs	246
Control initialization	247
Output bias	247

Contents

Output ranges and limits.....	250
Direct or reverse control	250
Set Point Ramping.....	251
PV tracking	255
PID equations	256
Gain options	258
Timeout monitoring.....	260
Windup handling.....	261
Override feedback processing	261
PID parameters	262
PIDFF (PID with Feedforward) Block	263
Description.....	263
Function.....	279
Functional scenario	280
Operating modes and mode handling.....	282
Required inputs	282
Input ranges and limits	283
Initializable outputs	284
Control initialization	285
Output bias	285
Output ranges and limits.....	287
Direct or reverse control	288
Set Point Ramping.....	288
PV tracking	293
Feedforward add or multiply action and equations	294
Feedforward value status	296
PID equations	297
Gain options	299
Timeout monitoring.....	301
Windup handling.....	302
Bypassing feedforward control action.....	302
Override feedback processing	303
PIDFF parameters	304
POSPROP (Position Proportional) Block	305
Description.....	305
Function.....	316
Operating modes and mode handling.....	318
Required inputs	319
Input ranges and limits	319
Output.....	320
Initializable inputs and outputs.....	320
Output ranges.....	321
Set Point Ramping.....	321
PV tracking	326
Timeout monitoring.....	326
Equations	327
Control Initialization	328

Secondary initialization option	329
Override feedback processing	329
Raise/Lower limit switches.....	329
Bad control processing	330
POSPROP parameters.....	331
PULSECOUNT Block.....	332
Description.....	332
Function.....	334
Required inputs.....	335
Output.....	335
Initializable inputs and outputs.....	336
PULSECOUNT parameters	336
PULSELENGTH Block.....	337
Description.....	337
Function.....	338
Required inputs.....	340
Output.....	340
Initializable inputs and outputs.....	341
PULSELENGTH parameters	341
RAMP SOAK Block.....	342
Description.....	342
Function.....	354
Required inputs.....	356
Input ranges and limits.....	356
Initializable outputs	357
Output ranges and limits.....	358
Mode handling	358
Hold command.....	359
CEE idle or Control Module inactivate command	359
Profile statistics.....	360
Guaranteed ramp rate	360
Guaranteed soak time	361
Event timer functions	361
Control initialization.....	362
Override feedback processing	362
RAMP SOAK parameters	362
RATIOBIAS Block	363
Description.....	363
Function.....	372
Configuration example.....	373
Operating modes and mode handling.....	375
Required inputs.....	375
Input ranges and limits.....	376
Initializable outputs	376
Output ranges and limits.....	377
Control initialization.....	378

Contents

Output bias	379
Timeout monitoring	381
Override feedback processing	382
Windup handling	383
RATIOBIAS parameters	384
REGCALC (Regulatory Control Calculator) Block	385
Description	385
Function	396
Operating modes and mode handling	396
Inputs	396
Input ranges and limits	397
Initializable outputs	397
Output ranges and limits	398
Assignable outputs	399
Output assignment rules	400
Control initialization	401
Output bias	402
Timeout monitoring	405
Override feedback processing	406
Windup handling	407
Expressions	408
Parameters in Expressions	410
Guidelines for Writing Expressions	410
REGCALC parameters	411
REMCAS (Remote Cascade) Block	412
Description	412
Function	422
Configuration example	423
Inputs	426
Input ranges and limits	426
Input descriptors	426
Outputs	427
Output ranges and limits	427
Mode handling	428
Timeout monitoring	428
Timeout processing	428
Input switching	430
Equations	430
Output bias	431
Control Initialization	433
Override feedback processing	434
REMCAS parameters	434
SWITCH Block	435
Description	435
Function	445
Inputs	448
Input ranges and limits	448

Input descriptors	448
Initializable Outputs	448
Output ranges and limits	449
Mode handling	450
Timeout monitoring	450
Timeout processing	450
Equations	451
Bad input handling	453
Bypass processing	454
Input switching	454
Output bias	454
Error handling	457
Control initialization	458
Override feedback processing	459
SWITCH parameters	459
 UCN Interface	 461
Universal Control Network (UCN) Interface Block	461
Functional overview	461
UCNOUT	461
Description	461
About remote cascade	463
Configuration form overview	464
Input/Output	465
Configuration example	466
 Exchange Functions	 469
Exchange Function Blocks	469
Functional overview	469
REQFLAGARRAY Block	471
Description	471
Function	474
Input/Output	474
REQFLAGARRAY parameters	474
REQNUMARRAY Block	475
Description	475
Function	478
Input/Output	478
REQNUMARRAY parameters	478
REQTEXTARRAY Block	479
Description	479
Function	482

Input/Output.....	483
REQTEXTARRAY parameters	483
RSPFLAGARRAY Block	484
Description.....	484
Function.....	485
Input/Output.....	485
RSPFLAGARRAY parameters	485
RSPNUMARRAY Block.....	486
Description.....	486
Function.....	487
Input/Output.....	487
RSPNUMARRAY parameters.....	487
RSPTTEXTARRAY Block.....	488
Description.....	488
Function.....	489
Input/Output.....	490
RSPTTEXTARRAY parameters.....	490
 Auxiliary Functions	 491
Auxiliary Function Blocks	491
Functional Overview	491
Common auxiliary block functions	492
AUXCALC (Auxiliary Calculation) Block.....	493
Description.....	493
Function.....	493
Configuration example.....	494
Input	496
Output.....	496
Expressions.....	496
Parameters in Expressions.....	496
Guidelines for Writing Expressions.....	497
Assignable Outputs	498
AUXCALC parameters	498
DEADTIME Block	499
Description.....	499
Function.....	501
Input	501
Output.....	502
PV status	502
Error handling.....	502
Delay type.....	503
Delay table.....	504
Restart condition.....	505

DEADTIME parameters	505
GENLIN (General Linearization) Block	506
Description.....	506
Function.....	508
Inputs.....	508
Outputs	508
Error handling	508
GENLIN parameters	509
LEADLAG Block.....	510
Description.....	510
Function.....	512
Input.....	512
Output.....	512
PV status	512
Error handling	513
Equation	513
Time constant recommendations.....	514
Restart condition.....	514
LEADLAG parameters	514
TOTALIZER Block	515
Description.....	515
Function.....	515
Configuration example.....	516
Input.....	518
Outputs	518
TOTALIZER states	518
Accumulator target value	519
Deviation trip points	520
Equations.....	520
Accumulated value calculation	522
Error handling	523
Restart and activation	523
TOTALIZER parameters.....	523
Data Acquisition Functions	525
DATAACQ (Data Acquisition) Block	525
Description.....	525
Function.....	531
Input.....	533
Input ranges and limits.....	533
P1 status.....	533
PV Characterization.....	534
Input filtering	535
Input clamping	536
Low signal cut off.....	536

Output.....	537
PV source selection.....	537
PV status.....	537
Alarm processing.....	538
PV significant-change alarming.....	541
Bad PV alarm.....	541
DATAACQ parameters.....	542
Pulse Input.....	543
Pulse Input Block.....	543
Functional overview.....	543
PITOTALIZER Block.....	543
Description.....	543
Function.....	544
Configuration example.....	544
Input.....	546
Outputs.....	546
PITOTALIZER states.....	546
Accumulator target value.....	547
Deviation trip points.....	547
Equations.....	548
Accumulated value calculation.....	550
Error handling.....	550
Restart and activation.....	551
PITOTALIZER parameters.....	551
Device Control.....	553
DEVCTL (Device Control) Block.....	553
Description.....	553
Function.....	563
Configuration examples.....	567
Inputs.....	571
Outputs.....	571
States.....	572
State parameters and descriptors.....	573
Two-State motor input example.....	574
Valve input example.....	575
Two-Input motor example.....	576
Reversible motor input example.....	577
Four-Input two-valve example.....	577
DI to PV state map.....	579
Two-State motor with latched output example.....	579
Valve Output Example.....	579
Three-State Motor output examples.....	580

Mode and mode attribute.....	581
Safe output state.....	581
Momentary state.....	581
Local manual	582
Permissive interlocks	583
Safety Override Interlock	583
Override Interlocks.....	584
Configurable Override/Permissive Interlock Bypass	584
Alarms	585
Seal-In option	586
Initialization Manual condition.....	587
OP Initialization Option	587
Initialization Manual Condition with Safety Override Interlock, Override Interlocks, LocalMan, and OP Initialization.....	588
Initialization with Pulse Output.....	588
Initialization Request Flags.....	589
OP and DO Initialization After Load	589
Maintenance Statistics.....	589
Output requests	590
Output command.....	590
Logic override OPREQ	591
DEVCTL parameters	591
Logic Functions	593
Logic Function Blocks.....	593
Functional Overview	593
2003.....	593
AND	594
CHECKBAD	594
DELAY	594
EQ	595
FTRIG.....	595
GE	596
GT.....	597
LE	598
LIMIT	598
LT	599
MAX.....	599
MAXPULSE	600
MIN	600
MINPULSE	601
MUX.....	602
MUX-REAL	602
MVOTE.....	602
NAND	602
NE.....	603
nOON	604

NOR	604
NOT	605
OFFDELAY	605
ONDELAY	606
OR	606
PULSE	607
QOR	608
ROL	608
ROR	608
RS	609
RTRIG	609
SEL	609
SEL-REAL	609
SHL	610
SHR	610
SR	611
TRIG	611
WATCHDOG	611
XOR	612
Parameters	612
 Utility Functions	 613
Utility Function Blocks	613
Functional overview	613
FLAG Block	614
Description	614
Function	614
Input/Output	615
FLAG parameters	615
FLAGARRAY Block	616
Description	616
Function	616
Input/Output	616
FLAGARRAY parameters	616
MESSAGE Block	617
Description	617
Function	617
Configuration and Operation Considerations	619
Input/Output	619
MESSAGE parameters	619
NUMERIC Block	620
Description	620
Function	620
Input/Output	620

NUMERIC parameters	620
NUMERICARRAY Block	621
Description	621
Function	621
Input/Output	621
NUMERICARRAY parameters	622
PUSH Block	623
Description	623
Function	623
Execution Status	623
Store Status	624
PUSH parameters	624
TEXTARRAY Block	625
Description	625
Function	625
Input/Output	626
TEXTARRAY parameters	626
TIMER Block	627
Description	627
Function	627
Input/Output	627
Commands	628
TIMER parameters	628
TYPECONVERT Block	629
Description	629
Function	629
Execution status	630
Input/Output	631
TYPECONVERT parameters	631
 Sequential Control	 633
SCM (Sequential Control Module) Block	633
Description	633
Functional Overview	640
Recipe and history support	642
Configuration example	643
SCM parameters	644
TRANSITION Block	645
Description	645
Function	647
Default Invoke Transition	647
Configuration example	648
Input conditions	649

Contents

Standard input Conditions	650
Operators and Functions	651
Memory optimization for SCM expressions	651
Example condition expressions	652
Failure handling	653
TRANSITION parameters	654
STEP Block	655
Description	655
Restart address example	659
Restart address configuration considerations	660
Function	661
Configuration example	661
Outputs	662
Output contents	663
Operators and Functions	664
Memory optimization for SCM expressions	665
Example output expressions	666
Output processing and failure handling	667
STEP parameters	667
HANDLER Block	668
Description	668
Functional Overview	670
Configuration example	671
Branching	672
Looping	675
Handlers versus SCM states	677
Check Handler	678
Main Handler	678
Interrupt Handler	679
Hold Handler	679
Restart Handler	679
Stop Handler	680
Abort Handler	680
Null Handler	680
Edit Handler	680
HANDLER parameters	680
SCM Interface and CM Interaction	681
Control module interaction	681
Control module parameters	682
Regulatory control and device control parameters	684
Control devices mode reference	691
SCM execution overview	692
Scope and status of SCM execution	693
Key SCM parameters	695
SCM Modes	697
A word about SCM mode attribute	698
SCM STATE and COMMAND interaction	700

Exception handling	701
A word about SCM/CM communications error.....	706
SCM access locks	707
SCM run-time monitoring color code reference	708
 Common SCMs	 711
Functional Overview	711
Alias Table	712
Structure of the Alias Table	712
Alias Table parameters.....	713
Alias Table Configuration.....	713
SCM Expression	718
Dynamic Indirection	719
Binding.....	719
Instance Selection	721
Fail-Safe Values	723
Relationship between Common SCM and Batch Level 1 Function	724
Prevent Two Function Combination When Configuring Instance Parameters	724
Prevent Two Function Combination When Configuring SCM Option of CMs	724
Performance Requirements	725
Scenarios and Examples	726
Scenarios.....	726
Examples.....	727

Part II – Notifications Theory

Notification System	1
Design Basics and Component Identification.....	1
Overall Scheme	1
Communications model	3
Distribution model	4
Control functions summary	6
CDA functions summary	8
About Transport services.....	9
System summary	9
Alarm and Event Processing	10
Alarm indication	10
Alarm types.....	14
NDM Overview	15
NDM tag-coding scheme	17
Example of NDM tag decoding	20
Multiple displays	22
Alarm suppression	24
Event indication	27
Event types.....	28
A word about errorhandling log.....	33
Message Processing	34
Message indication	34
Message types	35
Notification Reporting and Acknowledgment	37
Reporting scheme	37
Acknowledgment scheme.....	38
Notification Recovery	39
Recovery initiators	39
Recovery routine	39
Frequently Asked Questions About Experion PKS Alarming	40

Figures

Part I – Control Builder Components Theory

Figure 1	Generic single-loop controller functions.	2
Figure 2	Simplified overview of Experion PKS 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.	27

Part II – Notifications Theory

Figure 1	Experion PKS scheme for notification generation.	2
Figure 2	Experion PKS communications hierarchy comparison.....	3
Figure 3	Notifications data flow diagram.....	5
Figure 4	High alarm summary display example.....	11
Figure 5	Example of alarm indicator field on alarm summary display.	12
Figure 6	Alarm indication summary.	13
Figure 7	Example of NDM generated tags in the Alarm Summary display.	19
Figure 8	Example of NDM generated tags in the Event Summary display.....	20
Figure 9	Typical NDM coded tag references for CNIs.	21
Figure 10	Station display interaction for data disclosure.	23
Figure 11	Event summary display example.....	28
Figure 12	Control Builder event journal entries example.....	32
Figure 13	Station Message Summary display format.	34
Figure 14	Typical STEP output configuration for configured message in a MESSAGE block.....	36
Figure 15	Typical TRANSITION condition configuration for configured confirmation message in a MESSAGE block.	36

Tables

Part I – Control Builder Components Theory

Table 1 Expression operators, functions, and strings reference	408
Table 2	723

Part I
Control Builder Components
Theory

Control Builder Components

Some Underlying Concepts

Introduction

As with any emerging technology, the Experion PKS 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 we delve into Experion PKS 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 Figure 1.

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

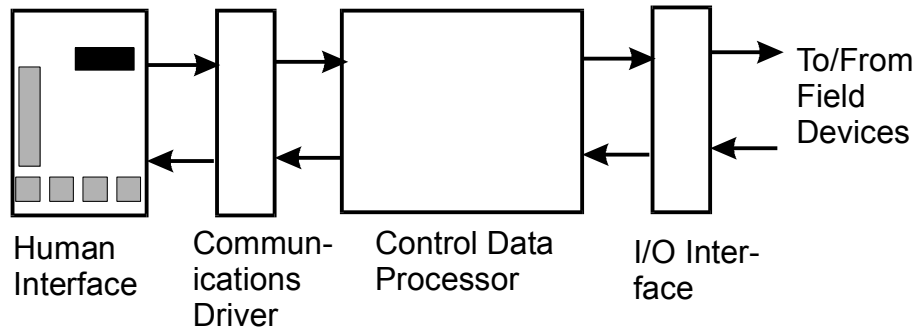


Figure 1 Generic single-loop controller functions.

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 PKS 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 PKS works. In a very simple way, Figure 2 shows how Experion PKS partitions control operations among its hardware components.

Experion PKS 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 PKS 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 PKS 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.

Control Builder Components

Some Underlying Concepts

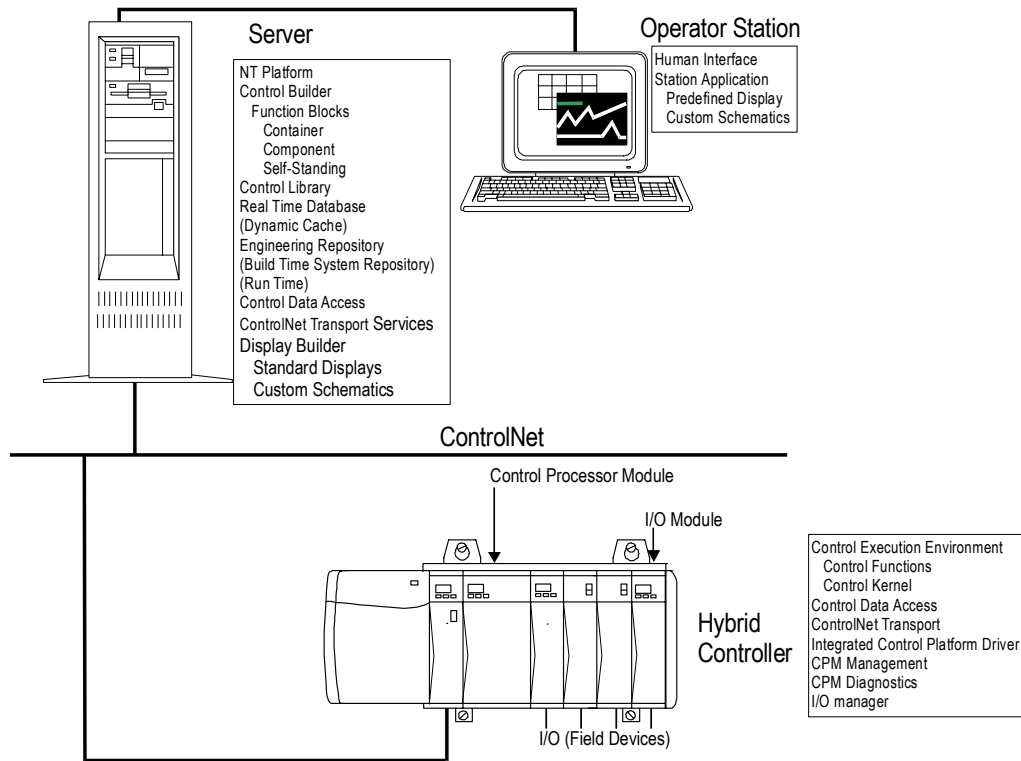


Figure 2 Simplified overview of Experion PKS architecture.

Blocks for building process control operations

If we had to use one term to sum up Experion PKS'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.

Figure 3 gives a block diagram view of how FBs are typically linked through Control Builder configuration.

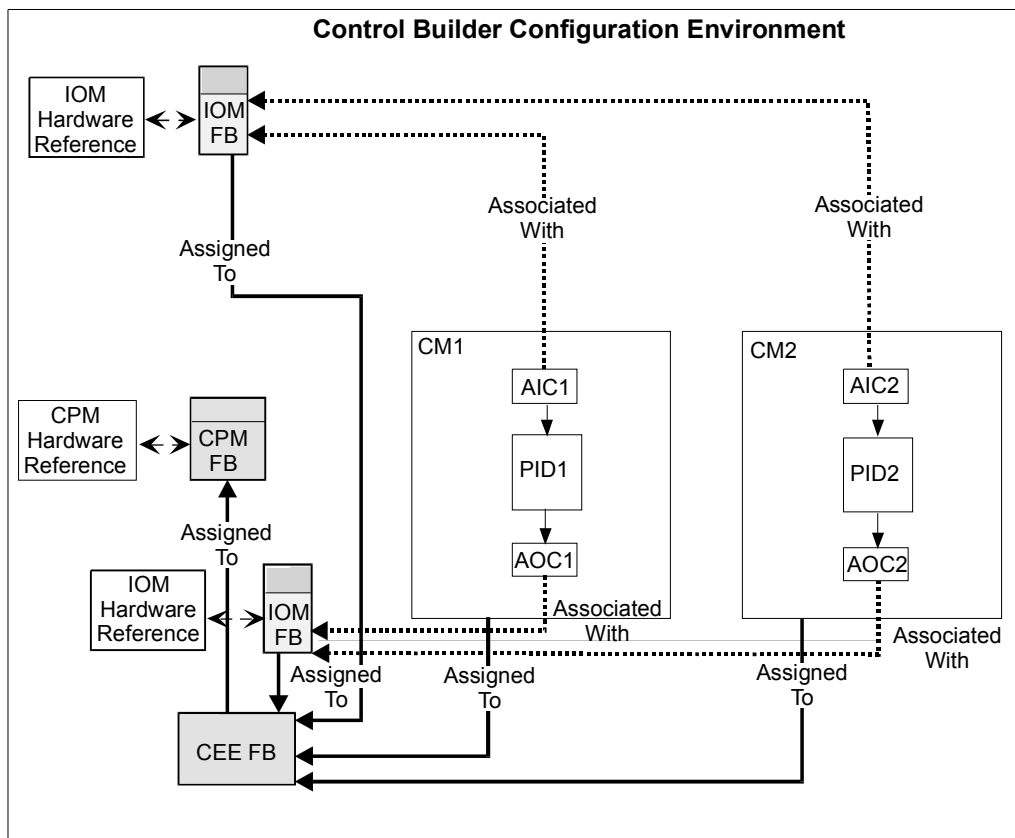


Figure 3 Typical linking of Function Blocks through Control Builder.

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 your 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 PKS 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 PKS 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 PKS 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 Figure 4. In this case, the Tag Name PIDA is dependent on the Control Module’s independent name of CM724 or CM725 for system wide recognition.

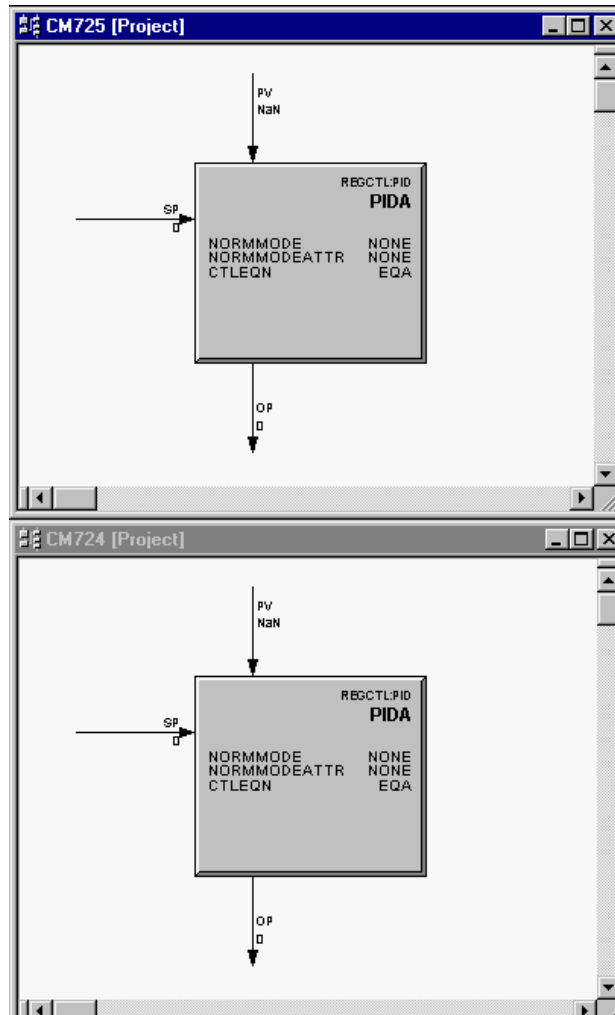


Figure 4 Component block names are dependent on container block tag name for system wide recognition.

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.

Figure 5 illustrates some typical Tag Name assignments used in a sample Control Builder configuration.

Control Builder Components

Some Underlying Concepts

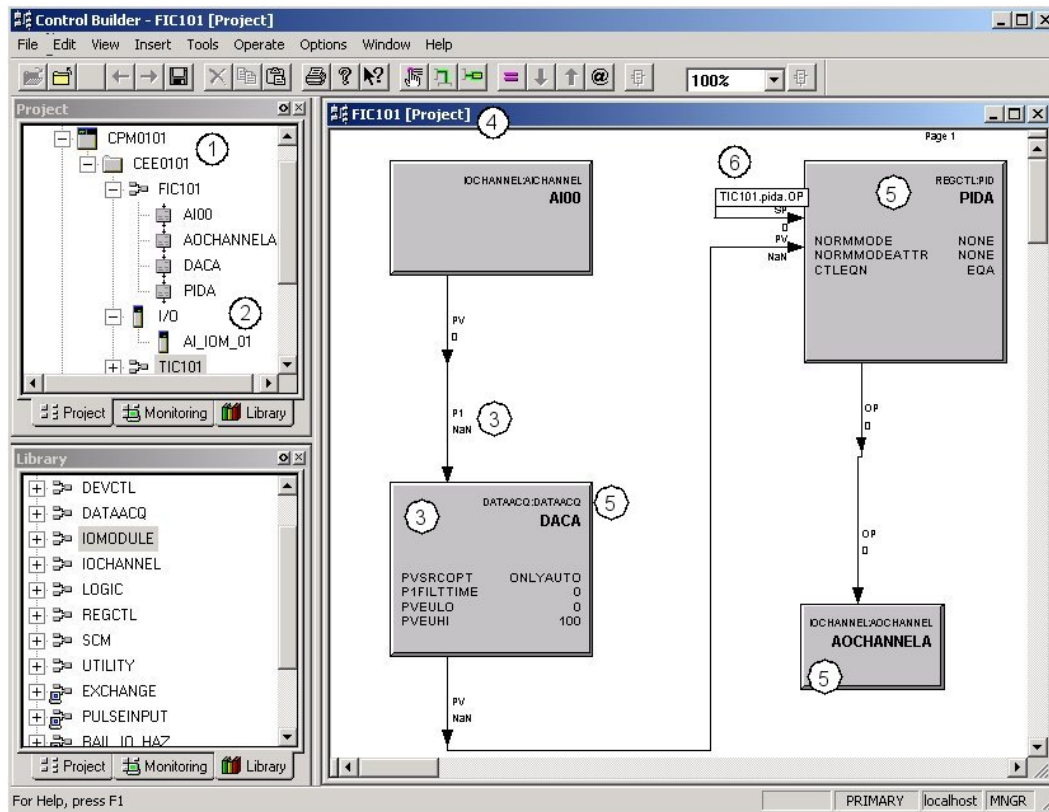


Figure 5 Sample Control Builder configuration with sample tag name assignments.

See the following table for a description of the callouts in Figure 5.

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.
 - 255 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 PKS 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 *Experion PKS 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 *Experion PKS Control Builder Parameter Reference*.

Cascade loop connections

Figure 6 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.

Control Builder Components

Some Underlying Concepts

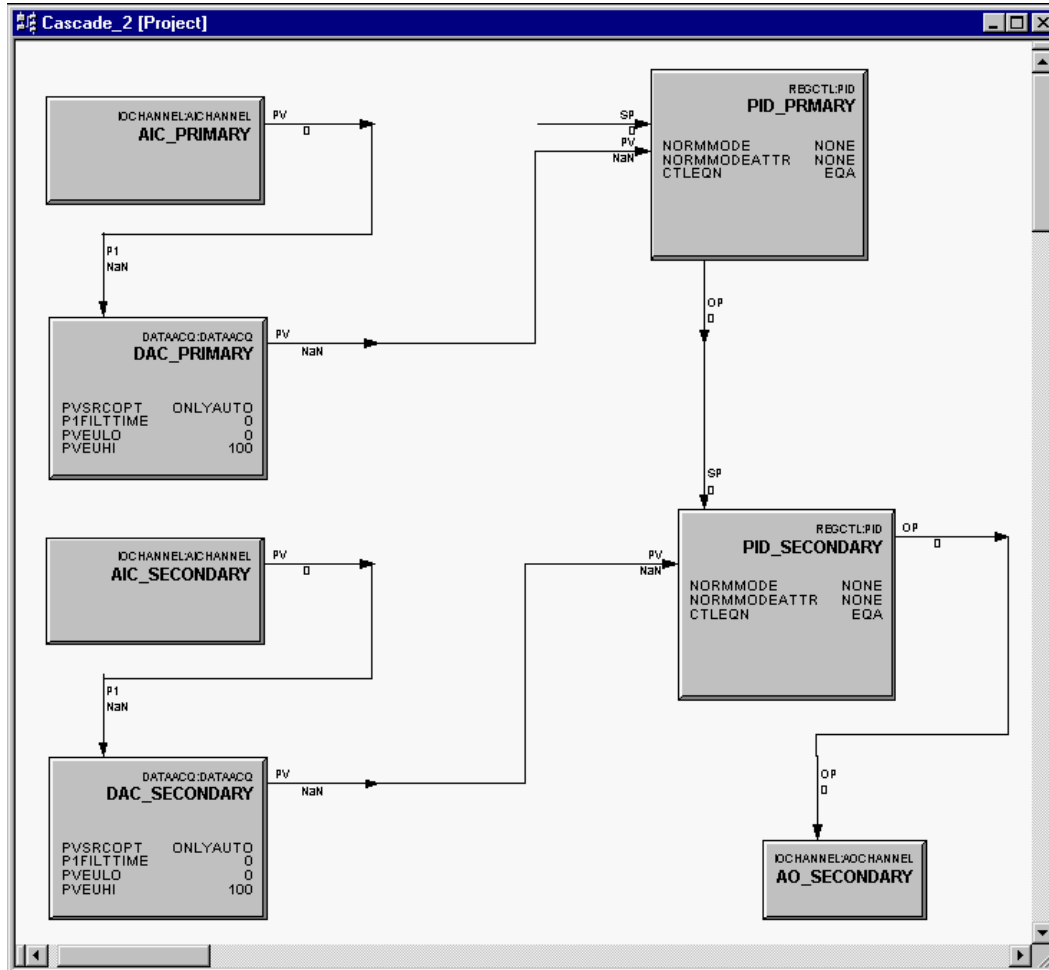


Figure 6 Sample PID cascade loop configuration .

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 Execution Environments

5 ms versus 50 ms

You have a choice of loading a 5 ms CEE or 50 ms CEE personality image in a C200 CPM for use in a non-redundant Controller architecture. You can **not** use a 5 ms CEE in a C100 CPM or a C200 CPM that is being used in a Redundant Chassis Pair (RCP).

The major difference between the 5 ms and 50 ms CEEs is the configurable execution periods. The following table lists the differences in configurable execution periods as well as summarizing the affects on other related hardware and process function configurations.

You have an additional choice of a 500 ms CEE in an ACE supervisory controller that can reference data provided by OPC servers.

Function	5 ms CEE/CPM	50 ms CEE/CPM	500 ms CEE/ACE
CM/SCM Execution Periods - Configurable	5, 10, 20, 50, 100 and 200 ms (Default is 200 ms)	50, 100, 200, 500, 1000, and 2000 ms (Default is 1000 ms)	500 ms, 1s, 2s, 5s, 10s, 20s (Default is 2s)
Peer-to-Peer Update Rates (Periods) – Configurable <i>(Defines the period at which data is updated for all “pull/get” requests for peer data required by all blocks within a CEE.)</i>	10*, 20*, 50*, 100*, 200*, 500, and 1000 ms (Default is 100 ms)	100*, 200*, 500 and 1000 ms (Default is 500 ms)	500 ms, 1s, 2s, 5s, 10s (Default is 2s) <i>(Applies to both CEE and non-CEE peers such as OPC servers of an ACE.)</i>
Controller Redundancy compatible	No	Yes	N/A
Remote I/O Supported	No	Yes	N/A
I/O Module (IOM) Execution Period	5 ms	50 ms	N/A
Publication rate for local Digital IOMs in Controller chassis	1 ms	25 ms	N/A

Control Builder Components
Control Execution Environments

Function	5 ms CEE/CPM	50 ms CEE/CPM	500 ms CEE/ACE
Maximum number of IOMs per CEE/CPM <i>(Chassis plus Rail IOMs in any combination – see exceptions noted in following entries.)</i>	12 (In Controller chassis only. Rail I/O is not supported)	64	N/A
Maximum number of FIMs per CEE/CPM <i>(Each FIM counts as 3 IOMs in the 64 Max calculation above.)</i>	N/A	21	N/A
Maximum Number of Analog IOMs per CEE/CPM – including Pulse Input Modules (PIM)	12	32	N/A
Maximum Number of Serial Interface Modules (SIM) per CEE/CPM	1	3	N/A
Maximum Number of Field Termination Assemblies (FTAs) per SIM <i>(Each FTA connected counts as 4 IOMs in the 64 Max calculation above and the 24 IOMs/CNI calculation below.)</i>	2	2	N/A
Maximum Number of remote I/O Chassis plus Rail Gateways (combined) per CEE/CPM	0 (Local I/O Only)	8	N/A
Maximum Number of IOMs per Downlink CNI – Rack plus Rail Modules in any combination, Except SIM and PIM as noted <i>(Each PIM counts as 1.5 IOMs in the 24 IOM/CNI calculation.)</i>	0 (Local I/O Only)	24	N/A
Maximum Number of Downlink CNIs per Controller Chassis	0 (Local I/O Only)	4	N/A

Control Builder Components
Control Execution Environments

Function	5 ms CEE/CPM	50 ms CEE/CPM	500 ms CEE/ACE
*Supervisory Ethernet LAN systems only guarantee a Pull/Get Request Rate of 500 ms or greater. Faster Pull/Get rates can be configured, however, data arrival at the configured rate cannot be guaranteed for rates faster than 500 ms.			

- **To obtain a pulse output duration on the order of 10 ms (+/- 1 ms)**, use only the 24 Vdc Digital Output Modules (DOMs) TC-ODJ161 and TC-ODD321. These models have an output delay time (ON-OFF and OFF-ON) on the order of 1 millisecond maximum. Do **not** use the Vac DOMs TC-ODX081, TC-ODK161, and TC-ODA161. These models have output signal delay times on the order of 10 milliseconds.
- **To obtain an end-to-end latency of 10 ms**, use only Vdc Digital Input Modules (DIMs) TC-IDD321, TC-IDJ161, and TC-IDX161 with Vdc DOMs TC-ODJ161 and TC-ODD321 to guarantee this latency. Do **not** use the Vac DIMs TC-IDA161, TC-IDK161, TC-IDW161, and TC-IDX081. These models have large input signal delays.
- Do **not** use the high density Analog Input Module (AIM) TC-IAH161 with a 5 ms CEE.

CEE Communication Performance

Since the performance requirements differ for a CPM running a 5 ms CEE, be sure you review the data in the following table before implementing a control strategy with a 5 ms CEE.

Your control strategy can also include 500 ms CEE in an ACE supervisory controller that can reference data provided by OPC servers.

Some of the ratings are specified in terms of the average Parameters Per Second (PPS)

Overall Communications Performance per CEE	5 ms CEE/CPM	50 ms CEE/CPM	500ms CEE/ACE
Maximum Total Parameter Access Response Rate <i>(Includes display, Fast/Slow History, Excel I/ODBC Exchange, and peer communications.)</i>	2000 PPS	2000 PPS	2000 PPS

CEE to CEE – Peer-to-Peer Communications Performance per CEE	5 ms CEE/CPM	50 ms CEE/CCPM	500 ms CEE/ACE
Maximum number of peer-to-peer connections to other ControlNet resident CEE type environments (CPM, FIM, IOLIM)	5 to CPMs 21 to FIMs	5 to CPMs 21 to FIMs	30
Maximum number of peer-to-peer connections as target initiated by other ControlNet resident CEE type environments (CPM, FIM, IOLIM)	5 from CPMs 21 from FIMs	5 from CPMs 21 from FIMs	30
Maximum Initiator Node Pull/Get Request Rate - To all target nodes. <i>(Based on the number of requests for process data and the response rate.)</i>	500 PPS	500 PPS	500 PPS
	ControlNet	ControlNet	ControlNet

Control Builder Components
Control Execution Environments

CEE to CEE – Peer-to-Peer Communications Performance per CEE	5 ms CEE/CPM	50 ms CEE/CCPM	500 ms CEE/ACE
<i>peer data and the peer update rate.)</i> <i>(There are no guarantees that an Ethernet network can assure the timely delivery of critical data packets. Users are cautioned to consider the non-deterministic nature of an Ethernet network before designing and configuring a mission critical control strategy to use it. An Ethernet network is subject to collisions, deferred transmissions, and other anomalies that may delay the arrival of critical data packets. The results of internal tests show that under a full load condition approximately 1 in 30,000 peer fetches arrive at a peer node at greater than twice the configured peer update rate.)</i>	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	250 @ 500 ms 500 @ 1 sec 1000 @ 2 sec 2500 @ 5 sec 5000 @ 10 sec
	Ethernet	Ethernet	Ethernet
	250 @ 500 ms 500 @ 1 sec	250 @ 500 ms 500 @ 1 sec	250 @ 500 ms 500 @ 1 sec 1000 @ 2 sec 2500 @ 5 sec 5000 @ 10 sec
Maximum Target Node Response Rate to Pull/Get Requests - From all initiator nodes.	500 PPS	500 PPS	500 PPS
Maximum Initiator Node Push/Store Request Rate - To all target nodes. (The SCM Step and Push are the only block types that can currently initiate peer push/store requests for CEE to CEE peer communications.)	50PPS	50 PPS	50 PPS
Maximum Target Node Response Rate to Push/Store Requests - From all initiator nodes.	50 PPS	50 PPS	50 PPS
Maximum Initiator OPC Pull/Get Request Rate - To all target OPC servers. <i>(There is no limit imposed on number of different OPC servers that can be accessed by the ACE/CEE.)</i>	N/A	N/A	1000 PPS

CEE to CEE – Peer-to-Peer Communications Performance per CEE	5 ms CEE/CPM	50 ms CEE/CCPM	500 ms CEE/ACE
Maximum Initiator OPC Push/Store Request Rate - To all target OPC servers. <i>(There is no limit imposed on number of different OPC servers that can be accessed by the ACE/CEE.)</i>	N/A	N/A	100 PPS
CEE to PLC – Peer-to-Peer Communications Capacity per CEE	5 ms CEE/CPM	50 ms CEE/CCPM	500 ms CEE/ACE
Maximum Number of REQUEST blocks per CEE	32	32	N/A
Maximum Number of RESPONSE blocks per CEE	32	32	N/A
Maximum Number of “active” Target Devices for REQUEST blocks per CEE.	8	8	N/A
Maximum Number of DHRIO Modules per CEE.	2 (local chassis only)	2	N/A

We recommend that you use an update rate of 100ms or greater between 5 ms CEE and 50 ms CEE environments.

CEE/CPM Processing Resources

The following table summarizes general process resources for a CEE/CPM based on a Processing Unit (PU). A PU represents a platform independent amount of processing resources (time) required to complete a predefined amount of computational (control) work. It also lists the Maximum Cycle Loading in percent, which the “Average CPU Used” (CPUCYCLEAVG) statistic is not to exceed over a cycle (0-39). (Note that the minimum reserved CPU to be maintained during run time (CPUFREEAVG) is 20 percent.)

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%
CEEACE	15000 PU/sec	

CEE/CPM Memory Resources

The memory resources and relevant block configuration capacities are as follows. The Memory Unit term represents one kilobyte or 1024 bytes.

- Maximum available CEE/CPM memory resource is 4000 MU.
- Maximum available CEE/ACE memory resource is 32000 MU.
- Maximum total number of CMs, SCMs, and IOMs configurable per CEE/CPM is 1000.
- Maximum number of component blocks per CM is 40.
- Maximum number of parameters per CM is 1000.
- Maximum total number of Steps and Transitions (in all Handlers) per SCM is 160 (80 Step/Transition pairs).

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 Figure 7. 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.

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

Control Builder Components

Function Block Execution Schedules

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
50	500	7	7, 17, 27, 37
50	500	8	8, 18, 28, 38
50	500	9	9, 19, 29, 39

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

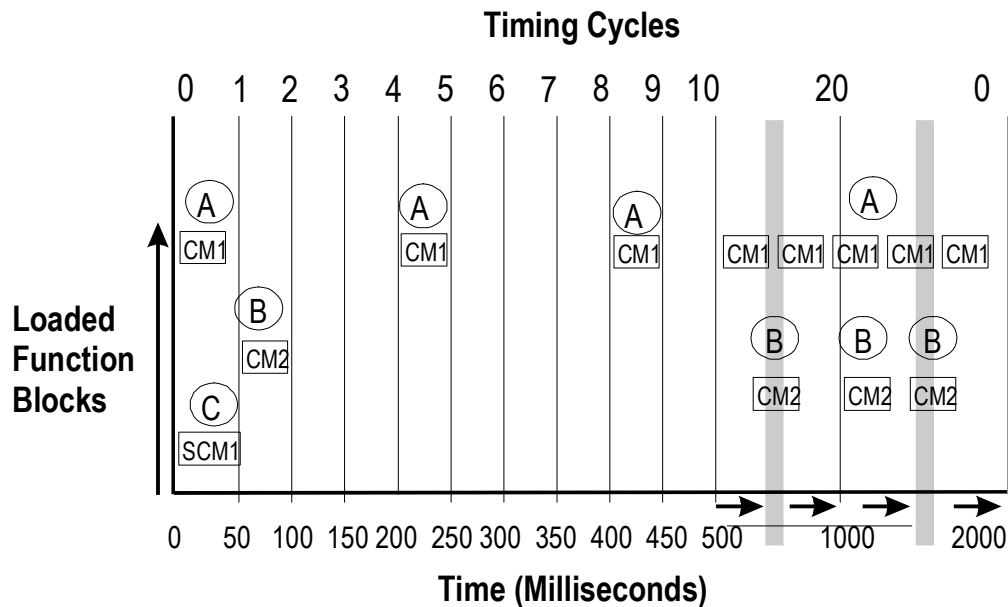
Control Builder Components

Function Block Execution Schedules

If Period in milliseconds is. . .		And Phase is . . .	Then, cycles of execution are. . .
5 ms CEE	50 ms CEE		
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 65535) 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.

Figure 7 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.



- Ⓐ 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.
- Ⓑ 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.
- Ⓒ 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.

Component Function Block schedule

Execution of component Function Blocks is always subordinate to their parent or containing SCM FB or CM FB. This means that component Function Blocks execute during the same Period as their parent container FB given that the conditions for execution are satisfied.

For Control Module FBs, you can configure the Execution Order value (0 to 65535) for each component block through its Parameters Configuration form. This determines the relative execution order of blocks within the CM, since the blocks execute in ascending order beginning with the block with the lowest Execution Order value. You can choose to display the block's Execution Order value through the block's ORDERINCM parameter. If two component FBs have the same ORDERINCM value, the Control Builder assigns the relative execution order and maintains it.



ATTENTION

Input type IOCHANNEL FBs do a sample and hold operation when they execute. They 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 IOC FBs will hold values static even if the corresponding data is changing at the IOM FB.

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 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 CEE 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 well 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 PKS 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 the *Experion PKS Notifications Theory* for more information.

Memory Usage for CEE on CPM

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 [CEE/CPM Processing Resources](#) and [CEE/CPM Memory Resources](#) 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

Control Builder Components
Memory Usage for CEE on CPM

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)
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
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

Control Builder Components
Memory Usage for CEE on CPM

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)
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 <u>total</u> 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 <u>total</u> of 10 Steps and 10 Transitions among the 4 Handlers	2.0	3.0	124.5

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 <i>Typical Requirements</i> section.

CPU utilization limits and estimates

This is based on the previous maximum loading cycle ratings listed in the table in the *CEE/CPM 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.

Peer-to-Peer Functionality

Basic Peer-to-Peer Design Concepts

About Peer-to-Peer

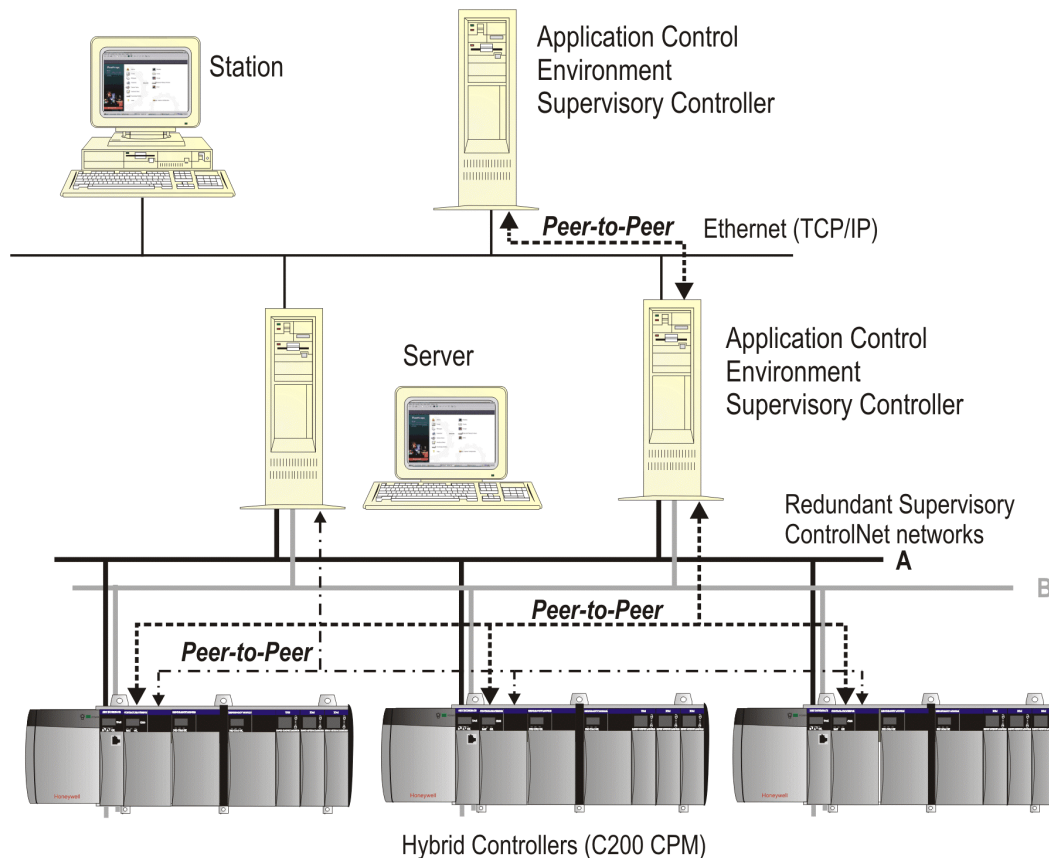
You can implement peer-to-peer communications among CEEs in multiple C200 Control Processor Modules (CPMs) networked to the same Server.

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 Ethernet (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 Ethernet, between ACE supervisory controller and Hybrid Controllers over ControlNet supervisory network, and between Hybrid Controllers over the supervisory network.

Peer-to-Peer Functionality

Basic Peer-to-Peer Design Concepts



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 response time values through the applicable CEE configuration form in Control Builder. This includes other ACE supervisory controllers, Hybrid Controllers, and external servers, such as OPC servers. The configuration form also specifies default subscription 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 cyclic "get" requests for peer references. The peer store response time expiration time parameter defines the expiration 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 PKS'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 PKS 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.

Peer-to-Peer Functionality
Basic Peer-to-Peer Design Concepts

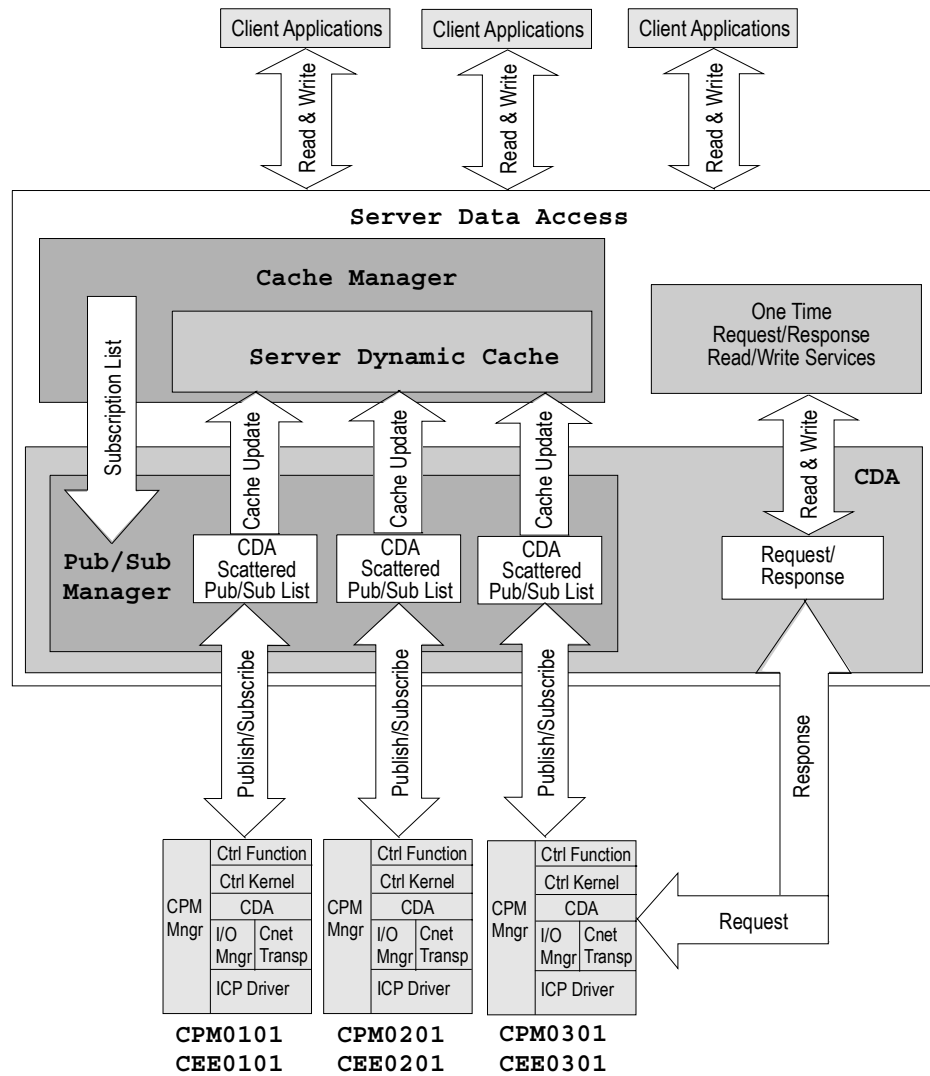


Figure 8 Peer-to-peer is layered upon CDA connection-oriented communications.

A word about ACE/CEE architecture

The overall architecture for the ACE supervisory controller is similar to the Hybrid Controller. It features separate CEE and Control Data Access-supervisory 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 Hybrid 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 Expiration 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 Expiration 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 communication 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 . . .
 HONEYWELL (Typical Fieldbus Device vendor library.)	CPM/CEE
 SYSTEM	ACE/CEE CPM/CEE (Restrictions apply - loaded strategies must contain supported blocks only.)
 AUXILIARY	ACE/CEE CPM/CEE
 DEVCTL	ACE/CEE CPM/CEE
 DATAACQ	ACE/CEE CPM/CEE
 IOMODULE	CPM/CEE
 IOCHANNEL	CPM/CEE
 PMIO	CPM/CEE
 LOGIC	ACE/CEE CPM/CEE
	ACE/CEE

Peer-to-Peer Functionality

Implications for Control Builder Configuration

If Function Block Is From This Library in Control Builder . . .	Then, It Can Be Used With This Control Environment . . .
 REGCTL	CPM/CEE
 SCM	ACE/CEE CPM/CEE
 UTILITY	ACE/CEE CPM/CEE
 PULSEINPUT	CPM/CEE
 RAIL_IO_HAZ	CPM/CEE
 PBUSIF	CPM/CEE
 RAIL_IO	CPM/CEE
 FBUSIF	CPM/CEE
 EXCHANGE	CPM/CEE
 UCNIF	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 STEP 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

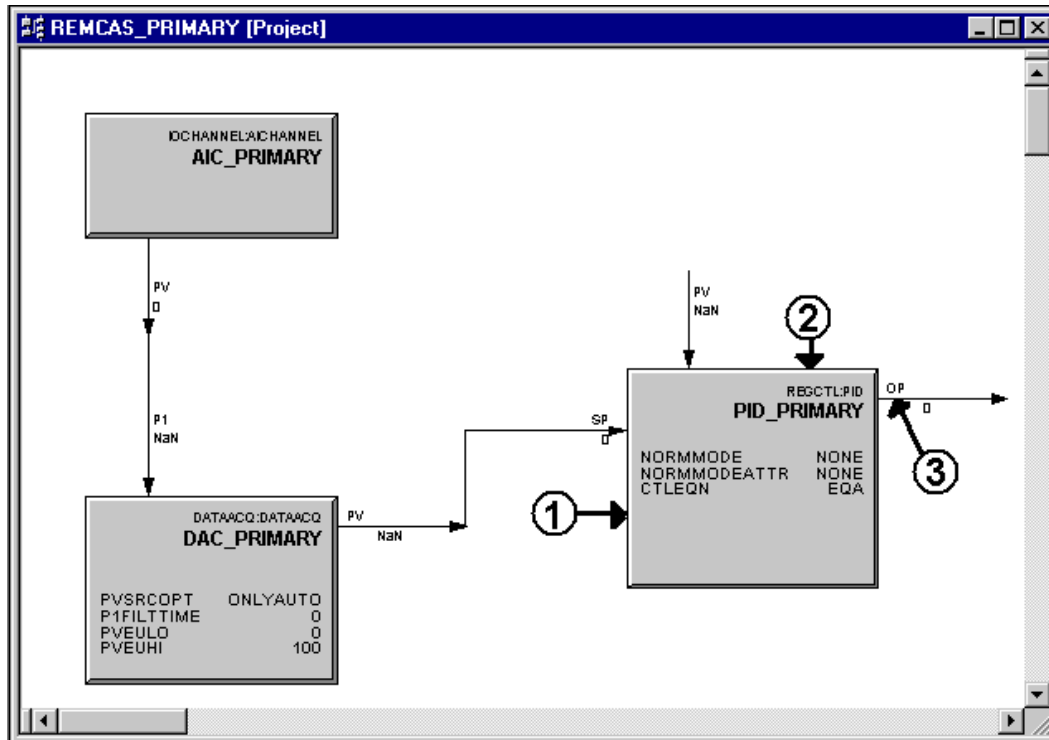
Figure 9 (Views A and B) and its companion callout description table show a sample configuration that uses a REMCAS block to form a cascade control loop with remote and local primary loops.

For this example, assume that this application includes two Hybrid 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.

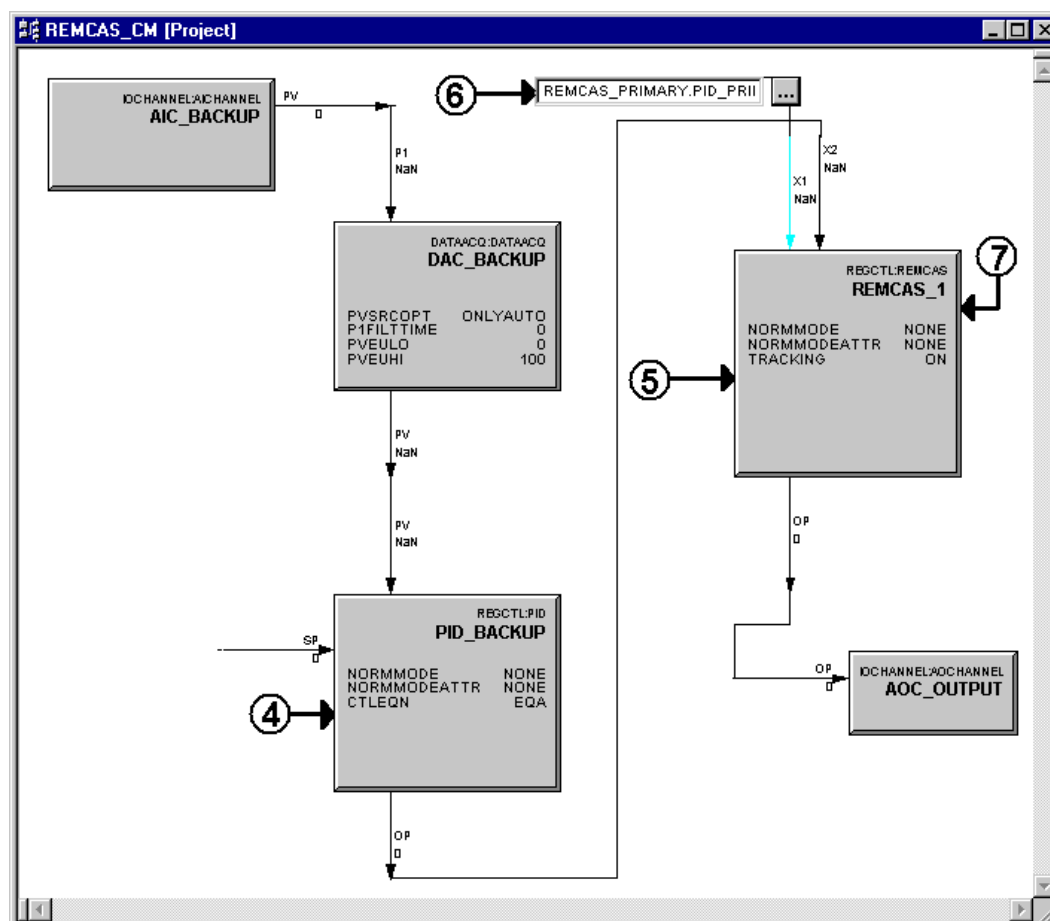
In this example, the PID_PRIMARY and REMCAS_1 blocks have both DEF and REF relationships for peer-to-peer communications.

Peer-to-Peer Functionality

Implications for Control Builder Configuration



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 Figure 9.

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 Hybrid Controllers or ACE supervisory controllers networked to the same Experion PKS 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.

Peer-to-Peer Functionality

Implications for Control Builder Configuration

- 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 example in Figure 9. This means a REF block can also be a DEF block to another DEF 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.

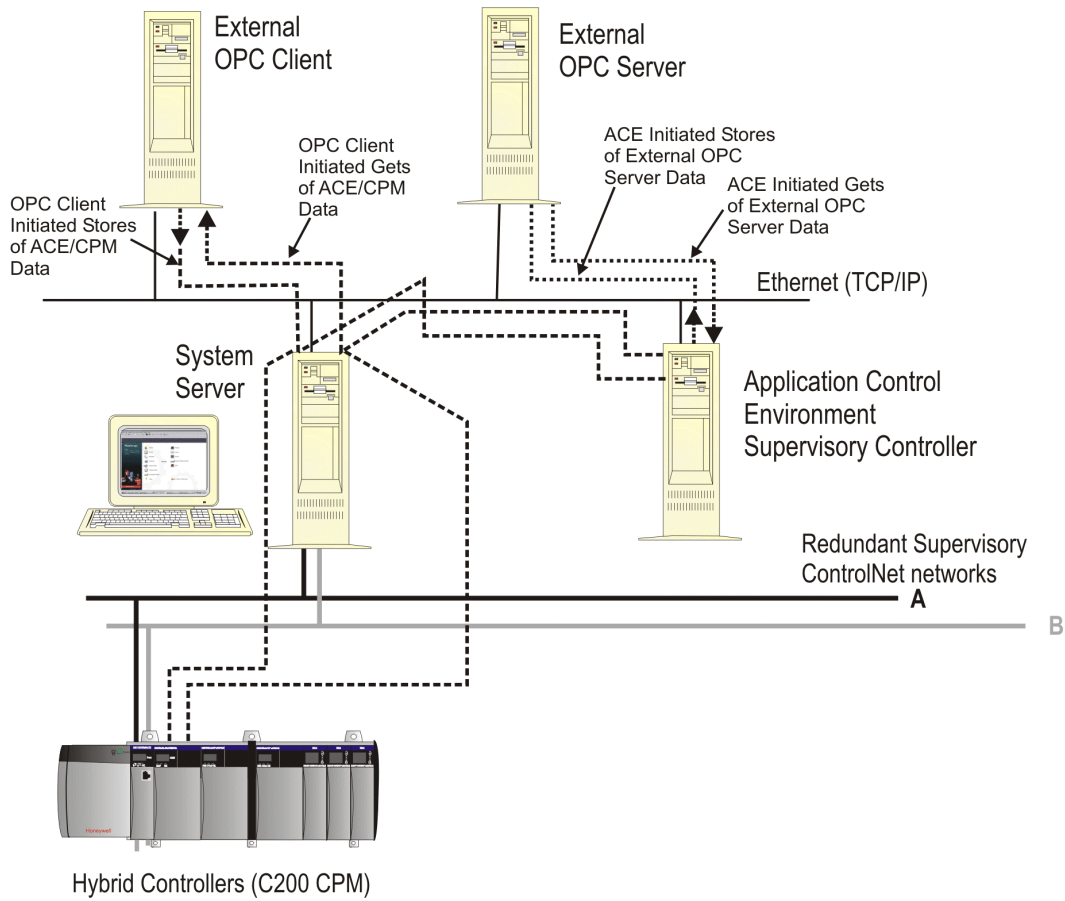
Users specify the communications path to the OPC server through the blocks configuration form. This data is stored in the System Repository when the block is loaded.

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 Hybrid Controller does not support an OPC client, but OPC clients can access its data through the OPC server of the system Server.

External OPC Server Support

OPC Data Access



OPC Data References

OPC Data Name Syntax

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

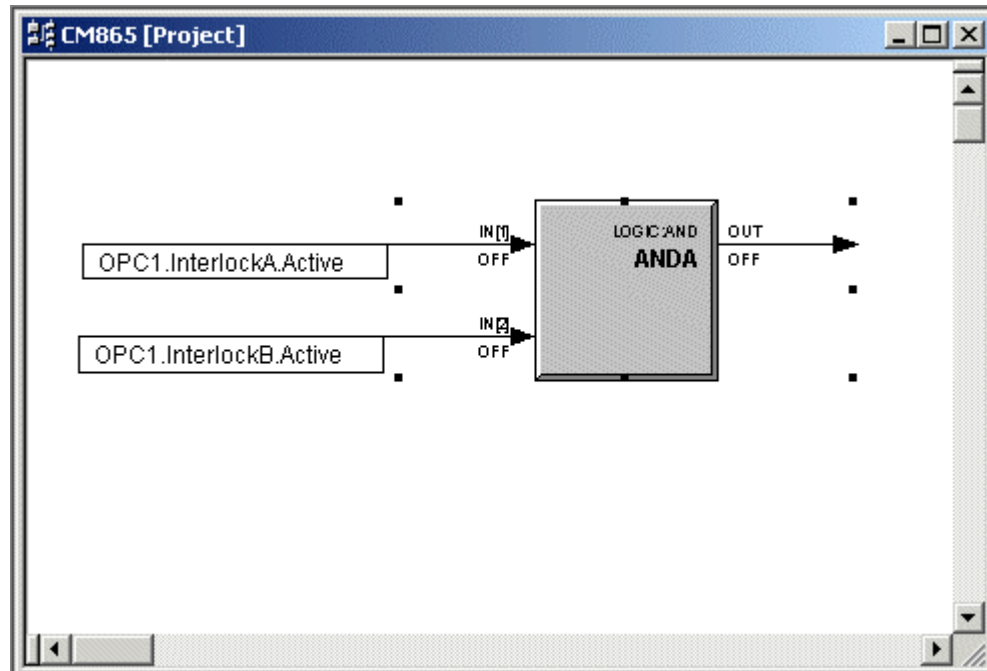
<OPC server function block name>.<OPC server data specific name>

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 EXTREFSTRUCT 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.

SYSTEM:SCM Block, example_scm - Parameters [Project]

Main | Handlers | Alarm | Recipe | History | Aliases | Security/History | Displays | Status

☒ Enable Alias Configuration

Number of Aliases:

Number of Instances: Instance Selected:

#	Alias	Model Block	Model Param	Status	Instance 1
1	alias1	SYSTEM.OPCSERVER	EXTREF		OPC46.ref1
2	alias2	UTILITY.NUMERIC	PV		cm42.n1.PV

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
EXTRE F_STR UCT	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 INT16 value of 333 is converted to INT8 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

External OPC Server Support

OPC Data Type Conversions

- 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 Boolean, 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 PKS 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 PKS 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.

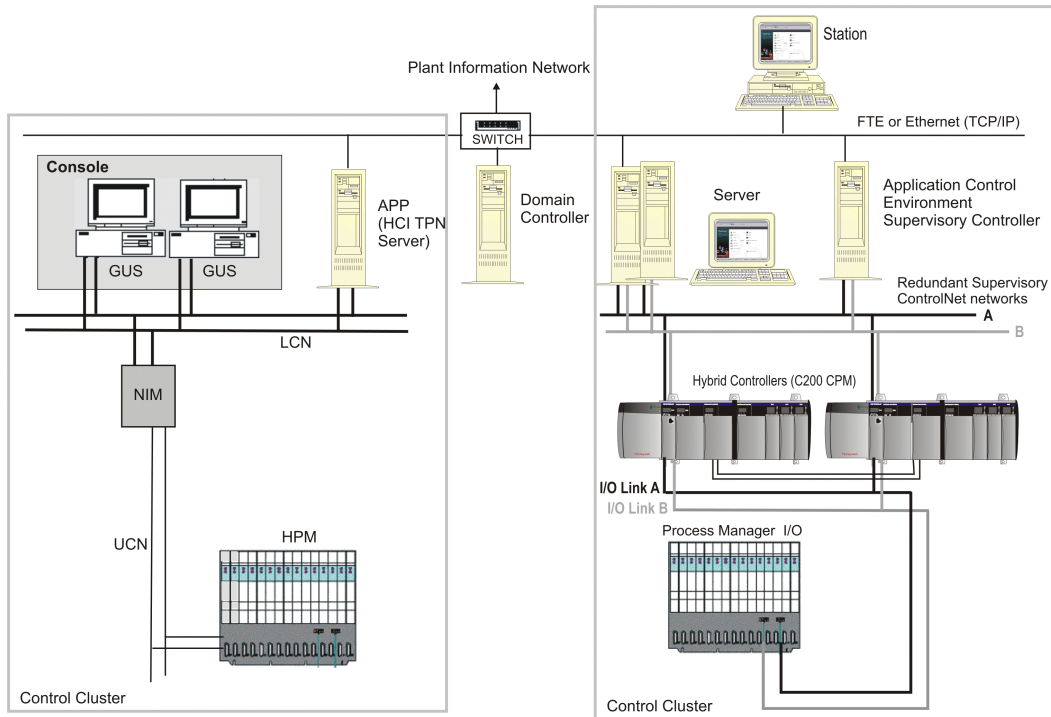


TIP

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.

External OPC Server Support

ACE interface to TPS system as OPC server



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.

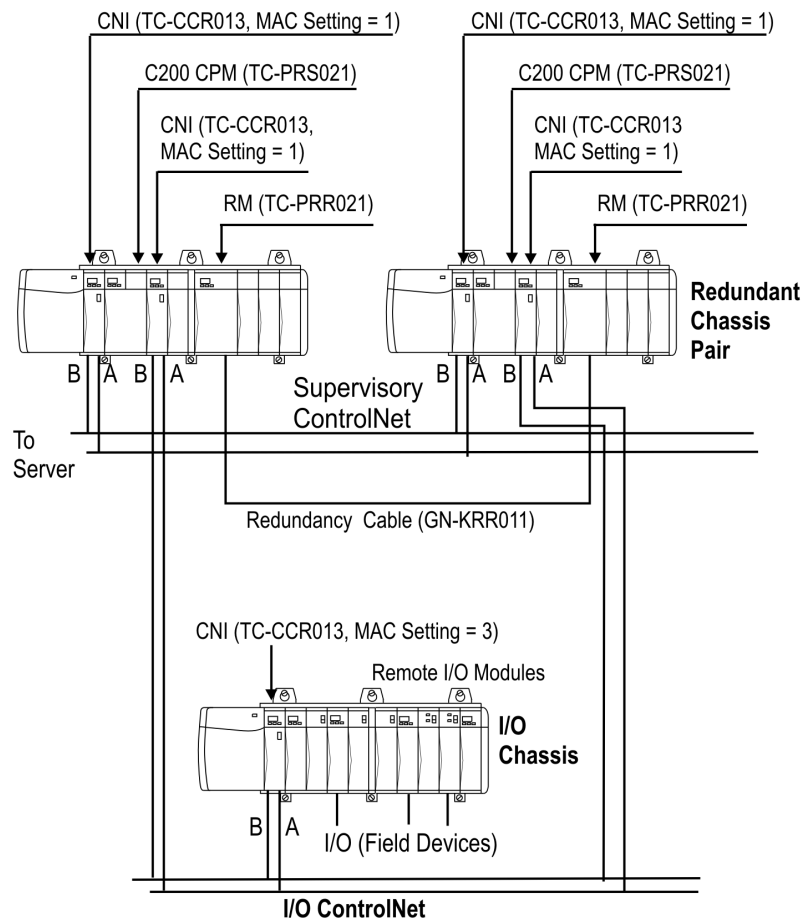


Figure 10 Typical RCP setup in 10-slot chassis.

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

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.

- Disqualified — This is the state of non-readiness and a Secondary in this state cannot assume the Primary state.
- Synchronizing — This is the state where the Secondary chassis is copying database information from the Primary. A Secondary in this state cannot assume the Primary state.
- Synchronized — This is the state where 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. A Secondary in this state can assume the Primary state upon a switchover.
- Standby (Not available at this time) — This is the state where some or none of the database in the Secondary is aligned with the database of the Primary and the Secondary does not track changes in the Primary database. This state is normally used for product or database migration. A Secondary in this state can assume the Primary state upon a switchover.

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.

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.

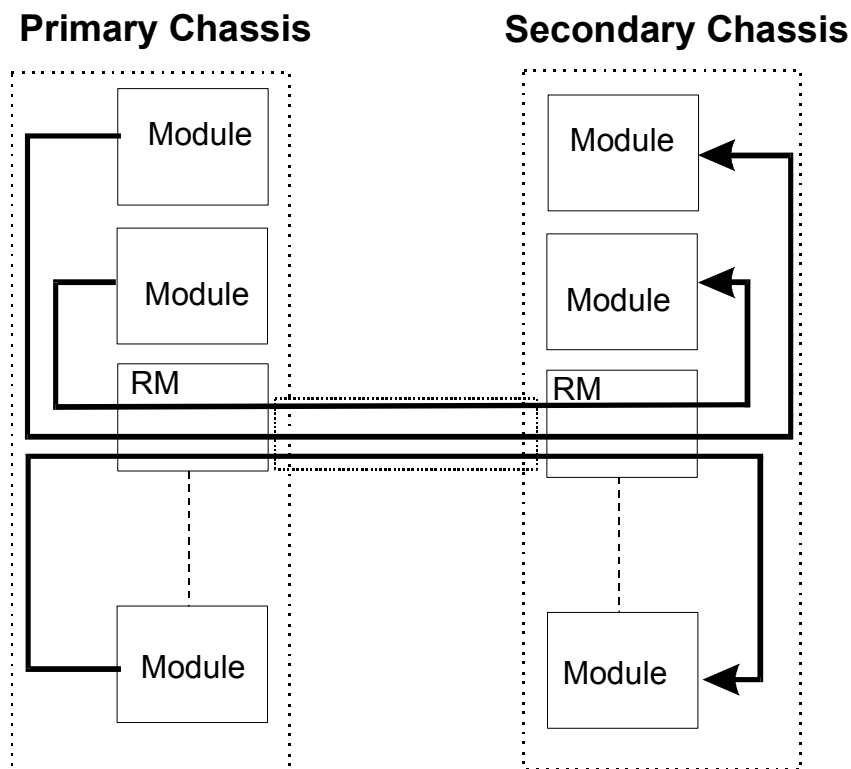


Figure 11 Typical redundancy communications path.

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.

Role of the C200 Control Processor

The redundancy compliant C200 Control Processor module (CPM) features a physical memory based tracking function to support initial synchronization. When a Secondary chassis is synchronized, only the CPM in the Primary chassis actively executes the assigned control functions while the CPM in the Secondary chassis remains dormant until a switchover occurs. This is known as the Logical Shared Memory redundancy technique because the primary and secondary CPMs act as if they have a single shared memory, which the primary CPM acts upon. However, each CPM has its own memory and a tracker mechanism helps maintain the integrity of the memory at a cleanpoint of execution.

A cleanpoint is defined as a point in time at which a set of data entities reaches a state of consistency, as defined by the application.

The following figure shows the software subsystems that are directly affected by the C200 CPM design and they are briefly described below for general reference only.

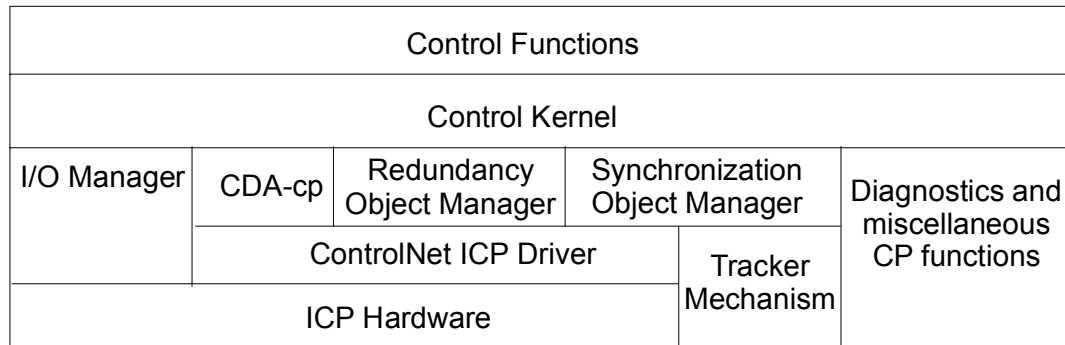


Figure 12 CPM software architecture

- **Control Functions** — The Control Functions subsystem consists of the various control algorithms in the user configured control strategy.
- **Control Kernel** — The Control Kernel (CK) is the execution thread responsible for modifications to the control database due to Function Block execution, invocation of Memory Management utilities, and parameter access.
- **I/O Manager** — The I/O Manager subsystem interfaces the Integrated Control Platform (ICP) Application Specific Integrated Circuit (ASIC) to serve as the sole channel for messages to and from I/O Modules. This subsystem executes in parallel on CPMs in both the Primary and Secondary chassis.
- **CDA-cp** — The Control Data Access (CDA) Controller Object Adapter (COA) subsystem provides the CPM with method invocation communications to and from other devices in the system. This subsystem executes in CPMs in both the Primary and Secondary chassis to publish any Secondary specific events. The CPM is responsible for the notification of all redundancy events.
- **Redundancy Object Manager** — The Redundancy Object Manager is responsible for communications with the RM and it participates in RM coordinated activities such as synchronization and switchover.
- **Synchronization Object Manager** — The Synchronization Object Manager is responsible for communications with the partner CPM over the RM-to-RM private path.

Controller Redundancy Functionality

Basic Redundancy Design Concepts

- **ControlNet ICP Driver** — The ICP Driver interface submits/receives ControlNet messages from the ICP backplane. It serves as the application interface to the ICP ASIC for all ControlNet objects implemented within the CPM. The ICP Driver is independently active on CPMs in both the Primary and Secondary chassis.
- **Tracker Mechanism** — The Tracker Mechanism is a CPM hardware engine to assist with the capture of writes into the Logical Shared Memory.
- **ICP Hardware** — The ICP ASIC serves as the CPM's hardware interface to the ICP Backplane through which all communications are performed.

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 CRCYCLOVRN(I) and LSCYCLOVRN(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 PKS 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 built-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* 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 activated.

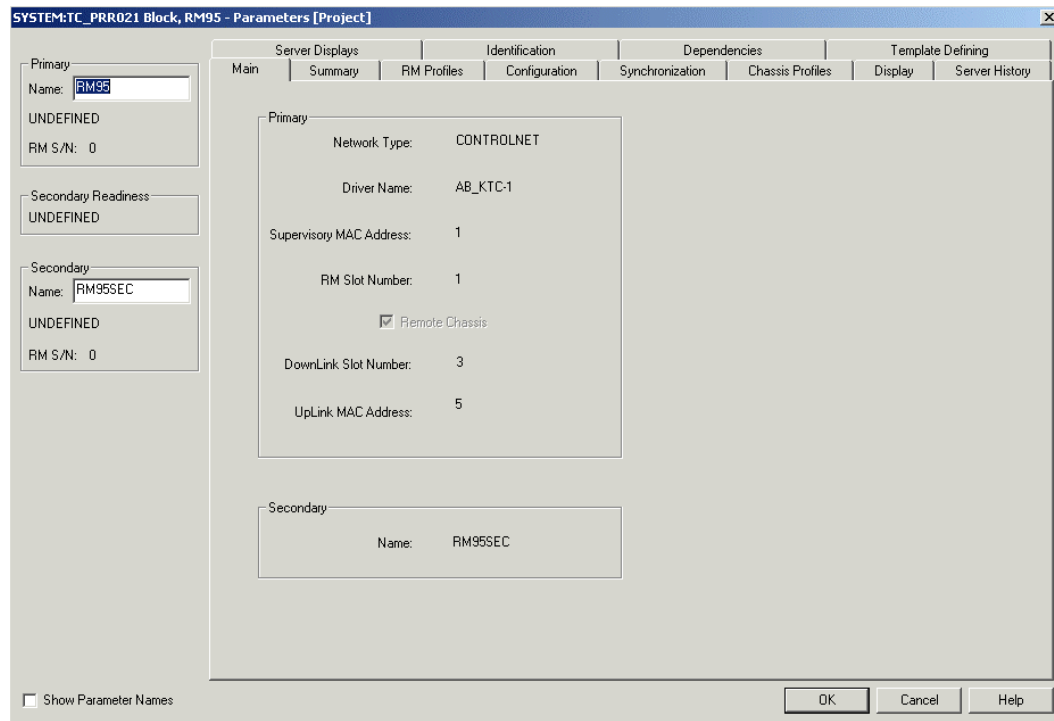


Figure 13 RM/RCP dialog box.

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.
 - **SYNCHING**: Secondary is in the process of being synchronized with the Primary and can not yet 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.

Server Displays		Identification		Dependencies		Template Defining	
Main	Summary	RM Profiles	Configuration	Synchronization	Chassis Profiles	Display	Server History
Primary Network Type: CONTROLNET Driver Name: AB_KTC-1 Supervisory MAC Address: 1 RM Slot Number: 1 ☒ Remote Chassis DownLink Slot Number: 3 UpLink MAC Address: 5 Secondary Name: RM95SEC							

Controller Redundancy Functionality

Implications for Control Builder Functions

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.

Server History	Server Displays	Identification	Dependencies	Template Defining		
Main	Summary	RM Profiles	Configuration	Synchronization	Chassis Profiles	Display

Auto-Synchronization

Option:
ALWAYS

State: ENABLED

Synchronize Secondary

Disqualify Secondary

Swap Control

Swap to Standby

Become Primary

Chassis Synchronization States

Slot	% Complete	Module	Compatibility
0	100	CNB	FULLY
1	100	CPM	FULLY
2	0	---	UNDEFINED
3	100	CNB	FULLY
4	0	---	UNDEFINED
5	100	RM	FULLY
6	0	---	UNDEFINED
7	0	---	UNDEFINED
8	0	---	UNDEFINED
9	0	---	UNDEFINED
10	0	---	UNDEFINED
11	0	---	UNDEFINED
12	0	---	UNDEFINED
13	0	---	UNDEFINED
14	0	---	UNDEFINED
15	0	---	UNDEFINED
16	0	---	UNDEFINED

Enter Standby

Initiate Switchover

- 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.

Server History		Server Displays		Identification		Dependencies		Template Defining	
Main	Summary	RM Profiles		Configuration	Synchronization	Chassis Profiles		Display	
IDENTIFICATION		Primary		Secondary					
Platform Ver.Rev:		1.1		0.0					
Boot Ver.Rev:		2.14		2.14					
Application Ver.Rev:		2.16		2.16					
Product Type-Code:		112 1		112 1					
Vendor Code:		14		3					
Serial Number:		4		2537820313					
Chassis ID:		CHASSIS_A		CHASSIS_B					
Description:									
Keyword:									
SUMMARY STATUS									
General State:		OK		OK					
Auto-Sync Option:		ALWAYS		ALWAYS					
Auto-Sync State:		ENABLED		ENABLED					
Program Cmd Recognition:		ENABLED		ENABLED					
Error Code:									
Fault Bits:		☐ Minor Recoverable ☐ Minor Unrecoverable ☐ Major Recoverable ☐ Major Unrecoverable		☐ Minor Recoverable ☐ Minor Unrecoverable ☐ Major Recoverable ☐ Major Unrecoverable					

- 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 PKS 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.

Server History		Server Displays		Identification		Dependencies		Template Defining	
Main	Summary	RM Profiles	Configuration	Synchronization	Chassis Profiles	Display			
Chassis Pair Auto-Synchronization Option: ALWAYS Program Command: ENABLED Chassis ID RM Serial Number: 4 2537820313 Chassis ID: CHASSIS_A CHASSIS_B Toggle Chassis ID									

Controller Redundancy Functionality

Implications for Control Builder Functions

- 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.

Maintenance Commands

Manually
Initiate
Synchronization
operations

Synchronize Secondary

Auto-Synchronization

Option:
ALWAYS

State: ENABLED

Maintenance Commands

Abort Synchronization
at next Sync Point,
with Secondary holding
latest Sync data

Enter Standby

Discontinue
Synchronization
Operations

Disqualify Secondary

Chassis Synchronization States

Slot	% Complete	Module	Compatibility
0	100	CNB	FULLY
1	100	CPM	FULLY
2	0	---	UNDEFINED
3	100	CNB	FULLY
4	0	---	UNDEFINED
5	100	RM	FULLY
6	0	---	UNDEFINED
7	0	---	UNDEFINED
8	0	---	UNDEFINED
9	0	---	UNDEFINED
10	0	---	UNDEFINED
11	0	---	UNDEFINED
12	0	---	UNDEFINED
13	0	---	UNDEFINED
14	0	---	UNDEFINED
15	0	---	UNDEFINED
16	0	---	UNDEFINED

Recent Synchronization Attempts

Order	Result	Cause
N...	SUCCE...	NOABORT
N-1	ABORT	EDIT_IN_PROG
N-2	ABORT	MOD_INSERT
N-3	NOATT...	NOABORT

Controller Redundancy Functionality

Implications for Control Builder Functions

- 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.

Controller Redundancy Functionality
Implications for Control Builder Functions

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.
MODPAIRINCOMP	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.

Controller Redundancy Functionality
Implications for Control Builder Functions

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 Disqualified 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.

Server History				Server Displays				Identification				Dependencies				Template Defining			
Main				Summary				RM Profiles				Configuration				Synchronization			
																Chassis Profiles			
Primary								Secondary											
Module	Gen State	Sec Readiness	C	Slot	Module	Gen State	Sec Readiness	C	Slot	Module	Gen State	Sec Readiness	C	Slot	Module	Gen State	Sec Readiness	C	Slot
CNB	OK	SYNCHRON...	FI	0	CNB	OK	SYNCHRON...	FI	0	CNB	OK	SYNCHRON...	FI	0	CNB	OK	SYNCHRON...	FI	0
CPM	UNDEF...	SYNCHRON...	FI	1	CPM	UNDEF...	SYNCHRON...	FI	1	CPM	UNDEF...	SYNCHRON...	FI	1	CPM	UNDEF...	SYNCHRON...	FI	1
---	UNDEF...	UNDEFINED	U	2	---	UNDEF...	UNDEFINED	U	2	---	UNDEF...	UNDEFINED	U	2	---	UNDEF...	UNDEFINED	U	2
CNB	OK	SYNCHRON...	FI	3	CNB	OK	SYNCHRON...	FI	3	CNB	OK	SYNCHRON...	FI	3	CNB	OK	SYNCHRON...	FI	3
---	UNDEF...	UNDEFINED	U	4	---	UNDEF...	UNDEFINED	U	4	---	UNDEF...	UNDEFINED	U	4	---	UNDEF...	UNDEFINED	U	4
RM	OK	SYNCHRON...	FI	5	RM	OK	SYNCHRON...	FI	5	RM	OK	SYNCHRON...	FI	5	RM	OK	SYNCHRON...	FI	5
---	UNDEF...	UNDEFINED	U	6	---	UNDEF...	UNDEFINED	U	6	---	UNDEF...	UNDEFINED	U	6	---	UNDEF...	UNDEFINED	U	6
---	UNDEF...	UNDEFINED	U	7	---	UNDEF...	UNDEFINED	U	7	---	UNDEF...	UNDEFINED	U	7	---	UNDEF...	UNDEFINED	U	7
---	UNDEF...	UNDEFINED	U	8	---	UNDEF...	UNDEFINED	U	8	---	UNDEF...	UNDEFINED	U	8	---	UNDEF...	UNDEFINED	U	8
---	UNDEF...	UNDEFINED	U	9	---	UNDEF...	UNDEFINED	U	9	---	UNDEF...	UNDEFINED	U	9	---	UNDEF...	UNDEFINED	U	9
---	UNDEF...	UNDEFINED	U	10	---	UNDEF...	UNDEFINED	U	10	---	UNDEF...	UNDEFINED	U	10	---	UNDEF...	UNDEFINED	U	10
---	UNDEF...	UNDEFINED	U	11	---	UNDEF...	UNDEFINED	U	11	---	UNDEF...	UNDEFINED	U	11	---	UNDEF...	UNDEFINED	U	11
---	UNDEF...	UNDEFINED	U	12	---	UNDEF...	UNDEFINED	U	12	---	UNDEF...	UNDEFINED	U	12	---	UNDEF...	UNDEFINED	U	12
---	UNDEF...	UNDEFINED	U	13	---	UNDEF...	UNDEFINED	U	13	---	UNDEF...	UNDEFINED	U	13	---	UNDEF...	UNDEFINED	U	13
---	UNDEF...	UNDEFINED	U	14	---	UNDEF...	UNDEFINED	U	14	---	UNDEF...	UNDEFINED	U	14	---	UNDEF...	UNDEFINED	U	14
---	UNDEF...	UNDEFINED	U	15	---	UNDEF...	UNDEFINED	U	15	---	UNDEF...	UNDEFINED	U	15	---	UNDEF...	UNDEFINED	U	15
---	UNDEF...	UNDEFINED	U	16	---	UNDEF...	UNDEFINED	U	16	---	UNDEF...	UNDEFINED	U	16	---	UNDEF...	UNDEFINED	U	16

- 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

Controller Redundancy Functionality

Implications for Control Builder Functions

- - - 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.
 - SYNCHING: Secondary is in the process of being synchronized with the Primary and can not yet 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.

Server Displays		Identification		Dependencies		Template Defining	
Main	Summary	RM Profiles	Configuration	Synchronization	Chassis Profiles	Display	Server History
Chassis ID RM Serial Number: 0 0 Chassis ID: UNDEFINED UNDEFINED Toggle Chassis ID Refresh Intervals (in ms) Primary RM: 0 Secondary RM: 0 Current CPU Free: Primary RM: 0 Secondary RM: 0							

Controller Redundancy Functionality

Implications for Control Builder Functions

Sever History tab

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

Access Levels

Control Level: 200 Control Area:

History Configuration

Number of History Parameters: 10

	Parameter	Description	FAST	STD	EXT	Gating Parameter	Gate State
1			☐	☐	☐		
2			☐	☐	☐		
3			☐	☐	☐		
4			☐	☐	☐		
5			☐	☐	☐		
6			☐	☐	☐		
7			☐	☐	☐		
8			☐	☐	☐		
9			☐	☐	☐		
10			☐	☐	☐		

Create New or Edit Existing Server Scripts

Control Level - A system wide configuration parameter for Experion PKS 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 PKS 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 historical 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.

Controller Redundancy Functionality

Implications for Control Builder Functions

Sever Displays tab

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

Main	Summary	RM Profiles	Configuration	Synchronization	Chassis Profiles	Display
Server History	Server Displays	Identification	Dependencies	Template Defining		

Point Detail Display	sysDtlRMA.dsp	Associated Display	
Group Detail Display	sysGrpRMA.dsp		

Trends

Number of Trends

	Trend #	Pen	Trend Parameter	Description
1	0	NONE		
2	0	NONE		
3	0	NONE		
4	0	NONE		
5	0	NONE		

Groups

Number of Groups

	Group #	Pos #	Group Parameter	Description
1	0	1		
2	0	1		
3	0	1		
4	0	1		
5	0	1		

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 PKS 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

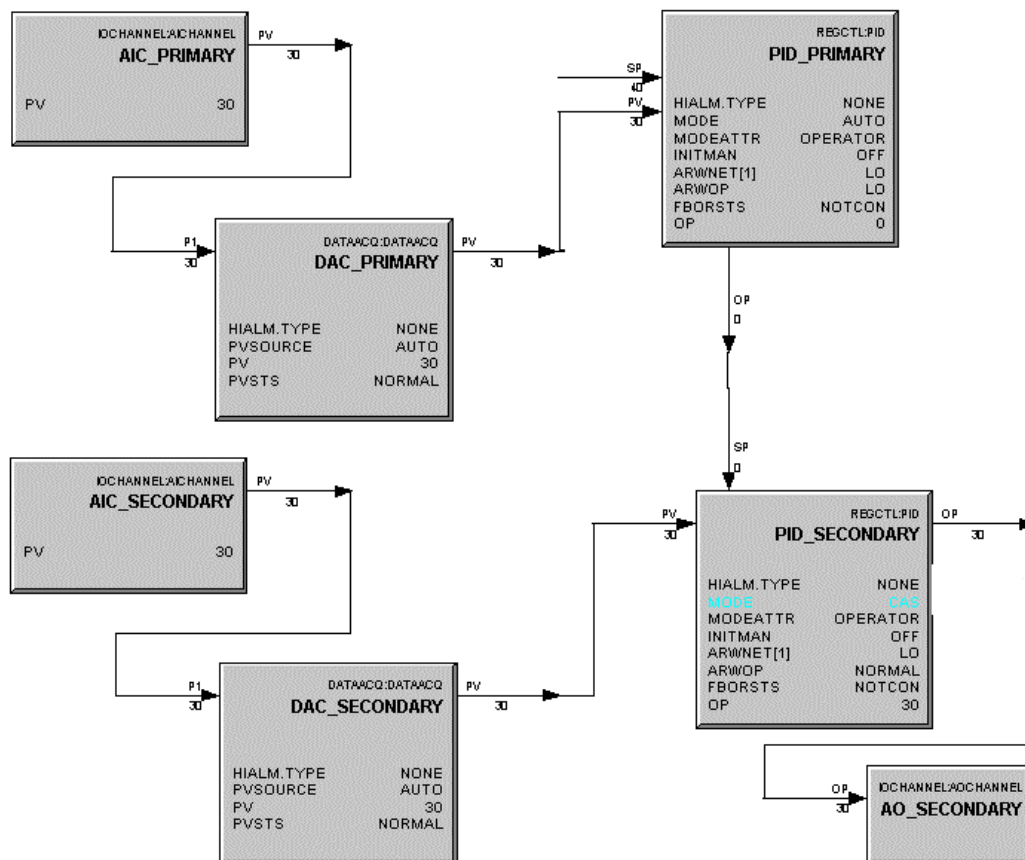
These common actions do **not** break the communications path or initiate a control mode shed:

- 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.
-

Control Mode Shed on Loss of I/O Functionality

Basic Control Mode Shed Design Concepts

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.

Control Mode Shed on Loss of I/O Functionality
Implications for Operation

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

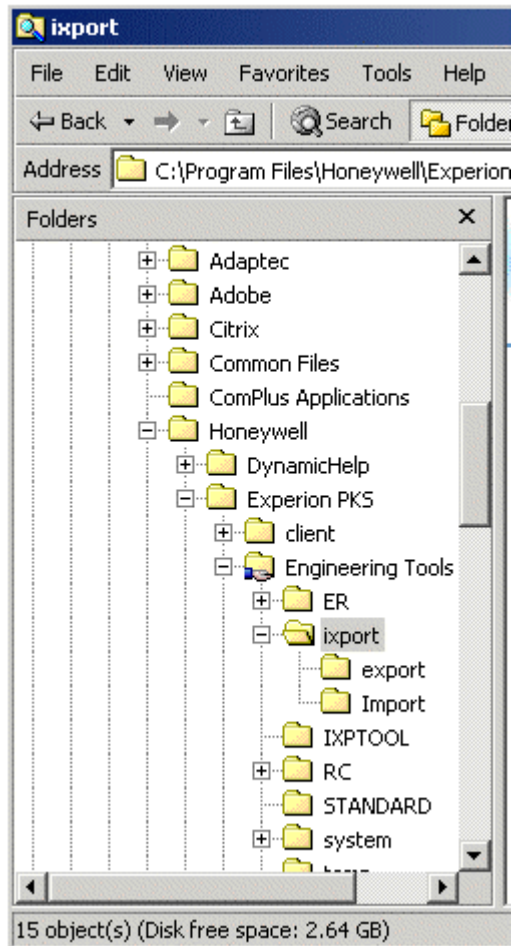
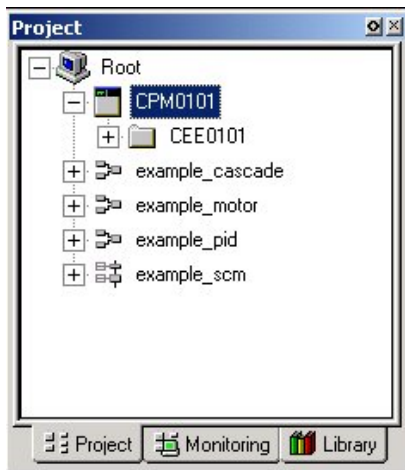
It may help to think of the Export function as a dynamic copy operation for the ERDB being accessed through Control Builder. This function lets you export a portion of the ERDB or the whole ERDB as viewed through the Project Tree in Control Builder. The exported or copied portion of the ERDB is automatically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

Control Builder Export and Import Functionality

Basic Export/Import Design Concepts

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.

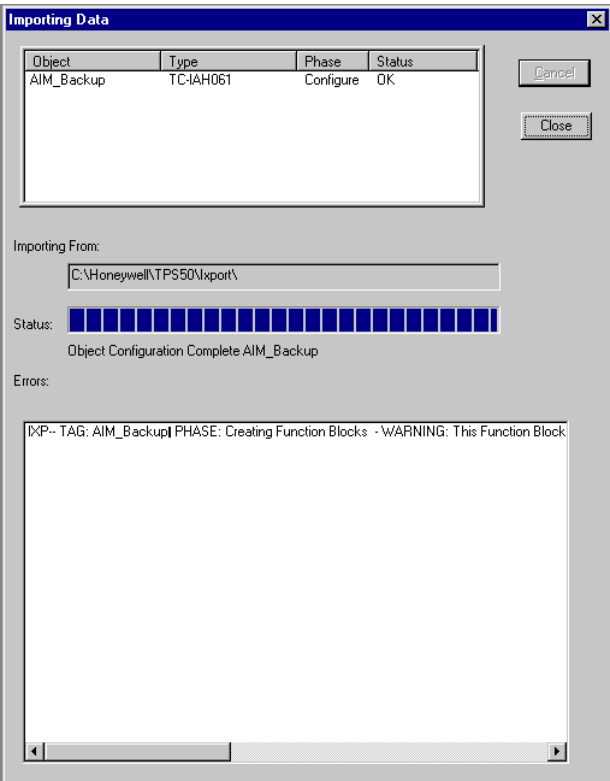
The exported portions or all of the ERDB are stored in the Ixport folder at directory location *c:\Program Files\honeywell\Experion PKS\Engineering Tools* by default.



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.

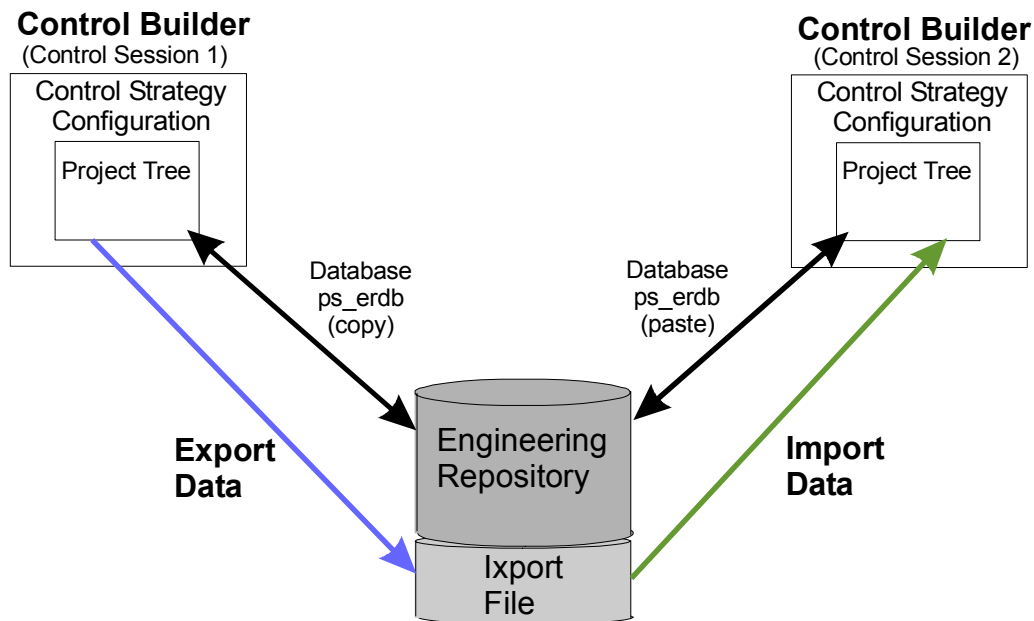
Import functionality

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.

Control Builder Export and Import Functionality
Some Operation Considerations

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

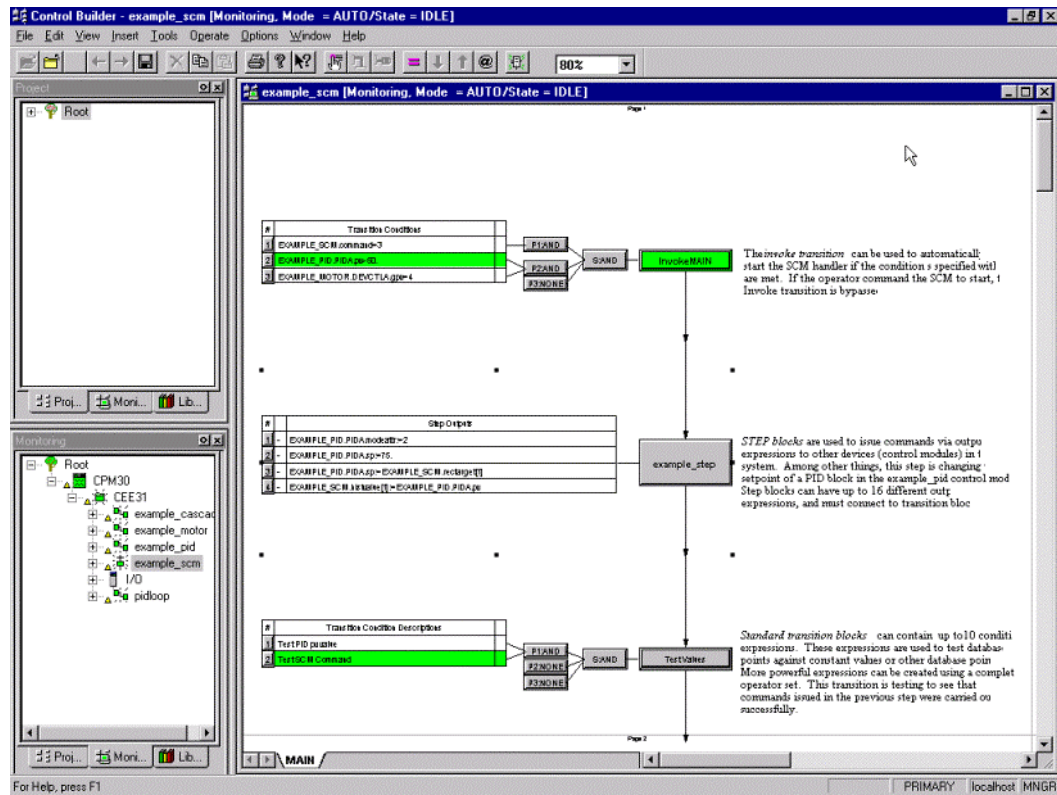
The ActiveX Controls needed to support the display and interaction of a Control Builder SCM or CM chart in the associated Detail Display in Station are loaded as part of the software. The software also includes updated SCM and CM Detail displays. The SCM display includes a Tab labeled "Chart". The redesigned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

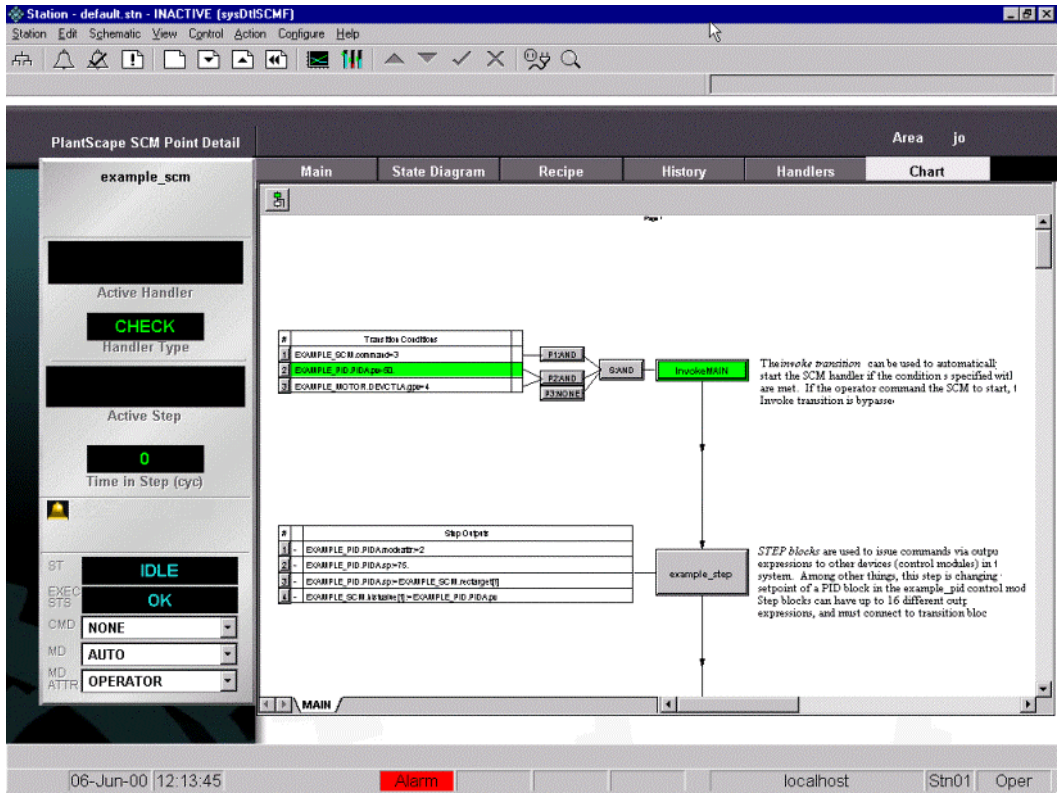
SCM and CM Chart Visualization Functionality

Basic Chart Visualization Design Concepts



View of example_scm chart in Control Builder Monitoring Tab.

SCM and CM Chart Visualization Functionality
Basic Chart Visualization Design Concepts

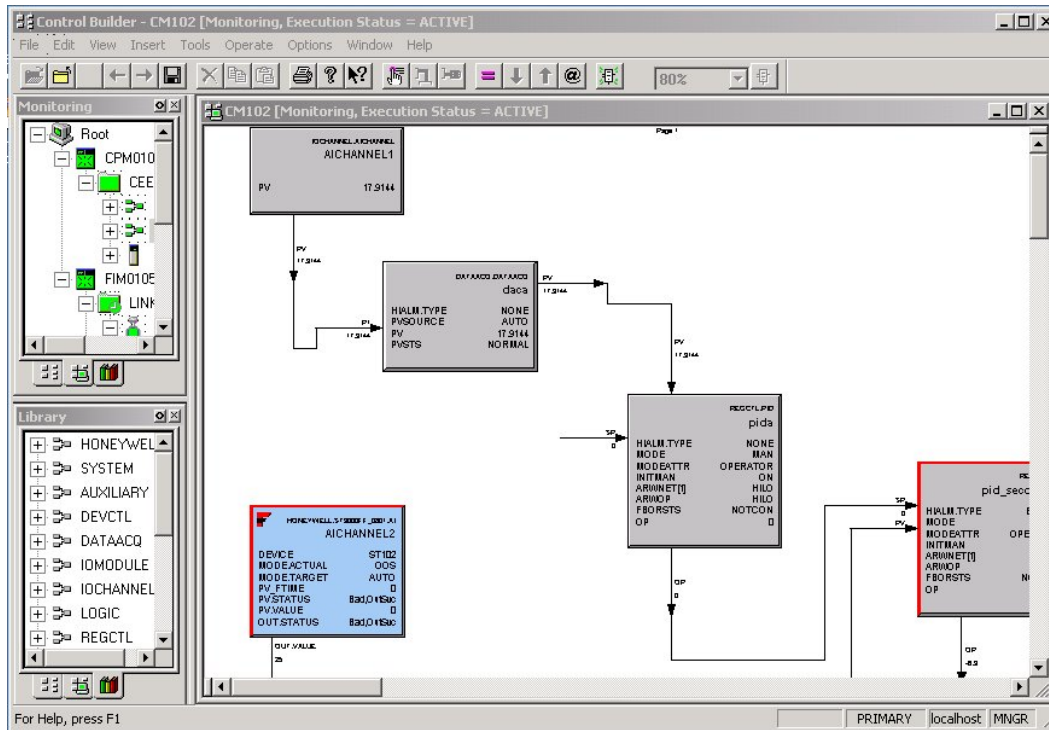


View of example_scm chart in SCM Detail display Chart Tab in Station.

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.

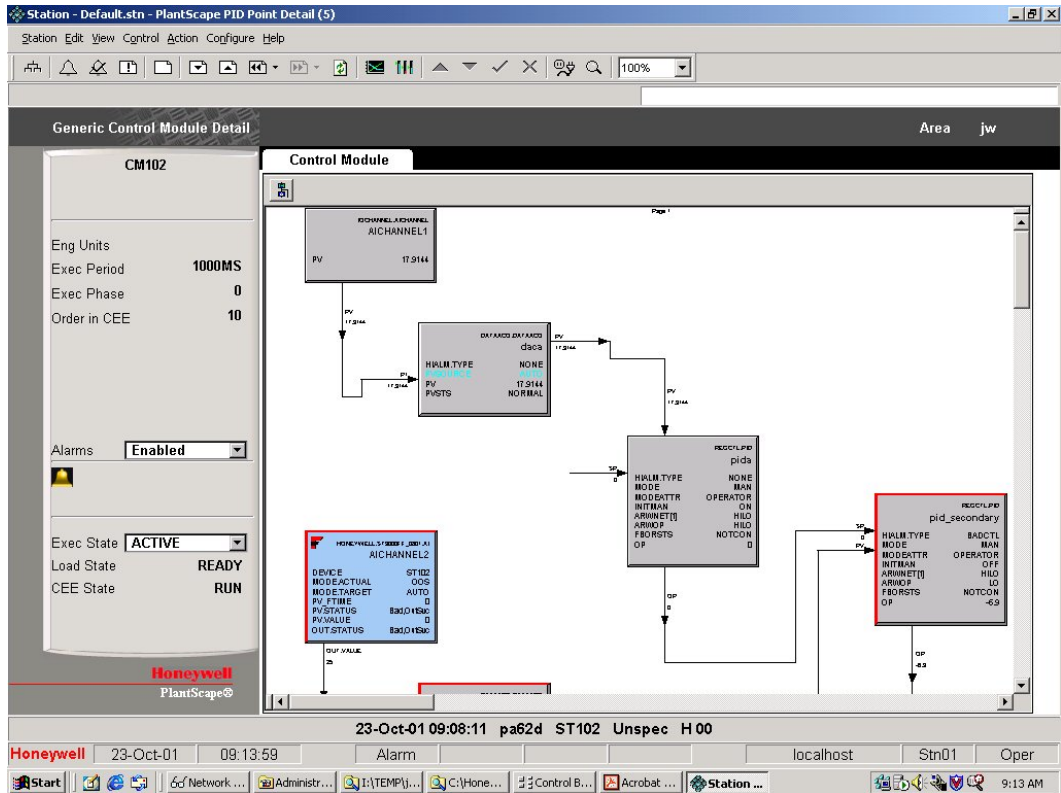
SCM and CM Chart Visualization Functionality

Basic Chart Visualization Design Concepts



View of CM102 chart in Control Builder Monitoring Tab.

SCM and CM Chart Visualization Functionality
Basic Chart Visualization Design Concepts



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.

Process Manager Input/Output Functionality

I/O Link Interface

Seamless integration

The I/O Link 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 a Experion PKS system. It lets user easily configure IOPs as part of a control strategy through the system's Control Builder application.

I/O Functions

The function of the IOPs remains the same. In conjunction with Field Termination Assemblies (FTAs), the IOPs 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 from 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.

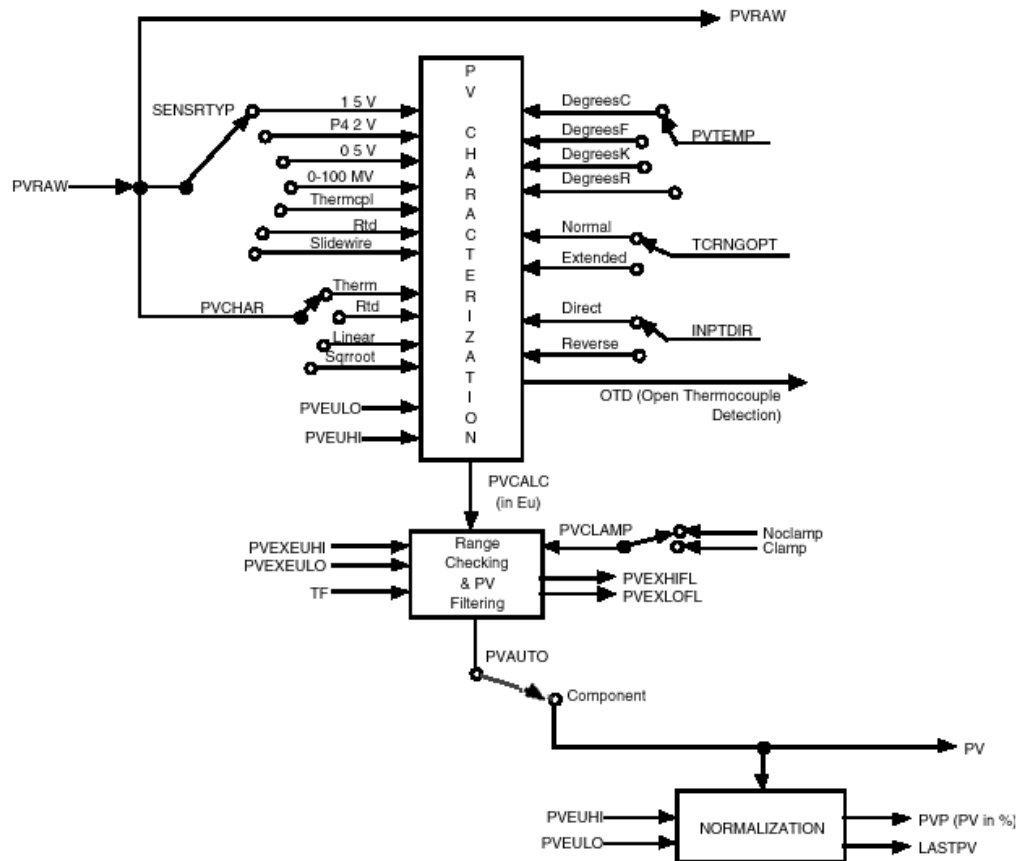
High and Low Level Analog Input Points

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

Process Manager Input/Output Functionality High and Low Level Analog Input Points



High-level points are located in the High Level Analog Input (HLAI and HLAI100) 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.

Process Manager Input/Output Functionality
High and Low Level Analog Input Points

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

Process Manager Input/Output Functionality
High and Low Level Analog Input Points

Sensor Type (SENSRTYP)	AI Processor Type	PVCHAR Options	PVRAW(1)	PV CALC	Bad PV Direction(2)
Legend: EU = Engineering Units HL = High Level Analog Input LL = Low Level Analog Input LLMUX = Low Level Analog Multiplexer Input PV CALC = 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: - 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: - For a thermocouple sensor type, PVRAW is in microvolts after reference junction compensation. If an open thermocouple is detected, PVRAW is set to NaN. - For an RTD sensor type, PVRAW is in milliohms after lead-wire compensation. If an open RTD is detected, PVRAW is set to NaN. - 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 (V_{in}/V_{src}). Where: V_{in} is the FTA voltage input for this data point (slot) V_{src} is the FTA voltage source at the slidewire source slot specified by parameter SLWSRCID. If V_{src} is zero (fails the under-range check), PVRAW is set to NaN. - For a 0 to 100 millivolt sensor type, PVRAW is the FTA voltage input for the slot. - If the diagnostics determine that the A/D converter has failed, PVRAW of the slot is set to NaN. - 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).					

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 PV signal (PVRW) whose units can be percent, ratio, millivolts, microvolts, or milliohms depending on the entry made for the SENSRTYP parameter. The PVRW signal is then converted to the engineering units. The engineering unit conversions that are performed in the HLA, LLA 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 PVRW 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:

$$PVCALC = \frac{(PVRW - PVRWLO)}{(PVRWHI - PVRWLO)} * (PVEUHI - PVEULO) + PVEULO$$

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

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

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

$$PVCALC = PVEUHI - \frac{(PVRW - PVRWLO)}{(PVRWHI - PVRWLO)} * (PVEUHI - PVEULO)$$

- 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 *Control Builder Parameter Reference* 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.

Process Manager Input/Output Functionality
High and Low Level Analog Input Points

RTD Type	Maximum Allowable Lead Resistance		Maximum Allowable Intrinsic Safety Barrier Resistance	
	<i>Entire Loop</i>	<i>Per Leg</i>	<i>Entire Loop</i>	<i>Per Leg</i>
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.)

PMIO:AICHANNEL Block, AICHANNEL_01 - Parameters [Project] [Read-Only]

Main Configuration Smart Transmitter

Transmitter Tag Name:

Digitally Enhanced Mode:

Range Value:

Upper Range Value:

Lower Range Value:

Range Limit:

Upper Range Limit:

Lower Range Limit:

Damping:

☒ Internal Cold Junction Comp

Secondary Variable:

Command:

Scratch Pad:

Database Discrepancy:

STI Engineering Units:

Serial Number:

Software Revision:

Current State:

Status:

PV Number:

☐ Show Parameter Names

OK Cancel Help

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 <i>User Manual</i> .
ResetErr	Refer to the Transmitter's <i>User Manual</i> .
Restart	Refer to the Transmitter's <i>User Manual</i> .
Shutdown	Refer to the Transmitter's <i>User Manual</i> .

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

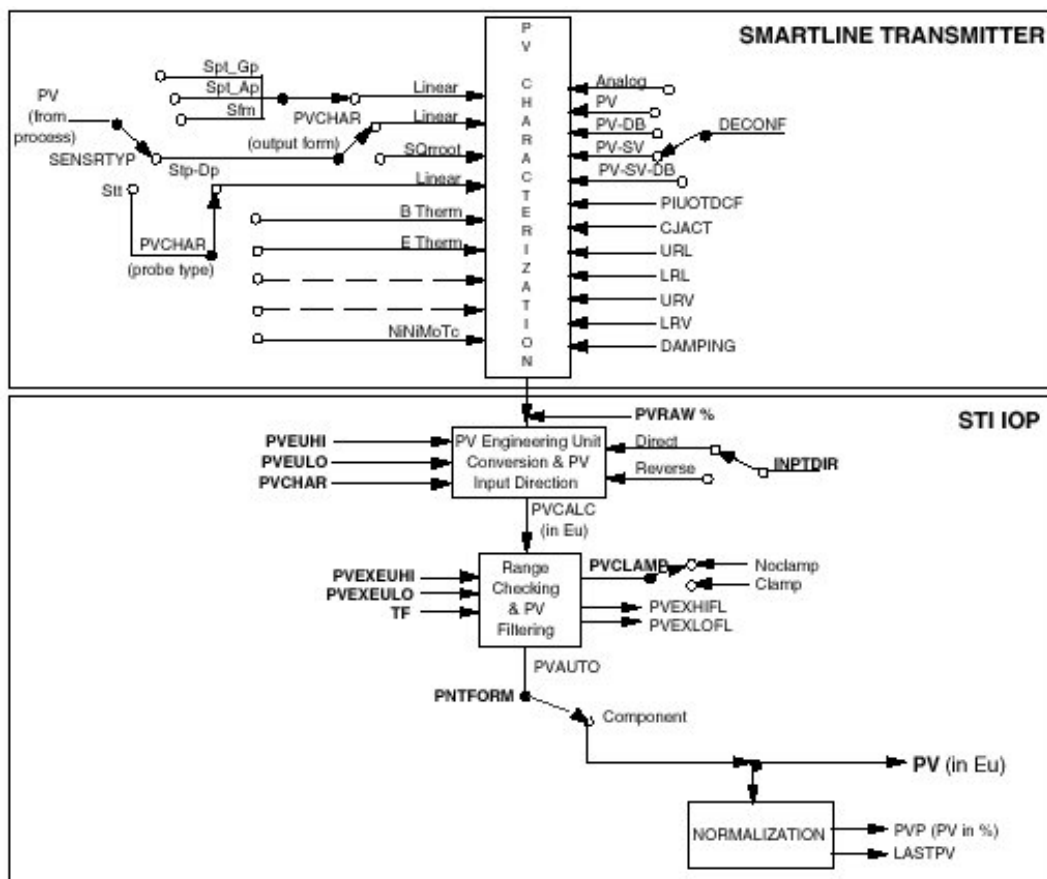
Process Manager Input/Output Functionality

Smart Transmitter Interface Point

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

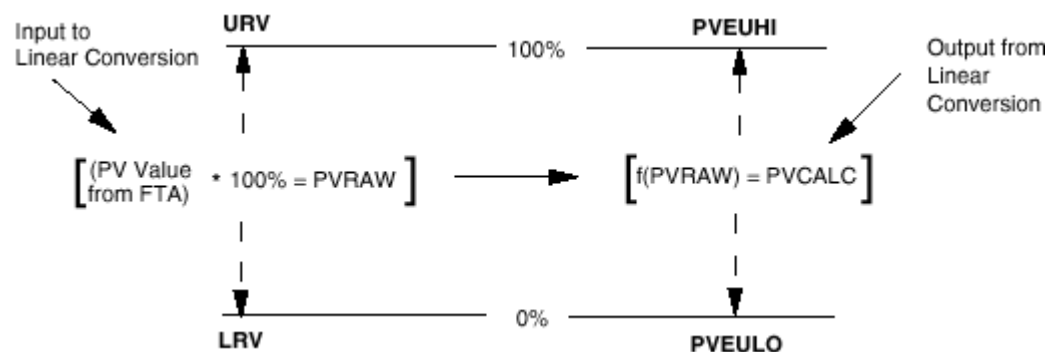
Process Manager Input/Output Functionality

Smart Transmitter Interface Point

Transmitter (Sensor) Type (SENSRTYP)	PVCHAR Options	PVRAW(1,3)	PV CALC	PV Detection(2)
Legend: EU = Engineering Units PV CALC = Calculated Process Variable PVCHAR = Process Variable Characterization PVRAW = Process Variable received from transmitter and multiplied by 100 by the STI IOP. Notes: 1. PVRAW is a percentage of the configured range for the sensor type. For Multivariable transmitters, PVRAW Engineering Units are different for each PV slot. 2. 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. 3. The normal operating range for PVRAW (0 percent equals PVRAWLO, 100 percent equals PVRAWHI) is configured by the user. 4. For the supported temperature ranges, refer to the definition of the PVCHAR parameter in the <i>Control Builder Parameter Reference</i>.				

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 PVEUHI and PVEULO 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

Process Manager Input/Output Functionality

Smart Transmitter Interface Point

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		
<i>Pressure (Spt)</i>	<i>Temperature (Stt)</i>	<i>Flow (Sfm)</i>
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

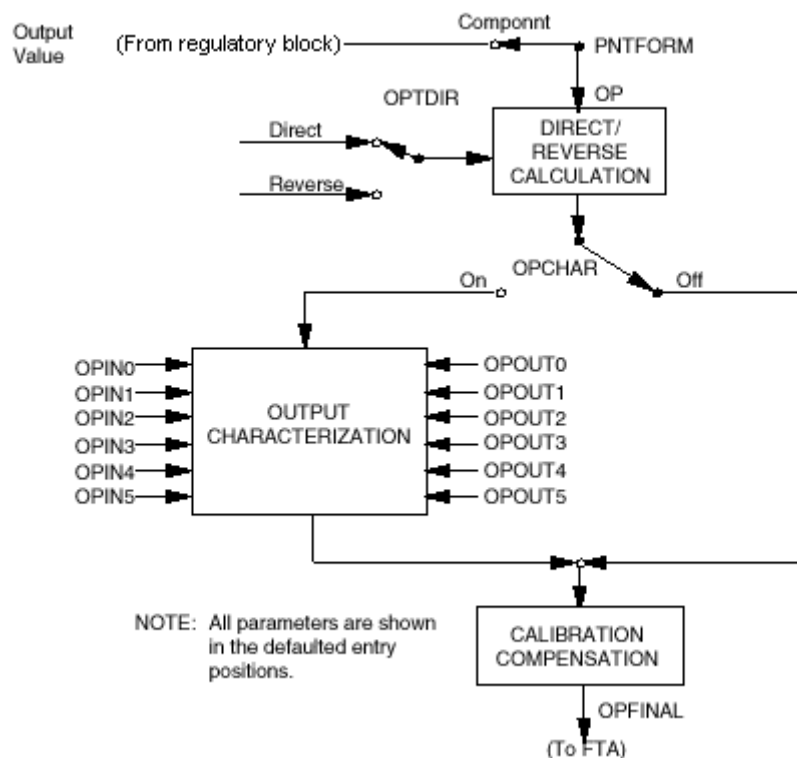
Analog Output Point

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.



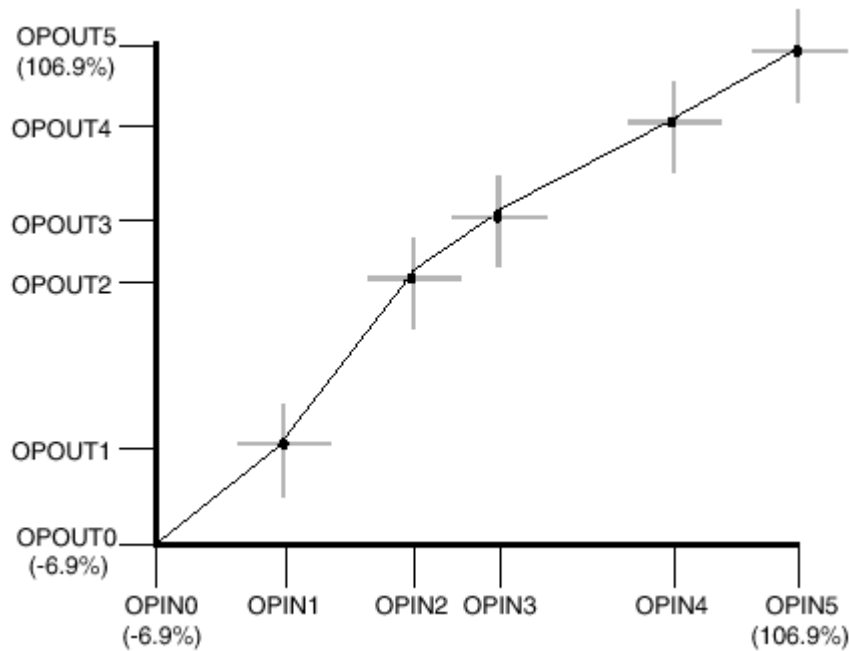
AO direct/reverse output

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

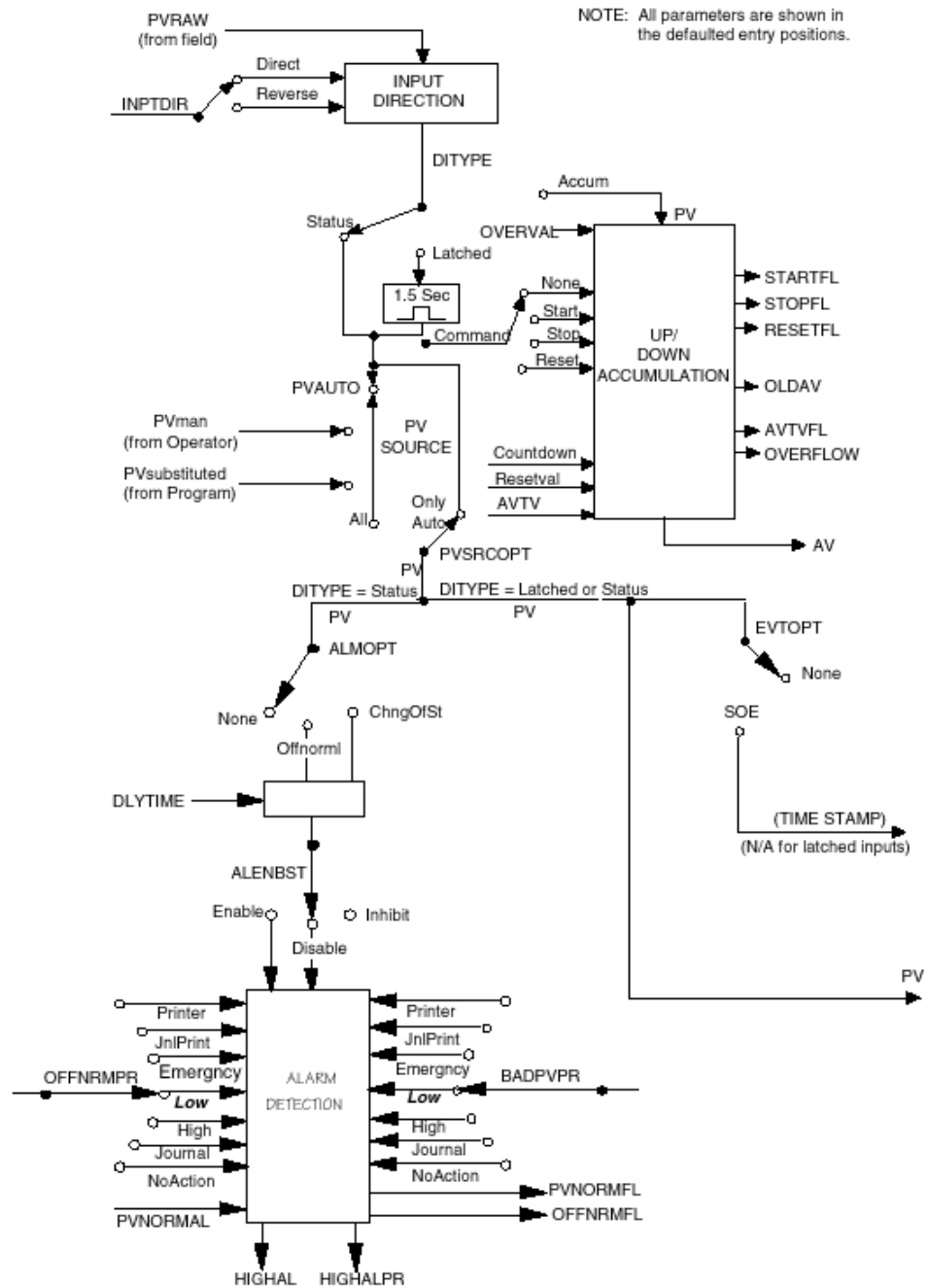
A digital input point converts a digital PVRAW signal received from the field to a PV that can be used by other data points in the control strategy. A functional diagram of the digital input point is shown in the following figure.

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.

Process Manager Input/Output Functionality Digital Input Point



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.)

PMIO:DICHANNEL Block, DICHANNEL_01 - Parameters [Project] [Read-Only]

Main Configuration

Digital Input Type: STATUS

Advanced Configuration

Point Form: COMPONENT

Input Direction: DIRECT

PV Source Option: ONLYAUTO

PV Source: AUTO

Alarm Configuration

Alarm Option: NONE

PV Normal State: OFF

Alarm Hold Delay (Seconds):

Events

Reporting Option: NONE

Contact Debounce (mSeconds):

PV Hold Delay (Seconds):

PV State Text:

☐ Show Parameter Names

OK Cancel Help

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.

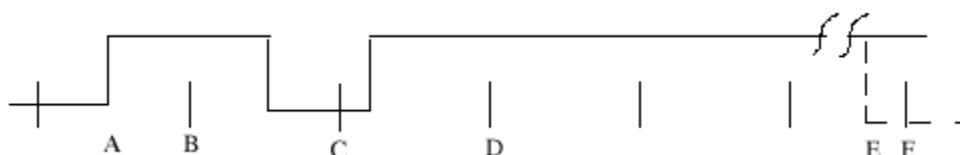
The following figures illustrate the debounce operation.



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 to 60 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.

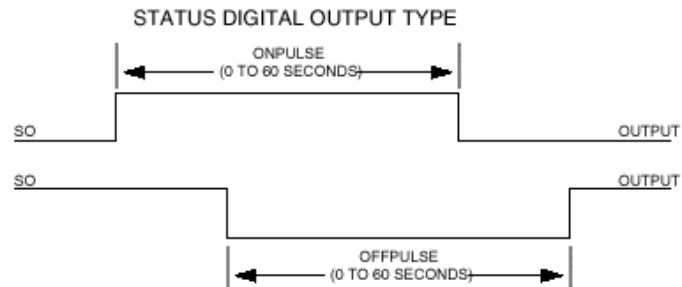
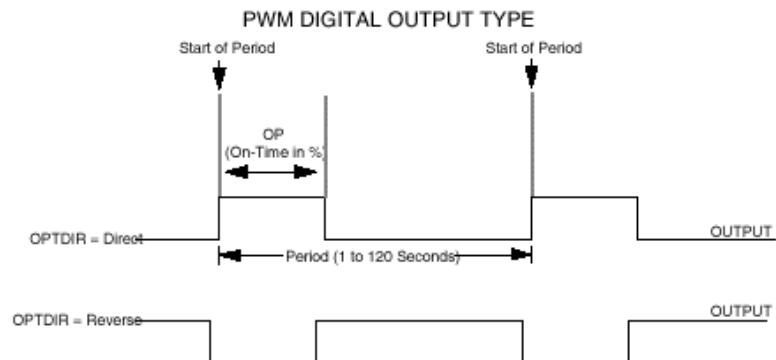
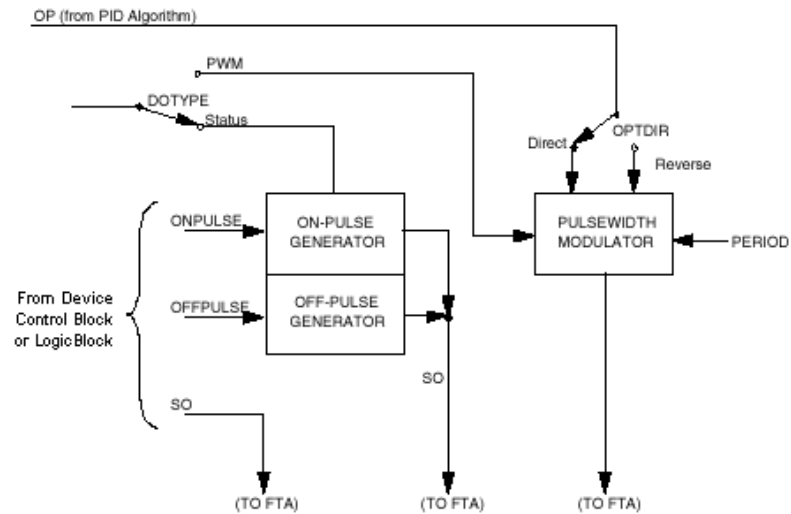
Digital Output Point

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.

Process Manager Input/Output Functionality

Digital Output Point



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:

$$\text{Pulse On-Time} = \frac{\text{OP\%} * \text{PERIOD}}{100}$$

For reverse action:

$$\text{Pulse On-Time} = \frac{100\% - \text{OP\%} * \text{PERIOD}}{100}$$

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 PKS 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 PKS Control Builder Components Reference*.

Component Categories and Types

Overview

Regulatory Control

Regulatory Control Blocks

Functional overview

Experion PKS 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.
Fanout	FANOUT Block	Sends one input to many outputs, whereby a different gain and bias may be assigned to each output.
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.

Regulatory Control

Regulatory Control Blocks

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 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 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.
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 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 outputs the selected value.

Common regulatory control functions



ATTENTION

The Regulatory Control blocks can interact with Sequential Control Modules (SCM) to provide interactive control action, please refer to the *SCM Interface and CM interaction* section 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.

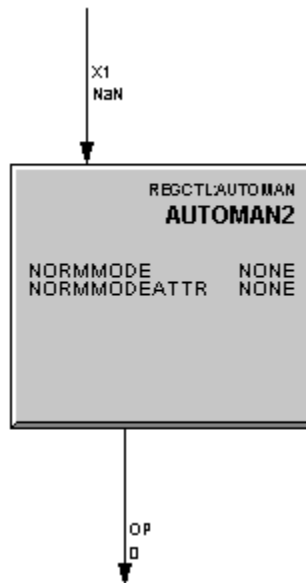
Major Function	Description
Input Processing	Provides these functions for all Regulatory Control function blocks to be used as needed. PV Processing – Regulatory control blocks that have PV input use this function to fetch the input value, status, and update the appropriate PV parameters. SP Processing – 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.
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.

Major Function	Description
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).

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 0 to 65535. 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, NONE, and NORMAL. 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, PROGRAM, and NORMAL. 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. 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

Configuration Tab	Description
	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. • 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 NOSLED.
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

Configuration Tab	Description
	(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 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 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

Configuration Tab	Description
	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, 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

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. 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 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

Configuration Tab	Description
	applicable for the given regulatory control function block: – NONE - No changes. – MAN - Set MODEREQ = MANUAL. – 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.

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

Figure 14 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.

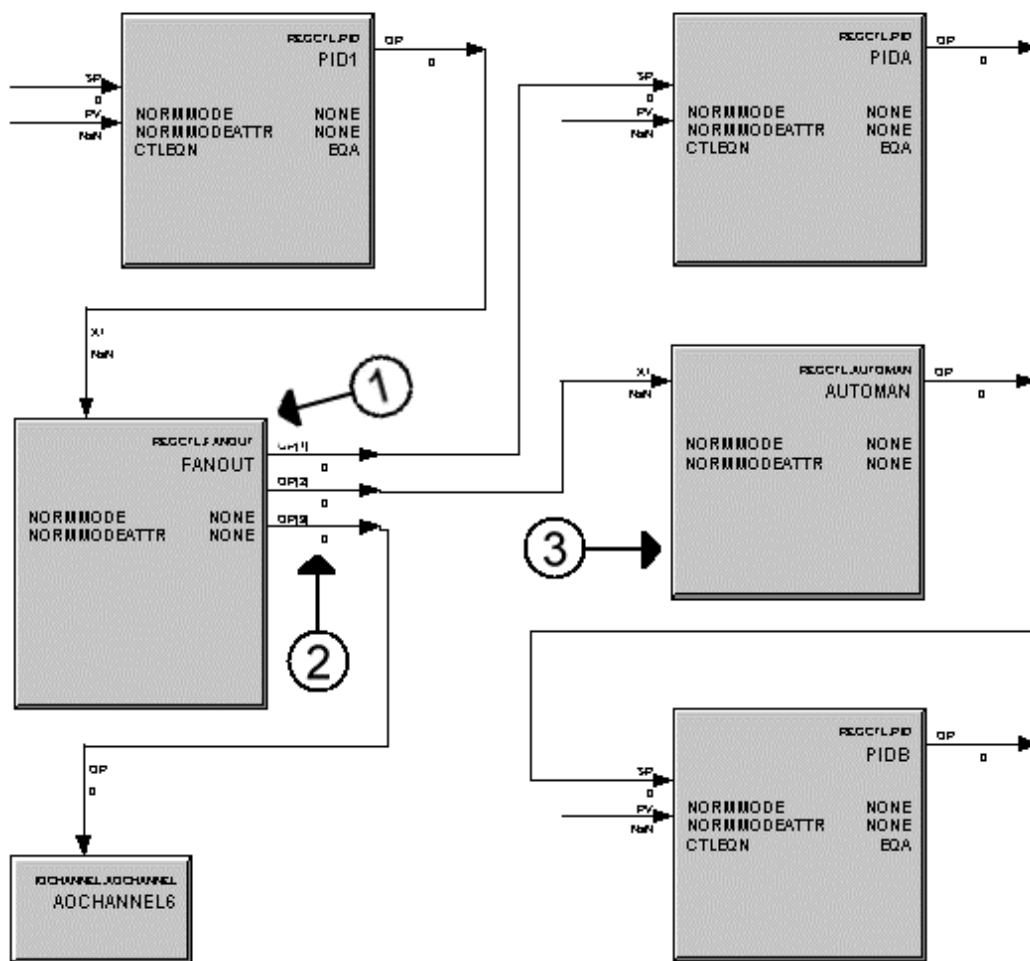


Figure 14 Example of CB configuration using AUTOMAN block.

The following table includes descriptions of the callouts in Figure 14.

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.

$$\text{OPBIAS.FLOAT} = \text{CV}_{\text{INIT}} - (\text{CV}_{\text{UNBIASED}} + \text{OPBIAS.FIX})$$

Where:

CV_{INIT}	=	initialization value received from the secondary
$\text{CV}_{\text{UNBIASED}}$	=	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 $\text{CV} + \text{OPBIAS.FIX}$ should be the same as CV_{INIT} 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 ramp rate 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, a 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.

Regulatory Control

AUTOMAN (Auto Manual) Block

$$\text{OPBIAS.FLOAT} = \text{OPBIAS.FLOAT} - (\text{OPBIAS.RATE} / \text{cycles\_per\_Min})$$

Where:

$$\text{cycles\_per\_min} = \text{number of times the function block executes per minute (calculated)}$$

– 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.



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:

$$\text{INITREQ}(X1) = \text{On}$$

$$\text{INITVAL}(X1) = \frac{\text{CV} - \text{OPBIAS.FIX}}{K}$$

– 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:

$$\text{feedback value for primary} = (\text{override feedback value from secondary} - \text{OPBIAS.FIX} - \text{OPBIAS.FLOAT}) / K$$

AUTOMAN parameters



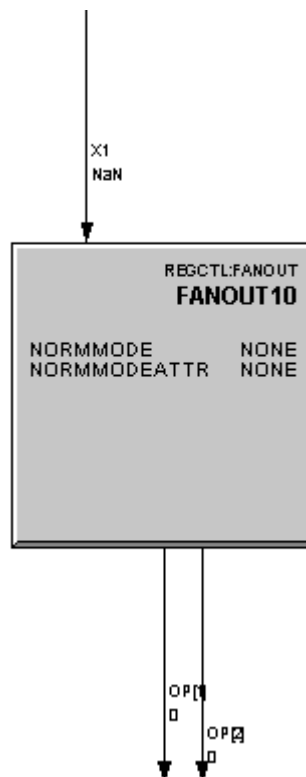
REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the AUTOMAN 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 0 to 65535. 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, NONE, and NORMAL. 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, PROGRAM, and NORMAL. 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 NOSLED. • 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 the Calculated Variable range (CVEUHI – CVEULO).

Configuration Tab	Description
	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

Configuration Tab	Description
	500 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 SCM Interface and CM Interaction section in this document for selection details. The default selection is ONESHOT. The selections are:

Configuration Tab	Description
	- 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). • Value (STARTVAL, STOPVAL, HOLDVAL) – Depending upon Option Type selection, lets you

Configuration Tab	Description
	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:

Regulatory Control

FANOUT Block

$$OP = (CV - CVEULO) / CVEUSPANBY100$$

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 Figure 14 in the AUTOMAN block section 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.

$$\text{OPBIAS.FLOAT} = \text{CV}_{\text{INIT}} - (\text{CV}_{\text{UNBIASED}} + \text{OPBIAS.FIX})$$

Where:

CV_{INIT} = initialization value received from the secondary

$\text{CV}_{\text{UNBIASED}}$ = 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 $\text{CV} + \text{OPBIAS.FIX}$ should be the same as CV_{INIT} 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 ramp rate 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.

$$\text{OPBIAS.FLOAT} = \text{OPBIAS.FLOAT} - (\text{OPBIAS.RATE} / \text{cycles\_per\_Min})$$

Where:

$$\text{cycles\_per\_min} = \text{number of times the function block executes per minute (calculated)}$$

- 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.



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:

$INITREQ(X1) = \text{On}$

$$INITVAL(X1) = \frac{CV - OPBIAS(last).FIX}{K(last)}$$

- 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:

$$\text{INTVAL (X1)} = \frac{\text{CV} - \text{OPBIAS (index).FIX}}{\text{K (index)}}$$

– 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:

1. 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.
2. Refer to Windup Processing below.

Windup processing

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.

FANOUT parameters



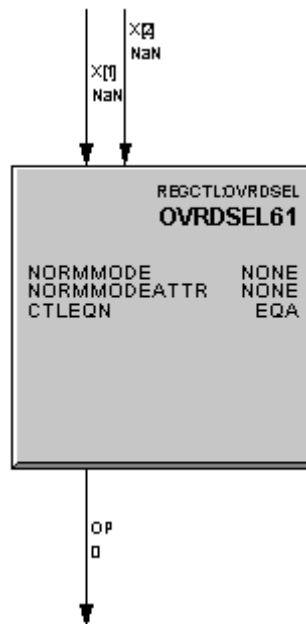
REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the FANOUT 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 0 to 65535. 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). • Normal Mode (NORMMODE) – Lets you specify the

Configuration Tab	Description
	MODE the block is to assume when the Control to Normal function is initiated through the Station display. Selections are MANual, AUTOmatic, CAScade, BackupCAScade, NONE, and NORMAL. 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, PROGRAM, and NORMAL. 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

Configuration Tab	Description
	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 NOSLED.
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 (ORBYPERM) – 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 the CEE subscription rate is 100 milliseconds and the TMOUTTIME is 5 seconds, the actual timeout time for

Configuration Tab	Description
	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 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 $481.05 + 50$

Configuration Tab	Description
	(CVEULO) equals 531.05. The default value is 106.9%. - Extended Low Limit (%) (OPEXL0LM) – 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. - 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

Configuration Tab	Description
	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 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.
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: - NONE - Alarm is neither reported nor annunciated. - JOURNAL - Alarm is logged but it does not appear on

Configuration Tab	Description
	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 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.

Configuration Tab	Description
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 (Not applicable for this block). – CAS - Set MODEREQ = CASCADE. – FIXEDOP - Set OPREQ = Configured Value. – HOLDPV - Set SPREQ = PV (Not applicable for this block).

Configuration Tab	Description
	– 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.

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.

Regulatory Control
OVRDSEL (Override Selector) Block

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

Figure 15 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.

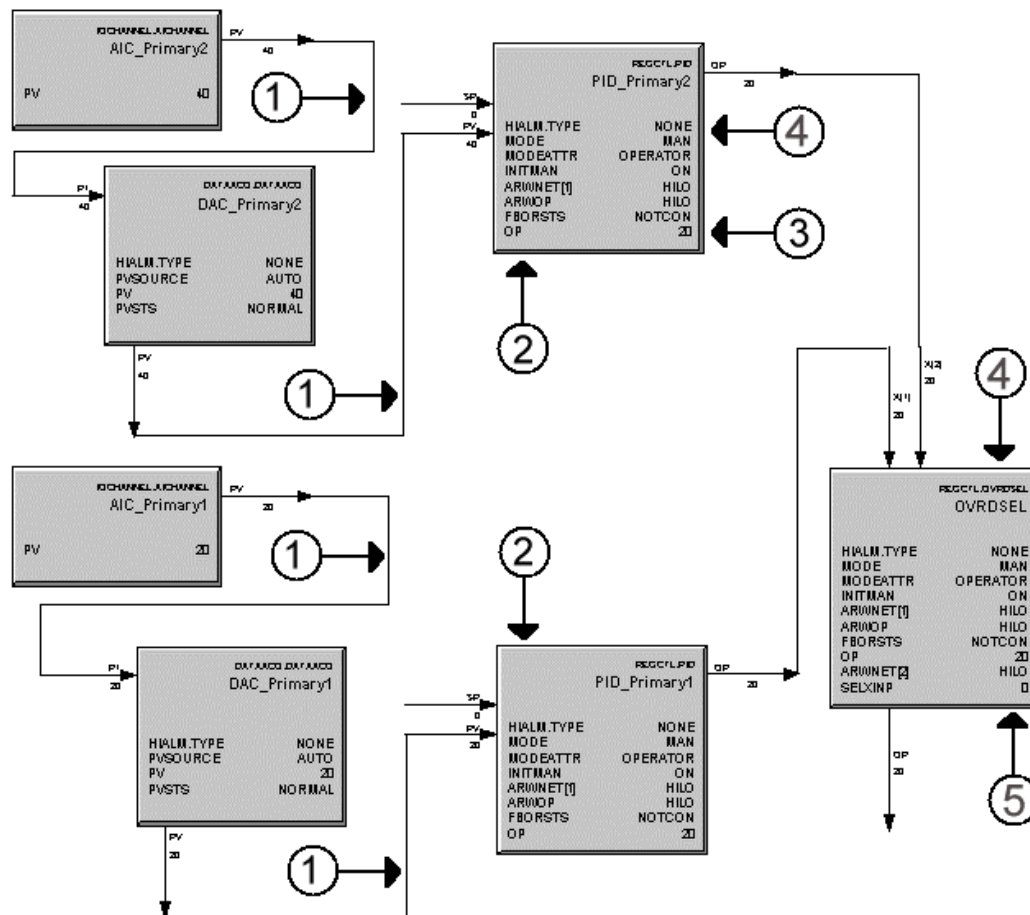


Figure 15 Example of CB configuration using OVRDSEL block.

Regulatory Control
OVRDSEL (Override Selector) Block

The following table includes descriptions of the callouts in Figure 15.

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. This 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 is 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.

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 brings 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. 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.

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.

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:

$$CV = \text{the highest input} + OPBIAS.FIX + OPBIAS.FLOAT$$

- Equation B - select the lowest of the non-bypassed inputs:

$$CV = \text{the lowest input} + OPBIAS.FIX + OPBIAS.FLOAT$$

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.

$$OPBIAS.FLOAT = OPBIAS.FLOAT - (OPBIAS.RATE / \text{cycles\_per\_Min})$$

Where:

$$\text{cycles\_per\_min} = \text{number of times the function block executes per minute (calculated)}$$

- 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.

Regulatory Control

OVRDSEL (Override Selector) Block

- 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:
 $CV = \text{override feedback value}$
 - If the offset flag is On, the PID applies an offset; it initializes its CV as follows:
 $CV = (\text{override feedback value}) + \text{Gain} * (\text{PVP} - \text{SPP})$ for direct control action.
 $CV = (\text{override feedback value}) - \text{Gain} * (\text{PVP} - \text{SPP})$ for reverse control action.
 - Additionally, the $\text{Gain} * (\text{PVP} - \text{SPP})$ term is set to 0.0; If:
 - à $\text{Gain} * (\text{PVP} - \text{SPP}) > 0.0$ and the downstream OVRDSEL block is a High selector. Or,
 - è $\text{Gain} * (\text{PVP} - \text{SPP}) < 0.0$ 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.

OVRDSEL parameters



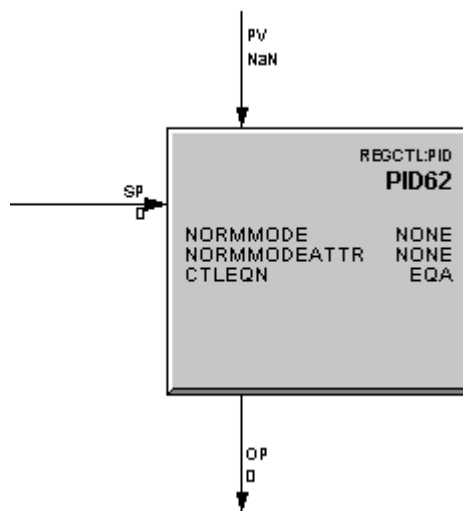
REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* 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. Figure 16 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.

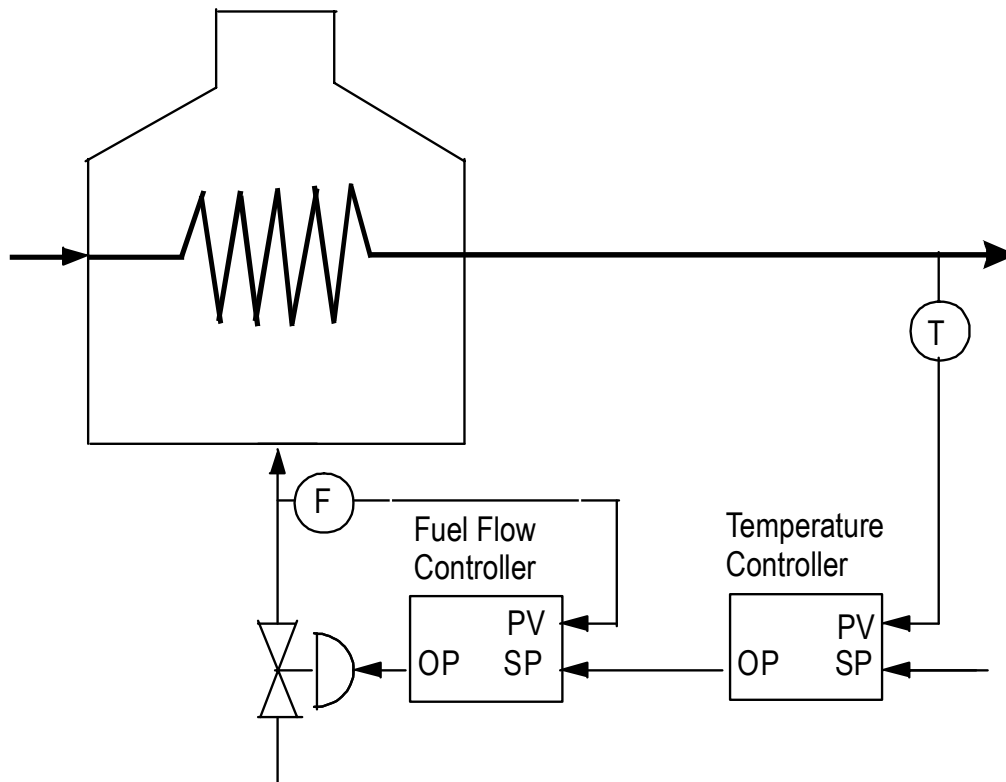


Figure 16 Simple cascade control loop example.

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 0 to 65535. 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, NONE, and NORMAL. 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, PROGRAM, and NORMAL. 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 NOSLED.
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

Configuration Tab	Description
	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. • 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.
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, BCAScacle, CAScacle,

Configuration Tab	Description
	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 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 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.

Configuration Tab	Description
Output	- 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.
	- 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

Configuration Tab	Description
	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. ‘ • 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.

Configuration Tab	Description
	• 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 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.
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

Configuration Tab	Description
	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 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

Configuration Tab	Description
	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. 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

Configuration Tab	Description
	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. – 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 OPEXL0LM 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.

Configuration Tab	Description
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.

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 Figure 17 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 Figure 17.
- 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.

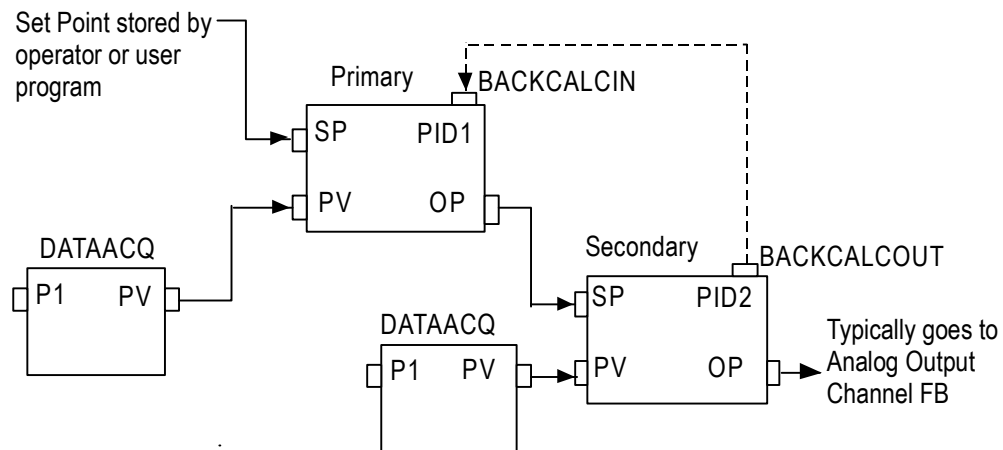


Figure 17 Functional block diagram of typical PID cascade operation.

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: Figure 18 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 Figure 18 depicts a loaded configuration in Monitoring mode.

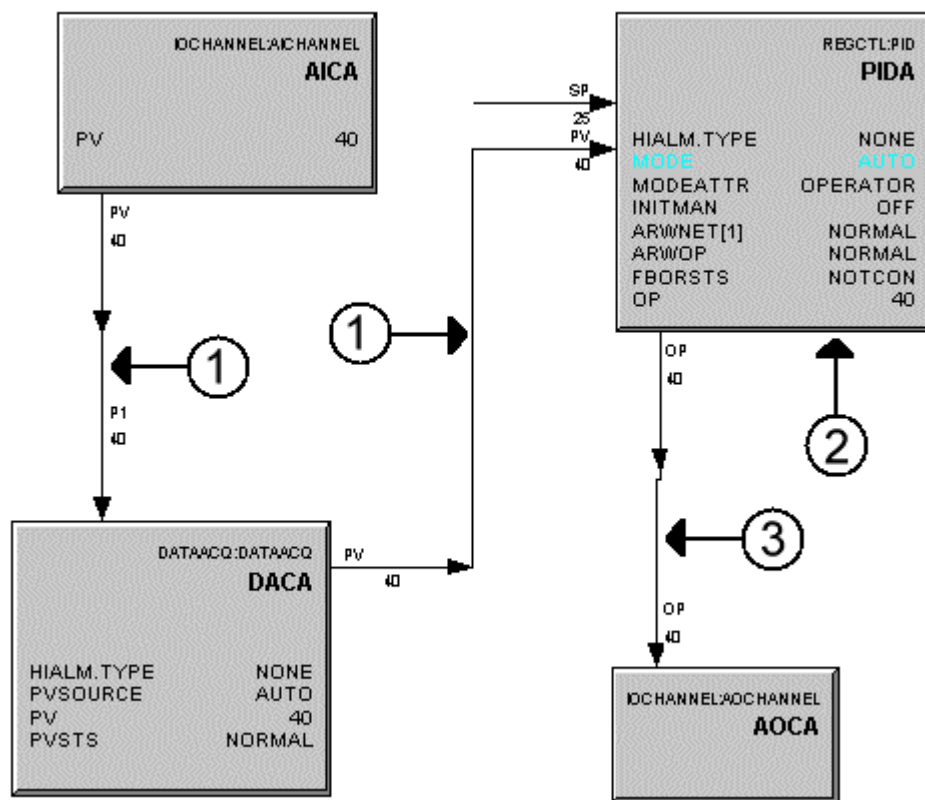


Figure 18 Example of CB configuration using a PID block for single loop control.

The following table includes descriptions of the callouts in Figure 18.

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.

Callout	Description
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 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: Figure 19 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 Figure 19 depicts a loaded configuration in Monitoring mode.

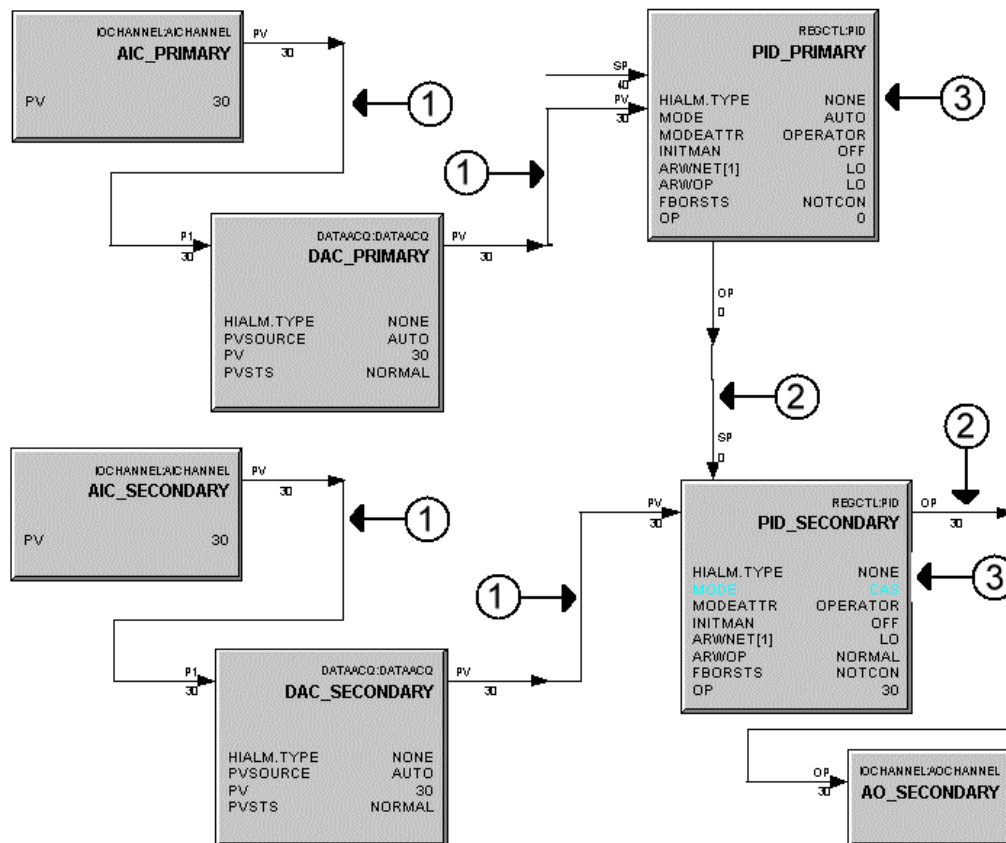


Figure 19 Example of CB configuration using two PID blocks for cascade loop control.

The following table includes descriptions of the callouts in Figure 19.

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 Figure 18 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.

$$\text{OPBIAS.FLOAT} = \text{CV}_{\text{INIT}} - (\text{CV}_{\text{UNBIASED}} + \text{OPBIAS.FIX})$$

Where:

CV_{INIT} = initialization value received from the secondary

$\text{CV}_{\text{UNBIASED}}$ = 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 ramp rate 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:

$$cycles\_per\_min = \text{number of times the function block executes per minute (calculated)}$$

- 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.

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: $\text{SPTVRATE} = \text{SPTVNORMRATE}$ $\text{SPTVTIME} = (\text{SPTV} - \text{SP}) / \text{SPTVRATE}$ • Otherwise,,: $\text{SPTVRATE} = \text{NaN}$ $\text{SPTVTIME} = 0 \text{ (That is, do a step change.)}$

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 Figure 20. 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: $\text{SPTVRATE} = (\text{SPTV} - \text{SP}) / \text{SPTVTIME}$ Otherwise: $\text{SPTVRATE} = \text{NaN}$ - 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: $\text{SPTVRATE} = \text{SPTVNORMRATE}$ $\text{SPTVTIME} = (\text{SPTV} - \text{SP}) / \text{SPTVRATE}$ Otherwise, : $\text{SPTVRATE} = \text{NaN}$ $\text{SPTVTIME} = 0$ (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.

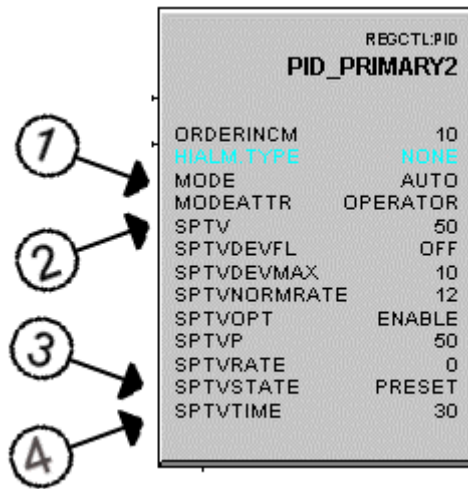


Figure 20 PID block with SP ramping parameters configured for monitoring.

The following table includes descriptions of the callouts in Figure 20.

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.
-

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:

$$CV = K * L^{-1} \left[\left(1 + \frac{1}{T1_S} + \frac{T2_S}{1 + a * T2_S} \right) * (PVP_S - SPP_S) \right]$$

- 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:

$$CV = L^{-1} \left[\frac{1}{T1_s} * (PVP_s - SPP_s) \right]$$

- 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:

$$K = \text{KLIN}$$

When PV is inside the gap ($\text{SP} - \text{GAPLOLM} \leq \text{PV} \leq \text{SP} + \text{GAPHILM}$):

$$K = \text{KLIN} * \text{KMODIFGAP}$$

- 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)

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, 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.

- 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).

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.

When the override status changes from selected to unselected, the PID block does the following:

- Recomputes CV:
 - If the override offset flag is Off:
$$CV = \text{override feedback value from secondary}$$
 - If the override offset flag is On and the PID block is using direct control action:
$$CV = (\text{override feedback value from secondary}) + K * (PVP - SPP)$$
 - If the override offset flag is On, and the PID block is using reverse control action:
$$CV = (\text{override feedback value from secondary}) - K * (PVP - SPP)$$

Where:

K = overall gain
PVP = PV in percent
SPP = SP in percent

- 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.

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:

PV = PV in engineering units



ATTENTION

You can use SECINITOPT to ignore override requests from the secondary.

PID parameters



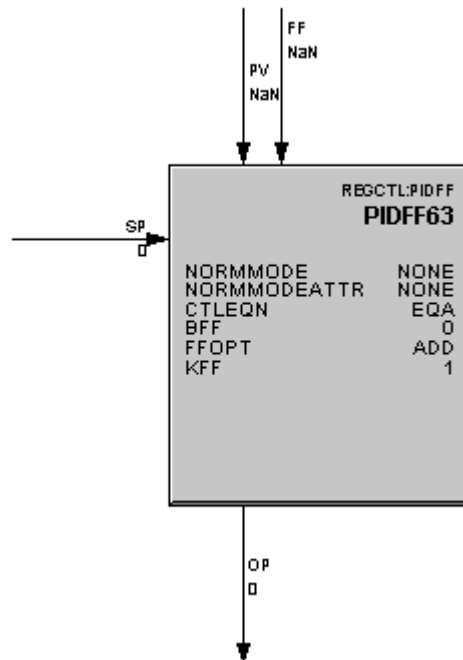
REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the PID 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. Figure 21 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.

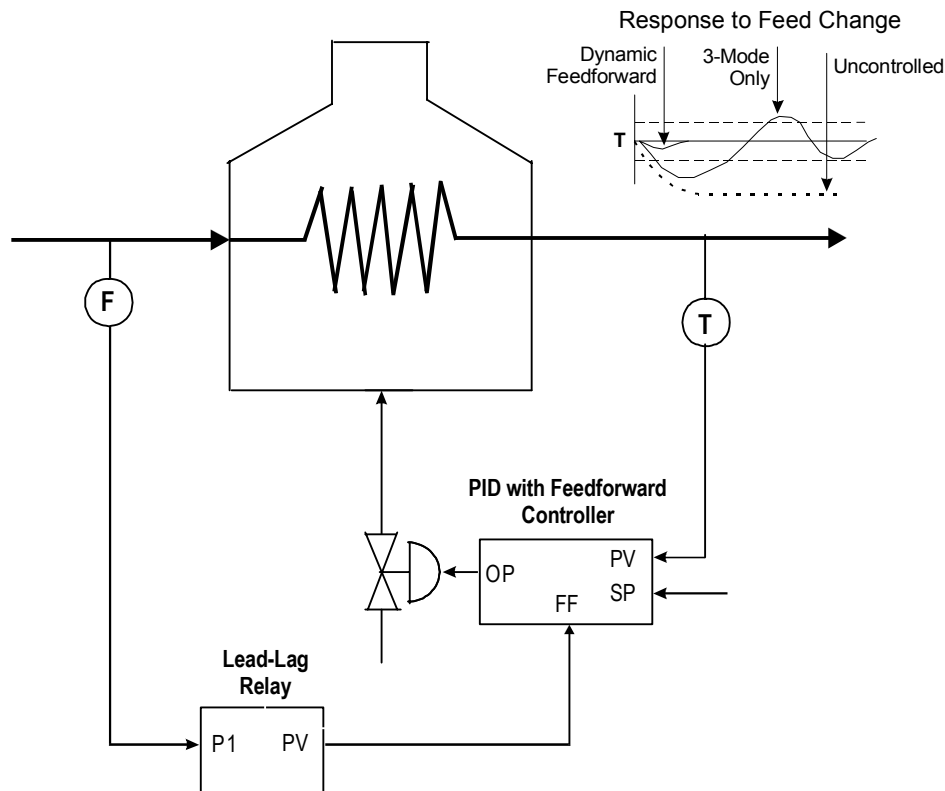


Figure 21 Simple PID with feedforward control loop example.

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 0 to 65535. 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 MODEATTR to Operator, disables external mode

Configuration Tab	Description
	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, NONE, and NORMAL. 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, PROGRAM, and NORMAL. 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 example, if the MODE is CAScade and the block is

Configuration Tab	Description
	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 interlock alarm. The selections are NO SHED,

Configuration Tab	Description
	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 NOSLED.
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 for the gain (K) value. If K is less than this value, it is

Configuration Tab	Description
	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. - 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

Configuration Tab	Description
	range is 0.0 to 1.0.
	- Linear Gain Factor (K_{LIN}) – 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: $K = K_{LIN} * K_{MODIFEXT}$ See External Gain in the Gain Options section for more details. The default value is 1. - 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 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

Configuration Tab	Description
	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 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.

Configuration Tab	Description
	• 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.
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

Configuration Tab	Description
	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 or 250. The default value is Not a Number (NaN),

Configuration Tab	Description
	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 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.
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 (OPHI.ALM.FL) – OP Low (OPLO.ALM.FL) – Deviation High (DEVHI.ALM.FL) – Deviation Low (DEVLO.ALM.FL) – Advisory Deviation (ADVDEVALM.FL) – Safety Interlock (SIALM.FL) – Bad Control (BADCTL.ALM.FL) • Enable Alarm (ADVDEVOPT and SIALM.OPT) – Lets

Configuration Tab	Description
	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 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

Configuration Tab	Description
	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 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 SCM Interface and CM Interaction section in this document for selection details. The default selection is ONESHOT. The selections are: – None – ONESHOT

Configuration Tab	Description
	– 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. – 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 OPEXL0LM 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

Configuration Tab	Description
	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.

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 Figure 22 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 Figure 22.
- PIDFF pulls FF from the LEADLAG function block as shown in Figure 22.
- The PV, FF, and OP connections are all good which means there are no communication errors or timeouts.

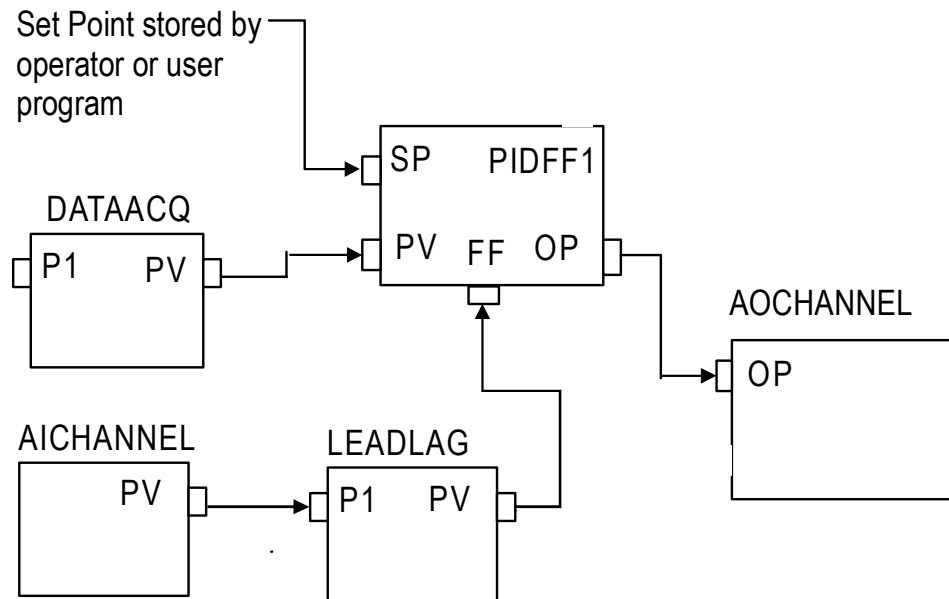


Figure 22 Functional block diagram of typical PID feedforward operation.

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.

$$\text{OPBIAS.FLOAT} = \text{CV}_{\text{INIT}} - (\text{CV}_{\text{UNBIASED}} + \text{OPBIAS.FIX})$$

Where:

CV_{INIT} = initialization value received from the secondary

$\text{CV}_{\text{UNBIASED}}$ = 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 $\text{CV} + \text{OPBIAS.FIX}$ should be the same as CV_{INIT} 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.

$$\text{OPBIAS.FLOAT} = \text{OPBIAS.FLOAT} - (\text{OPBIAS.RATE} / \text{cycles\_per\_Min})$$

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.

Regulatory Control

PIDFF (PID with Feedforward) Block

- 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.

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: $\text{SPTVRATE} = \text{SPTVNORMRATE}$ $\text{SPTVTIME} = (\text{SPV} - \text{SP}) / \text{SPTVRATE}$ - Otherwise: $\text{SPTVRATE} = \text{NaN}$ $\text{SPTVTIME} = 0 \text{ (That is, do a step change.)}$
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 ($\text{PV} - \text{SP}$) 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 Figure 23. 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: $\text{SPTVRATE} = (\text{SPTV} - \text{SP}) / \text{SPTVTIME}$ Otherwise: $\text{SPTVRATE} = \text{NaN}$ - 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: $\text{SPTVRATE} = \text{SPTVNORMRATE}$ $\text{SPTVTIME} = (\text{SPTV} - \text{SP}) / \text{SPTVRATE}$ Otherwise: $\text{SPTVRATE} = \text{NaN}$ $\text{SPTVTIME} = 0$ (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.

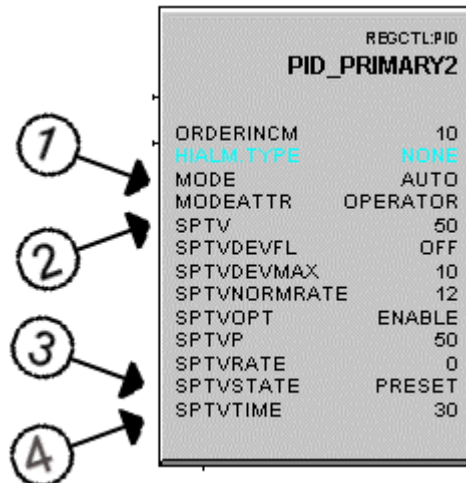


Figure 23 PID block with SP ramping parameters configured for monitoring.

The following table includes descriptions of the callouts in Figure 23.

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.
-

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:

$$CV = CV_{PID} * (KFF + BFF)$$

Regulatory Control

PIDFF (PID with Feedforward) Block

- 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:

$$CV = K * L^{-1} \left[\left(1 + \frac{1}{T1_S} + \frac{T2_S}{1 + a * T2_S} \right) * (PVP_S - SPP_S) \right]$$

- 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:

$$CV = L^{-1} \left[\frac{1}{T1_S} * (PVP_S - SPP_S) \right]$$

Regulatory Control

PIDFF (PID with Feedforward) Block

- 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:

$$K = \text{KLIN}$$

When PV is inside the gap ($\text{SP} - \text{GAPLOLM} \leq \text{PV} \leq \text{SP} + \text{GAPHILM}$):

$$K = \text{KLIN} * \text{KMODIFGAP}$$

Regulatory Control

PIDFF (PID with Feedforward) Block

- 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)

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, 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.

- 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).

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:

$$CV = (\text{override feedback value from secondary})$$

$$CV_{PID} = (\text{override feedback value from secondary}) / (KFF * FF + BFF)$$

- If the override offset flag is On and the PID is using direct control action, then:

$$CV = (\text{override feedback value from secondary}) + K * (PVP - SPP) * (KFF * FF + BFF)$$

$$CV_{PID} = (\text{override feedback value from secondary}) / (KFF * FF + BFF) + K * (PVP - SPP)$$

- If the override offset flag is On and the PID is using reverse control action, then:

$$CV = (\text{override feedback value from secondary}) - K * (PVP - SPP) * (KFF * FF + BFF)$$

$$CV_{PID} = (\text{override feedback value from secondary}) / (KFF * FF + BFF) - K * (PVP - SPP)$$

Regulatory Control

PIDFF (PID with Feedforward) Block

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

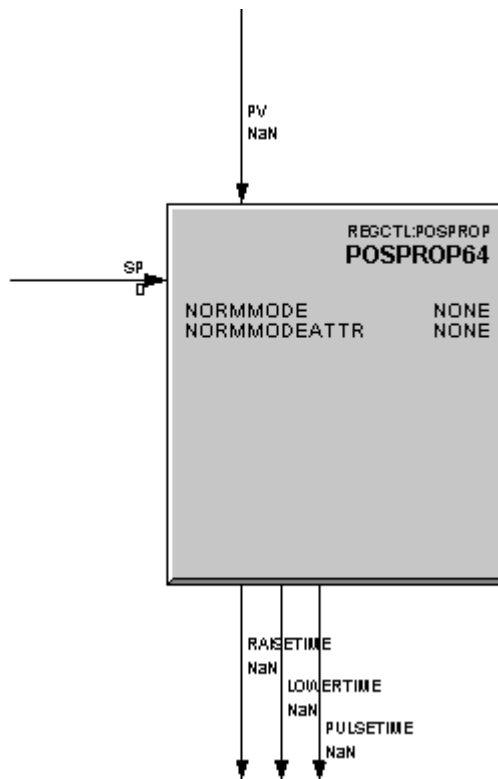
Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the PIDFF block.

POSPROP (Position Proportional) Block

Description

The POSPROP (Position Proportional) block provides pulsed digital outputs to drive a final control element to the desired position. The only valid output destinations are to Digital Output Channel blocks or the Pulse Count and Pulse Length blocks.

The POSPROP block requires a process variable (PV) and a set point (SP) as its inputs. The digital outputs are pulsed at time intervals specified by the cycle time parameter and the pulse width is proportional to the error signal. It looks like this graphically:



Each POSPROP 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 0 to 65535. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. • PVEU Range Hi (PVEUHI) – Lets you specify the high input range limit that represents 100% full scale input for the block. The default value is 100. • PVEU Range Lo (PVEULO) – Lets you specify the low input range limit that represents the 0 full scale 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: – NO_SHED - Idle. – SHEDHIGH - Raise. – SHEDLOW – Lower – SHEDSAFE – Depends on Safe State. – SHEDHOLD - Idle

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, NONE, and NORMAL. 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, PROGRAM, and NORMAL. 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). • 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 selection is NOSLED.
Algorithm	• Cycle Time (CYCLETIME) – Lets you specify a pulse cycle time in seconds. The default value is 10 seconds. • Extra Pulse Time Option (EXTRAPULSE) – Lets you specify whether or not to include the extra pulse time (EXTRAPULSETM) calculated over a maximum pulse in the algorithm. The default selection is OFF. • [Raise] Output Desc (RAISEDESC) – Lets you specify a description of up to 15 characters for the raise output. • [Raise] Overall Gain (KR) – Lets you specify an overall gain for the raise pulse generation. The default value is 1. • [Raise] Output Stroke Rate (RAISERATE) – Lets you specify a rate in percent per second for the raise stroke for the final control element. The default value is 100.

Configuration Tab	Description
	• [Raise] Stiction Compensation (STICTIONR) – Lets you specify a stiction compensation in seconds for raising the final control element. The default value is 0. • [Raise] Backlash Compensation (BACKLASHR) – Lets you specify a backlash compensation when raising the final control element. The default value is 0. • [Raise] Min. Pulse Time (MINPULSER) – Lets you specify the minimum pulse time in seconds for the raise pulses. The default value is 0. • [Raise] Max. Pulse Time (MAXPULSER) – Lets you specify the maximum pulse time in seconds for the raise pulses. The default value is 60. • [Raise] Error Deadband (ERRORDBR) – Lets you specify the error deadband in percent for the raise pulses. • Safe Output Command (SAFEOPCMD) – Lets you select the output mode to shed to for Bad control condition. The default selection is Idle. • Manual Pulse Time (MANPULSETIME) – Lets you specify the pulse time in seconds to be used in Manual mode. The default value is 1. • No Command (PULSECMDTEXT[0]) – Lets you specify a text description for the no command condition. The default is Idle. • Low Command (PULSECMDTEXT[1]) – Lets you specify a text description for the Low Command condition. The default text is Raise. • Raise Command (PULSECMDTEXT[2]) – Lets you specify a text description for the Raise Command condition. The default text is Lower. • [Lower] Output Desc (LOWERDESC) – Lets you specify a description of up to 15 characters for the lower output. • [Lower] Overall Gain (KL) – Lets you specify an overall gain for the lower pulse generation. The default value is 1. • [Lower] Output Stroke Rate (LOWERRATE) – Lets

Configuration Tab	Description
	you specify a rate in percent per second for the lower stroke for the final control element. The default value is 100. • [Lower] Stiction Compensation (STICTIONL) – Lets you specify a stiction compensation in seconds for lowering the final control element. The default value is 0. • [Lower] Backlash Compensation (BACKLASHL) – Lets you specify a backlash compensation when lowering the final control element. The default value is 0. • [Lower] Min. Pulse Time (MINPULSEL) – Lets you specify the minimum pulse time in seconds for the lower pulses. The default value is 0. • [Lower] Max. Pulse Time (MAXPULSEL) – Lets you specify the maximum pulse time in seconds for the lower pulses. The default value is 60. • [Lower] Error Deadband (ERRORDBL) – Lets you specify the error deadband in percent for the lower pulses.
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

Configuration Tab	Description
	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 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 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

Configuration Tab	Description
Alarms	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.
	• 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: – 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. • 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 (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 (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, DEVHIALM.DB and DEVLOALM.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, DEVHIALM.TM and DEVLOALM.TM) when the CM is loaded. If you configure the individual alarm parameters as

Configuration Tab	Description
SCM	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, DEVHIALM.DBU and DEVLOALM.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 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 - SEMICON - 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

Configuration Tab	Description
	block: – NONE - No changes. – MAN - Set MODEREQ = MANUAL. – AUTO - Set MODEREQ = AUTOMATIC. – CAS - Set MODEREQ = CASCADE. – FIXEDOP - Set OPREQ = Configured Value (Not applicable to this block). – 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). • 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.

Function

The POSPROP block is typically used to step a valve open or closed, raise or lower a rotary device, or move the plates of a pulp mill refiner together or apart.

The POSPROP block compares the error signal ($PV - SP$) with an error deadband for the raise and lower directions at an interval based on the configurable cycle time parameter (CYCLETIME). You can also configure the raise and lower deadband values that are denoted as the parameters ERRORDBR and ERRORDBL, respectively.

The block generates a raise pulse, when the PV is less than the SP minus the raise error deadband (ERRORDBR); or a lower pulse, when the PV is greater than the SP plus the lower error deadband (ERRORDBL) to reduce the error.

The pulse duration determines the magnitude of a pulse - the longer the duration, the bigger the pulse. The POSPROP block will not issue a raise or lower pulse that is longer than the configured cycle time (CYCLETIME) or the respective maximum pulse time parameter MAXPULSER or MAXPULSEL, whichever is smaller. The block uses the following values in its pulse duration calculation.

- Error signal ($PV - SP$)
- Raise or lower gain setting (KR or KL)
- Raise or lower pulse stroke rate (RAISERATE or LOWERRATE)
- Additional raise or lower pulse time (RAISEDEADTM or LOWERDEADTM) based on stiction compensation (STICTIONR or STICTIONL), when a motor starts up; or backlash compensation (BACKLASHR or BACKLASHL), when a motor changes direction.
- Minimum raise or lower pulse time (MINPULSER or MINPULSEL)

The calculation uses the additional pulse time and minimum pulse width parameters to keep noise from initiating continuous changes to the final control element. This block prevents instantaneous reversals by adding backlash compensation time (BACKLASHR or BACKLASHL) before commanding direction changes.

Figures 24 and 25 show examples of position proportional control loops to maintain a desired valve position using raise and lower pulse outputs or pulsetime output in conjunction with a pulse length or pulse count block, respectively. In these examples, the set point (SP) is the desired valve position and the PV is the actual valve position.

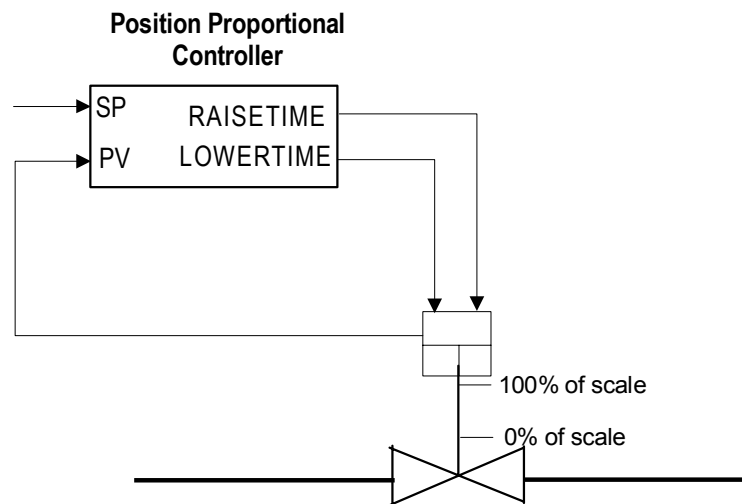


Figure 24 Example of Position Proportional loop for controlling valve position.

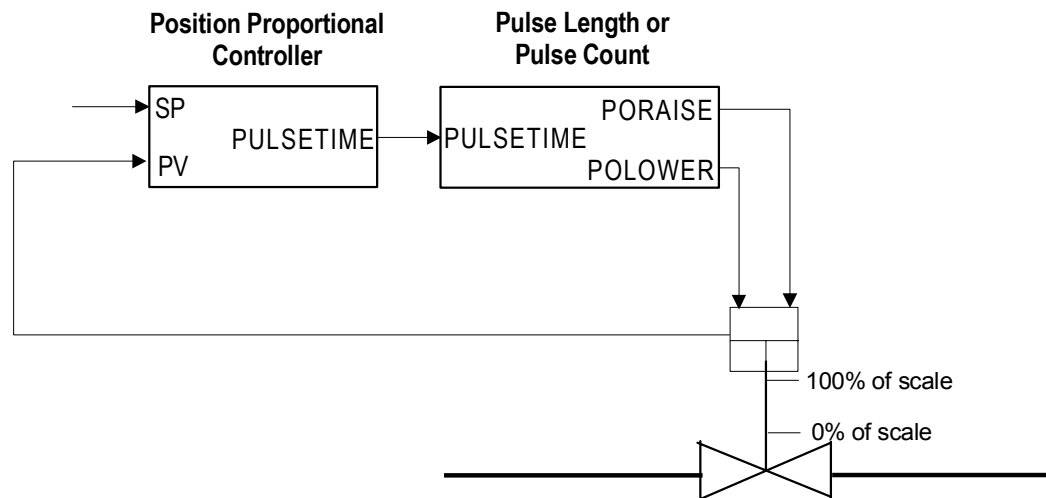


Figure 25 Example of Position Proportional loop for controlling valve position through pulse length or pulse count function.

Operating modes and mode handling

The POSPROP block operates in the following modes:

- MAN (MANual)
 - If mode is MANual, output may be stored by the operator through group or detail display in Station using designated Raise/Lower keys or buttons; 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 another function block; if the other block is off-control (that is, inactive or initializing) or the connection is bad, the POSPROP block invokes timeout processing.

Required inputs

The required number of inputs is determined by the mode of the POSPROP 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.
 - The POSPROP block may have one primary or none, depending on whether SP is configured or not; there is one primary per initializable input.

The optional raise and lower flag inputs (RAISELMFL and LOWERLMFL) may be set externally to inhibit raise and lower pulses, respectively. These optional inputs can be pulled from other function blocks.

Input ranges and limits

- You must specify a PV engineering unit range through the configurable PVEUHI and PVEULO parameters.
 - 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.
- The POSPROP 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 POSPROP block clamps it to the appropriate limit and sets the input 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.

Output

The POSPROP block has the following initializable outputs:

- RAISETIME = Raise pulse duration.
- LOWERTIME = Lower pulse duration.
- PULSETIME = Pulse duration.

You can connect RAISETIME and LOWERTIME outputs to DOCHANNEL blocks. You must connect the PULSETIME output to a PULSELENGTH or PULSECOUNT block whose output is then connected to a DOCHANNEL block. The PULSELENGTH or PULSECOUNT block sends the pulse duration from the POSPROP block to the DOCHANNEL block which generates device-specific ON/OFF commands.

(Note that you can connect the PULSETIME or RAISETIME output to the ONPULSE or OFFPULSE parameter of a DOCHANNEL block to cause a pulse of desired time. Since the ONPULSE and OFFPULSE parameters only accept positive values, you can not connect the LOWERTIME output to these parameters.)

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 a PID block to SP of a POSPROP block, Control Builder automatically creates the BACKCALCOUT to BACKCALCIN connection. In this case, the POSPROP block provides the SP input range (PVEUHI and PVEULO) to the primary PID block through the BACKCALC connection. The PID block uses this for its output range (CVEUHI/CVEULO).

Output ranges

The POSPROP block uses the maximum and minimum pulse parameters to define pulse duration ranges and limits.

- MAXPULSER and MAXPULSEL define the maximum pulse time in the Raise and Lower directions, respectively. The POSPROP block will not issue a Raise/Lower pulse with a duration that exceeds these values. If the output and CYCLETIME are greater than MAXPULSER/MAXPULSEL, the output is clamped to MAXPULSER/MAXPULSEL.
- MINPULSER and MINPULSEL define the minimum pulse time in the Raise and Lower directions, respectively. The POSPROP block will not issue a Raise/Lower pulse with a duration that is less than these values. If the output is less than MINPULSER/MINPULSEL, the output retains its old value.

(Note that the POSPROP block does not use these common regulatory control block range and limit parameters: CVEUHI, CVEULO, OPHILM, OPLOLM, OPEXHILM, and OPEXLOLM.)

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: $\text{SPTVRATE} = \text{SPTVNORMRATE}$ $\text{SPTVTIME} = (\text{SPTV} - \text{SP}) / \text{SPTVRATE}$ • Otherwise,: $\text{SPTVRATE} = \text{NaN}$ $\text{SPTVTIME} = 0 \text{ (That is, do a step change.)}$
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 ($\text{PV} - \text{SP}$) 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 Figure 26. You can access these parameters to invoke and monitor SP ramping while monitoring the control strategy through Control Builder or the 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: $\text{SPTVRATE} = (\text{SPTV} - \text{SP}) / \text{SPTVTIME}$ Otherwise: $\text{SPTVRATE} = \text{NaN}$ - 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: $\text{SPTVRATE} = \text{SPTVNORMRATE}$ $\text{SPTVTIME} = (\text{SPTV} - \text{SP}) / \text{SPTVRATE}$ Otherwise: $\text{SPTVRATE} = \text{NaN}$ $\text{SPTVTIME} = 0 \text{ (That is, do a step change.)}$
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.

Regulatory Control
POSPROP (Position Proportional) Block

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.

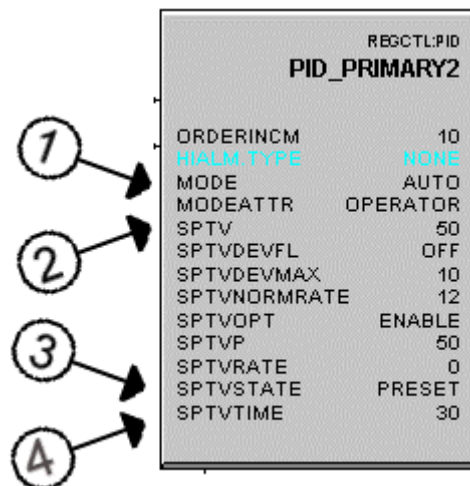


Figure 26 PID block with SP ramping parameters configured for monitoring.

The following table includes descriptions of the callouts in Figure 26.

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).

You select the Enable PV Tracking selection on the block's configuration form to enable the function (PVTRAKOPT = Track).

Typically, PV tracking is configured for POSPROP blocks in a cascade configuration strategy. This allows the POSPROPs to resume control with no error after initialization or when they are taken out of Manual mode.

If PV tracking is configured, the POSPROP block sets SP equal to PV (subject to SP limits) when either of the following conditions exist:

- POSPROP block is in Manual mode
- POSPROP 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 POSPROP block is in Auto mode.
 - a) If POSPROP block is in Auto mode, it means SP is normally stored by the user.
 - b) If PV tracking is initiated, this value is lost.
-

Timeout monitoring

- If mode is CASCADE, the POSPROP block performs timeout monitoring on SP – if a good SP value is not received within a predefined time, the POSPROP 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.

- If mode is CAScade and SP times out, the POSPROP 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 POSPROP 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.

Equations

The POSPROP block generates Raise and Lower pulses at a rate specified by the configurable cycle time (CYCLETIME) parameter. It calculates the pulse duration at the beginning of each cycle as follows.

- If PVP is less than (SPP – ERRORDBR) and the Raise limit flag (RAISELMFL) is OFF, then issue a Raise pulse with a duration of:

$$\text{RAISETIME} = \text{KR} * (\text{SPP} - \text{PVP}) / \text{RAISERATE} + \text{RAISEDEADTM} + \text{EXTRAPULSE TM}$$

- If PVP is greater than (SPP + ERRORDBL) and the Lower limit flag (LOWERLMFL) is OFF, then issue a Lower pulse with a duration of:

Regulatory Control

POSPROP (Position Proportional) Block

$$\text{LOWERTIME} = \text{KL} * (\text{PVP} - \text{SPP}) / \text{LOWERRATE} + \text{LOWERDEADTM} + \text{EXTRAPULSEM}$$

Where:

EXTRAPULSEM = The extra pulse time leftover from the last control interval, if you configured the Extra Pulse Time Option (EXTRAPULSE) to be ON.

KL = Overall gain for Lower pulse generation.

KR = Overall gain for Raise pulse generation.

LOWERDEADTM = Lower dead time in seconds. This is STICTIONL, if the last pulse was also a lower pulse; or is BACKLASHL, if the last pulse was a Raise pulse.

LOWERRATE = Lower stroke rate in percent per second.

LOWERTIME = Lower pulse time in seconds.

PVP = PV in percent.

RAISEDEADTM = Raise dead time in seconds. This is STICTIONR, if the last pulse was also a Raise pulse; or is BACKLASHR, if the last pulse was a Lower pulse.

RAISERATE = Raise stroke rate in percent per second.

RAISETIME = Raise pulse time in seconds.

SPP = SP in percent.

- The PULSETIME output is set to either the RAISETIME or –LOWERTIME, when either RAISETIME or LOWERTIME is non-zero.

Control Initialization

The POSPROP block accepts initialization information from its three initializable outputs: RAISETIME, LOWERTIME, and PULSETIME. If any output requests initialization, the POSPROP block sets its INITMAN parameter to ON. When no output requests initialization, the POSPROP block sets its INITMAN parameter to OFF. When cycling resumes after initialization, the Raise and Lower outputs are both set to OFF (or their normal states) and the cycle time is restarted.

The SP is set equal to the PV (subject to set point limits), if any of the following conditions exist:

- Mode is MANual.
- The POSPROP block is being processed for the first time after being activated.
- The POSPROP block is a secondary and is going through one-shot 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 request from the secondary.
- If SECINITOP equals Disable, it means the function block should ignore initialization request from the secondary.

Override feedback processing

The POSPROP block does not propagate override feedback data. It ignores any override feedback requests.

Raise/Lower limit switches

You can use the Raise and Lower limit flags (RAISELMFL and LOWERLMFL) to indicate the status of valve position limit switches. These flags are usually set by bringing limit indicators from a SWITCH or Logic block.

When the RAISELMFL is ON, the POSPROP block does not output Raise pulses; and when the LOWERLMFL is ON, the block does not output Lower pulses.

Bad control processing

The action the POSPROP takes during Bad control conditions depends upon how you have configured the Bad Control Option (BADCTLOPT) as follows:

If Bad Control Option is. . .	Then, control processing is. . .
NO_SHED	The PULSECMD equals Idle. The POSPROP block issues no more output pulses.
SHEDHOLD	The PULSECMD equals Idle. The mode sheds to MANUal, but the POSPROP issues no new Raise or Lower pulse – the output changes to zero.
SHEDLOW	The mode sheds to MANUal and POSPROP issues a Lower pulse (LOWERTIME) that equals 10 times the Manual Pulse Time (MANPULSETM) and PULSETIME output equals – LOWERTIME until the PV is less than or equal to the PVEULO or the Lower limit flag (LOWERLMFL) is ON. If the PV is bad, the test for PV less than or equal to PVEULO is ignored. Note that the POSPROP output ignores MINPULSER/MINPULSEL.
SHEDHIGH	The PULSECMD equals Raise. The mode sheds to MANUal and POSPROP issues a Raise pulse (RAISETIME) that equals 10 times the Manual Pulse Time (MANPULSETM) and PULSETIME output equals – RAISETIME until the PV is greater than or equal to the PVEUHI or the Raise limit flag (RAISELMFL) is ON. If the PV is bad, the test for PV less than or equal to PVEUHI is ignored. Note that POSPROP clamps the output at MAXPULSER/MAXPULSEL or CYCLETIME, whichever is less.

If Bad Control Option is. . .	Then, control processing is. . .
SHEDSAFE	The mode sheds to MANual. The output of the POSPROP block depends on how you configured the Safe Output Command (SAFEOPCMD) as follows: • If SAFEOPCMD equals Idle, the POSPROP generates no more output pulses. • If the SAFEOPCMD equals Raise, the POSPROP issues Raise pulses until PV is greater than or equal to PVEUHI or the Raise limit flag (RAISELMFL) comes ON. If the PV is bad, the test for PV is greater than or equal to PVEUHI is ignored. • If the SAFEOPCMD equals Lower, the POSPROP issues Lower pulses until the PV is less than or equal to PVEULO or the Lower limit flag (LOWERLMFL) comes ON. If the PV is bad, the test for PV is less than or equal to PVEULO is ignored.

POSPROP parameters



REFERENCE - INTERNAL

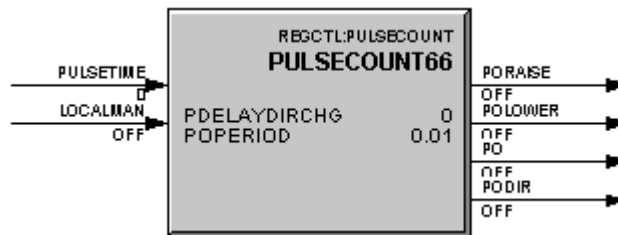
Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the POSPROP block.

PULSECOUNT Block

Description

The PULSECOUNT block generates pulses according to its pulse count control algorithm. The pulsed outputs are usually fed to Digital Output Channel blocks.

The PULSECOUNT block requires a pulse time parameter and a user configurable pulse output period as its inputs. The digital outputs are pulsed in relation to the configured period and the pulse time that is requested. It looks like this graphically:



Each PULSECOUNT 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. • 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 0 to 65535. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. • Direction Change Delay (PDELAYDIRCHG) – Lets you specify the delay time in seconds before starting a pulse output (PORAISE,POLOWER, PO) after a change in direction. This gives the final control element time to react to an upcoming change in direction. The default value is 0. • Pulse Output Period (POPERIOD) – Lets you specify the pulse output period in seconds. This generates 50% duty cycle pulses in the requested pulse time. The default value is 0.01.
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.

Function

The PULSECOUNT block is typically used in conjunction with a POSPROP block to step a valve open or closed, raise or lower a rotary device, or move the plates of a pulp mill refiner together or apart.

The POSPROP block feeds the PULSETIME input parameter to the PULSECOUNT block. This parameter is an internal structure that contains the pulse width specification (in seconds). It also contains a Serial Number that changes every time there is a new pulse width value. The PULSECOUNT block checks for a change in the Serial Number before reacting to the pulse width specification.

Figure 27 shows a sample of output pulses generated by the Pulse Count control algorithm. Keep the following things in mind when viewing Figure 27.

- The + PULSETIME or -PULSETIME come from the POSPROP block at the beginning of a control interval.
- The control interval is a property of the connected POSPROP block.
- The individual pulses are generated in relation to the configured POPERIOD. The number of pulses is determined as follows: $\text{Pulse Count} = \text{PULSETIME} / \text{POPERIOD}$
- The PODIR only changes at the beginning of a control interval. The sample pulse shown in Figure 27 has a configured Direction Change Delay (PDELAYDIRCHG) of non-zero.

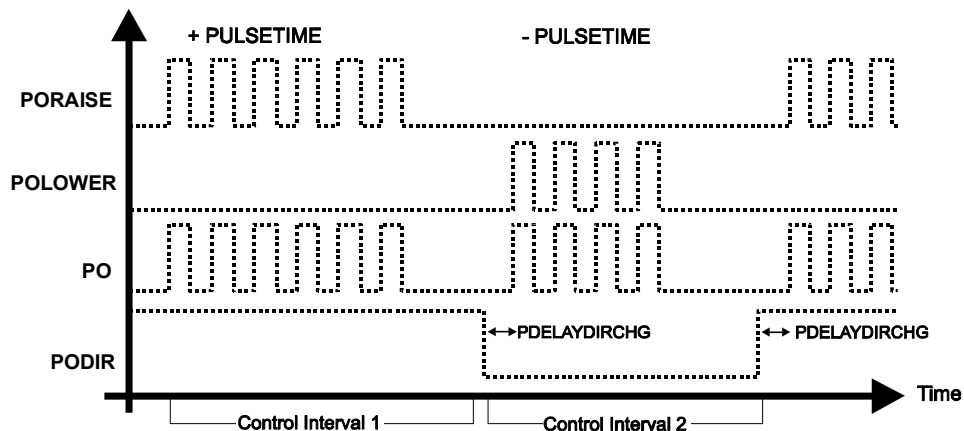


Figure 27 Example of pulse count control algorithm outputs

Required inputs

The PULSECOUNT block requires a pulse time (PULSETIME) input from another block. This is usually supplied by a POSPROP block.

The POPERIOD input is user configurable in seconds.

The PDELAYDIRCHG input is user configurable in seconds.

The optional LOCALMAN input should come from another block in a logic strategy where an ON condition means that the CEE is not controlling the output of the device. If the LOCALMAN (Local Manual Initialization) is True, all the outputs of the PULSECOUNT block are turned OFF. The back calculation (BCALCOUT), initialization manual (INITMAN), and initialization request (INITREQ) outputs are turned ON.

Output

The PULSECOUNT block has the following initializable outputs:

- PORAISE = Pulse output for Raise pulses. These pulses are generated if the pulse width specified by the PULSETIME input is positive.
- POLOWER = Pulse output for Lower pulses. These pulses are generated if the pulse width specified by the PULSETIME input is negative.
- PO = Pulse output for both Raise and Lower pulses. These pulses are generated as a logical OR between the PORAISE and POLOWER pulses.
- PODIR = Direction for PO. This output is OFF for a Lower pulse and is ON for a Raise pulse.

You normally connect PORAISE/POLOWER or PO/PODIR outputs in pairs to DOCHANNEL blocks

(Note that you can connect the PORAISE output to the ONPULSE or OFFPULSE parameter of a DOCHANNEL block to cause a pulse of desired time. Since the ONPULSE and OFFPULSE parameters only accept positive values, you can not connect the POLOWER or PO output to these parameters.)

The PULSECOUNT block has the following status outputs:

- INITMAN = Initialization manual. This output is turned ON, if the LOCALMAN input is ON or the secondary of the PULSECOUNT block is requesting initialization. It is turned OFF only if both of the requests turn OFF and the primary of the PULSECOUNT block has received the request.
- INITREQ = Initialization request. This output is turned ON, if the LOCALMAN input is ON or the secondary of the PULSECOUNT block is requesting initialization. It is turned OFF only if both of the requests turn OFF and the primary of the PULSECOUNT block has received the request.

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 PORaise from a PULSECOUNT block to ONPULSE of a DOCHANNEL block, Control Builder automatically creates the BACKCALCOUT to BACKCALCIN connection. In this case, the back calculation input for PORaise is BCALCINPOR.

PULSECOUNT parameters



REFERENCE - INTERNAL

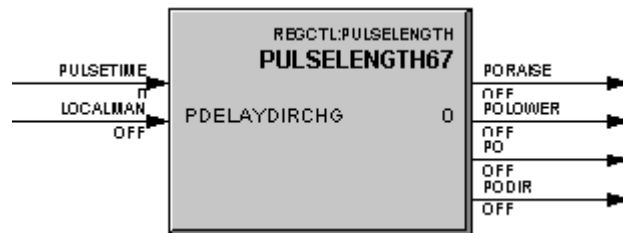
Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the PULSECOUNT block.

PULSELENGTH Block

Description

The PULSELENGTH block generates pulse trains according to its pulse length control algorithm. The pulsed outputs are usually fed to Digital Output Channel blocks.

The PULSELENGTH block requires a pulse time parameter as its input. The digital outputs are pulsed in relation to the pulse time that is requested. It looks like this graphically:



Each PULSELENGTH 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. - 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 0 to 65535. The default value is 110. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. - Direction Change Delay (PDELAYDIRCHG) – Lets you specify the delay time in seconds before starting a pulse output (PORAISE, POLOWER, PO) after a change in direction. This gives the final control element time to react to an upcoming change in direction. The default value is 0.
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.

Function

The PULSELENGTH block is typically used in conjunction with a POSPROP block to step a valve open or closed, raise or lower a rotary device, or move the plates of a pulp mill refiner together or apart.

The POSPROP block feeds the PULSETIME input parameter to the PULSELENGTH block. This parameter is an internal structure that contains the pulse width specification (in seconds). It also contains a Serial Number that changes every time there is a new pulse width value. The PULSELENGTH block checks for a change in the Serial Number before reacting to the pulse width specification.

Figure 28 shows a sample of output pulses generated by the Pulse Length control algorithm. Keep the following things in mind when viewing Figure 28.

- The + PULSETIME or -PULSETIME come from the POSPROP block at the beginning of a control interval.
- The control interval is a property of the connected POSPROP block.
- The PODIR only changes at the beginning of a control interval. The sample pulse shown in Figure 28 has a configured Direction Change Delay (PDELAYDIRCHG) of Zero (0).

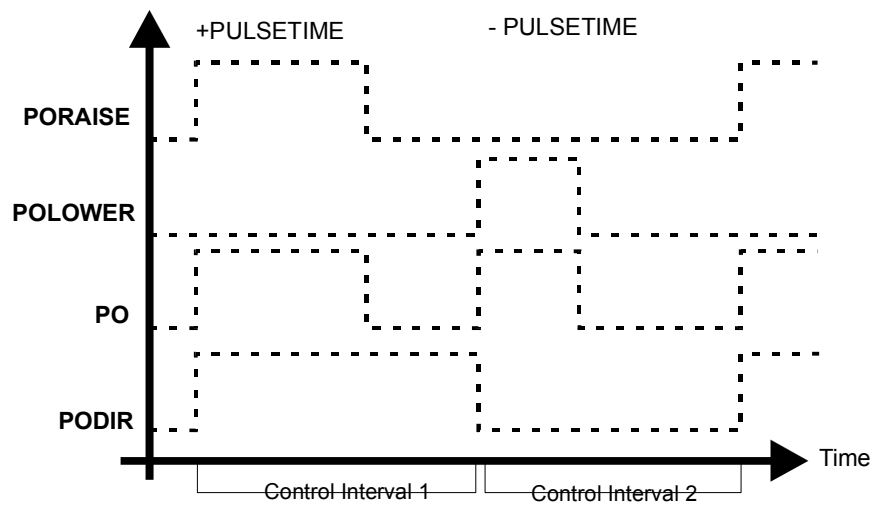


Figure 28 Example of pulse length control algorithm outputs

Required inputs

The PULSELENGTH block requires a pulse time (PULSETIME) input from another block. This is usually supplied by a POSPROP block.

The PDELAYDIRCHG input is user configurable in seconds.

The optional LOCALMAN input should come from another block in a logic strategy where an ON condition means that the CEE is not controlling the output of the device. If the LOCALMAN (Local Manual Initialization) is True, all the outputs of the PULSELENGTH block are turned OFF. The back calculation (BCALCOUT), initialization manual (INITMAN), and initialization request (INITREQ) outputs are turned ON.

Output

The PULSELENGTH block has the following initializable outputs:

- PORAISE = Pulse output for Raise pulses. These pulses are generated if the pulse width specified by the PULSETIME input is positive.
- POLOWER = Pulse output for Lower pulses. These pulses are generated if the pulse width specified by the PULSETIME input is negative.
- PO = Pulse output for both Raise and Lower pulses. These pulses are generated as a logical OR between the PORAISE and POLOWER pulses.
- PODIR = Direction for PO. This output is OFF for a Lower pulse and is ON for a Raise pulse.

You normally connect PORAISE/POLOWER or PO/PODIR outputs in pairs to DOCHANNEL blocks

(Note that you can connect the PORAISE output to the ONPULSE or OFFPULSE parameter of a DOCHANNEL block to cause a pulse of desired time. Since the ONPULSE and OFFPULSE parameters only accept positive values, you can not connect the POLOWER or PO output to these parameters.)

The PULSELENGTH block has the following status outputs:

- INITMAN = Initialization manual. This output is turned ON, if the LOCALMAN input is ON or the secondary of the PULSELENGTH block is requesting initialization. It is turned OFF only if both of the requests turn OFF and the primary of the PULSELENGTH block has received the request.

- INITREQ = Initialization request. This output is turned ON, if the LOCALMAN input is ON or the secondary of the PULSELENGTH block is requesting initialization. It is turned OFF only if both of the requests turn OFF and the primary of the PULSELENGTH block has received the request.

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 PORAISE from a PULSELENGTH block to ONPULSE of a DOCHANNEL block, Control Builder automatically creates the BACKCALCOUT to BACKCALCIN connection. In this case, the back calculation input for PORAISE is BCALCINPOR.

PULSELENGTH parameters



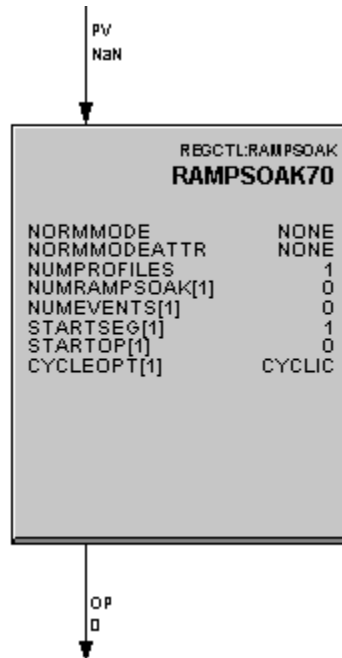
REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the PULSELENGTH block.

RAMPSOAK Block

Description

The RAMPSOAK block provides an output that tracks a user configured set point versus time profile. The block supports up to 10 separate profiles with up to 30 user configured ramp and soak segment pairs per profile. This lets you implement a set point program control function by driving the set point of another regulatory control function block. The RAMPSOAK block looks like this graphically:



The RAMPSOAK block has one analog input identified as a process variable (PV). The block monitors the PV value and guarantees that its output (OP) will not deviate from the input (PV) by more than the user configured limits.

Each RAMPSOAK 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 0 to 65535. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. • High Limit (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. • Low Limit (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). • 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, NONE, and NORMAL. 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

Configuration Tab	Description
	is to assume when the Control to Normal function is initiated through the Station display. Selections are NONE, OPERATOR, PROGRAM, and NORMAL. 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. (Note that the AUTomatic mode is not a valid initial configuration mode for the RAMPSOAK block, since the block's mode must be MANual after it is loaded to the Controller and the Control Module containing it is activated.) – 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

Configuration Tab	Description
	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 • Reset Segment Timers on Profile (RESETTIMER) – Lets you control the timers when restarting a profile. When checked (or ON), all timers are reset when the profile starts. When unchecked (or OFF), all timers resume with their previous values when the profile starts. The default is checked (or ON). (Note that whenever a new profile is loaded the RESETTIMER parameter is automatically set to ON.) • Safety Interlock Option (SIOPT) – Lets you specify MODE and OP the 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 the block is to assume if CV goes BAD. The selections are NO_SHED, SHEDHOLD, SHEDLOW, SHEDHIGH, and SHEDSAFE. The default selection is NOSLED.
Profile	• Profile ID – List box shows the configured profiles. The New, Copy, and Delete buttons let you manipulate profiles as desired. • Description (PROFILEDESC[n]) – Lets you enter a unique profile name of up to 16 characters long for the profile selected in the list box. The default name is Profilen. Where “n” equals the assigned profile number from 1 to 10. • Max Ramp Dev (MAXRAMPDEV[n]) – Lets you specify a desired maximum ramp deviation value between PV and OP to assure a guaranteed ramp rate. You can specify a different value for each profile. The default value is NaN (Not-a-Number), which means no ramp rate checking is done.

Configuration Tab	Description
	- Max Hi Soak Dev (MAXHISOAKDEV[n]) – Lets you specify a desired maximum high soak deviation value between PV and OP to assure a guaranteed soak. You can specify a different value for each profile. The default value is NaN (Not-a-Number), which means no high soak value checking is done. - Max Lo Soak Dev (MAXLOSOAKDEV[n]) – Lets you specify a desired maximum low soak deviation value between PV and OP to assure a guaranteed soak. You can specify a different value for each profile. The default value is NaN (Not-a-Number), which means no low soak value checking is done. - Starting OP Value (STARTOP[n]) – Lets you specify a desired starting output (OP) value for each profile. The default value is 0. - Starting Segment ID (STARTSEG[n]) – Lets you specify the starting segment for each profile. The ramp segments have odd numbers and the soak segments have even numbers. The default value is 1. - Cycle Option (CYCLEOPT) – Lets you select how you want the profiles to be cycled. The SINGLE selection means that the selected (running) profile will stop after it executes its last ramp/soak pair. The CYCLIC selection means that the selected (running) profile will continuously cycle from start to end. This means it will restart at the starting segment once it executes the last ramp/soak pair. The ROUNDROBIN selection means that every consecutive profile configured for ROUNDROBIN will be executed in order through the last profile. This means that after the last ramp/soak pair in the first profile is executed the execution of the next profile begins and so on until the last profile is executed or the next profile is configured for SINGLE or CYCLIC action. - RampSoak Pair ID – Lets you configure ramp/soak pairs for the selected profile by entering desired Ramp Rate (RAMPRATE[n,s]), Soak Value (SOAKVAL[n,s]), and Soak Time (SOAKTIME[n,s]) in minutes. Where “s” equals the number of the ramp/soak pair from 1 to 30. - Even ID – Lets you configure up to 16 event flags

Configuration Tab	Description
	(EVENTFL[n,e]) for segments in the selected profile by entering the segment number (EVENTSEGID[n,e]), the start time (EVENTBGNTIME[n,e]) in minutes counted from the beginning of the selected segment when the event flag is turned ON, and the stop time (EVENTENDTIME[n,e]) in minutes counted from the beginning of the selected segment when the event flag is turned OFF. Where “e” equals the number of the event from 1 to 16.
Profile Graph	- Graph – Shows you a graphic representation of the configured set point versus time profile. - Profile ID – List box shows configured profiles that you can select for display in the graph. - No. of Segments – Number of segments in selected profile. - Target Profile Time (TOTALTIME[n]) – Total time in minutes to complete the selected profile. - Static – Provides an array of the configured event flags. A number in a box represents a configured event. Click the numbered box to display the event markers on the graph.
Active Profile Graph	- Graph – Shows you a graphic representation of the configured set point versus time profile with real time data when profile is running in CB Monitoring tab. - Mode (MODE) – Shows current mode selection and lets you change the mode of running profile in CB Monitoring tab. - Current Profile ID (CURPROFILEID) – Shows number of profile currently running. - No. of Segments – Shows the total number of segments in the current profile. - Current Segment ID (CURSEGID) – Shows the number of the segment currently being executed in the selected profile. - Total Elapsed Time (TOTELAPSEDTM) – Shows the total elapsed time for current profile execution. It includes time for stopped timers due to deviation

Configuration Tab	Description
	exceeding limits. - Net Elapsed Time (NETELAPSED TM) – Shows the net elapsed time for current profile execution. It does not include the time for stopped timers due to deviation exceeding limits. - Rem Soak Time (REMSOAK TIME) – Shows the remaining soak time for the current soak segment. - Soak Duration – Shows the duration of the current soak segment. - Events – Shows an array of the configured event flags for the current segment. Click the numbered box to display the event markers on the graph.
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%. (Note that you can not change this value through Monitoring mode after the configuration is loaded in the Controller.) - 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%. (Note that you can not change this value through Monitoring mode after the configuration is loaded in the Controller.) - 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%. (Note that you can not

Configuration Tab	Description
	change this value through Monitoring mode after the configuration is loaded in the Controller.) - 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%. (Note that you can not change this value through Monitoring mode after the configuration is loaded in the Controller.) - 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
	• 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).
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) – 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 checked or Yes (enabled). You can also configure the SIALM.OPT parameter as a block pin, configuration and/or monitoring parameter so it appears 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)

Configuration Tab	Description
	• Priority – Lets you set the desired priority level individually for each alarm type (OPHIALM.PR, OPLOALM.PR, DEVHIALM.PR, DEVLOALM.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, 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 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.

Configuration Tab	Description
	- AUTO - Set MODEREQ = AUTOMATIC. - CAS - Set MODEREQ = CASCADE. - FIXEDOP - Set OPREQ = Configured Value. - HOLDPV - Set SPREQ = PV. (Not applicable for RAMPSOAK block) - FIXED SP - Set SPREQ = Configured Value and SPRATEREQ = NaN. (Not applicable for RAMPSOAK block) - RAMPEDSP - Set SPTVREQ = Configured Value and SPRATEREQ = Configured Rate. (Not applicable for RAMPSOAK 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 OPEXL0LM to OPEXHILM 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.

Function

The RAMPSOAK block is typically used for automatic temperature cycling in furnaces and ovens. It can also be used for automatic startup of units and for simple batch-sequence control where the batch sequence is part of a process that is otherwise a continuous process.

The RAMPSOAK block usually feeds its output (OP) to the set point of a PID block. The PID block uses the PID algorithm to control a process variable (PV) according to the set point versus time profile OP. The PV input to the RAMPSOAK block is normally the same PV input used for the PID block.

Figure 29 shows a simple functional diagram of a PID loop with its set point driven by the output of a RAMPSOAK block according to the configured ramp and soak segments.

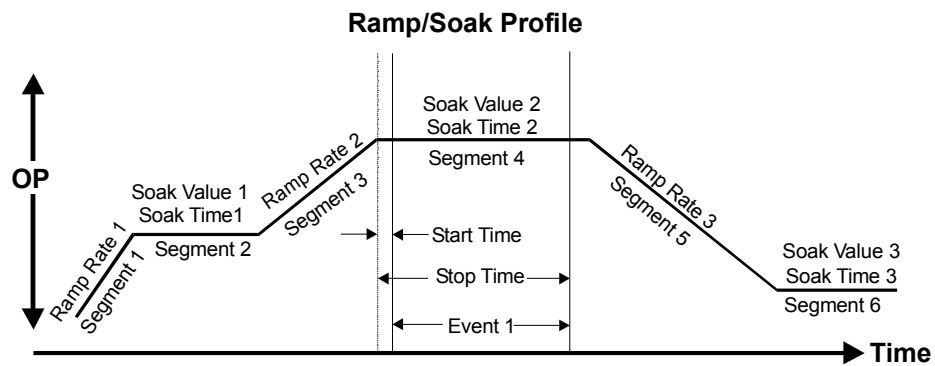
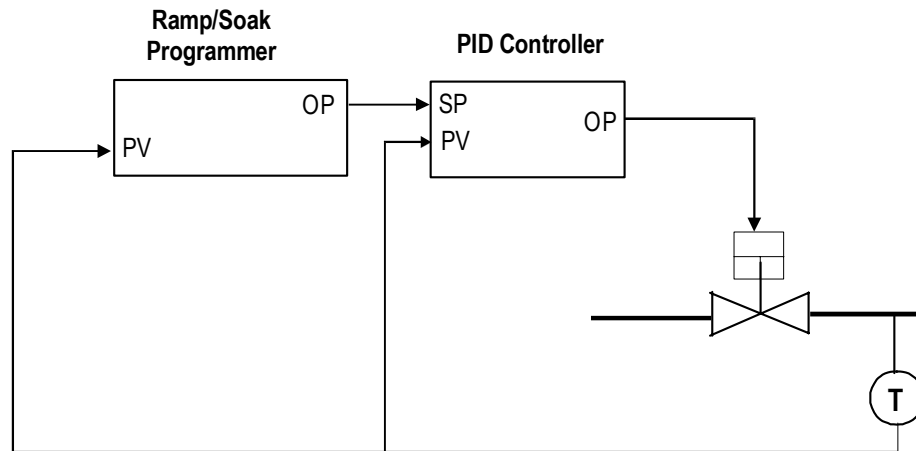


Figure 29 Functional diagram of ramp and soak (set point) programmer in PID control loop.

The RAMPSOAK block provides the following functions for a running ramp/soak profile.

- Calculates its output based on whether the current segment is a ramp or a soak.
 - If the current segment is a ramp, the block calculates the ramp output. If a guaranteed ramp rate was requested, the block makes sure the output does not deviate from the input by more than the user configured deviation (MAXRAMPDEV[n]).
 - If the current segment is a soak, the block calculates the soak output and updates the soak timers. If a guaranteed soak was requested, the block makes sure that the soak time does not transpire while the PV and CV are outside the user configured deviation limits (MAXHISOAKDEV[n] and MAXLOSOAKDEV[n]). The block stops the soak timer when the soak value exceeds the user configured deviation. It restarts the timer when the soak value returns to within limits.
- Updates all the events configured for the current profile. The block sets these timers based on the user configured event parameters: EVENTSEGID[n,e], EVENTBGNTIME[n,e], and EVENTENDTIME[n,e].

Required inputs

The RAMPSOAK block only requires a PV input for the guaranteed ramp option.

- 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.

Input ranges and limits

- You must specify a PV engineering unit range, PVEUHI and PVEULO. The default range is 0 to 100.
 - 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.
- The PID block assumes PV is within PVEUHI and PVEULO – it applies no range check – however, PV typically comes from an auxiliary or data acquisition (DATAACQ) block which applies its own limit and range checks.

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 RAMPSOAK block to SP on a PID 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 RAMPSOAK block (RAMPSOAK.OP) to the set point of a PID block (PIDA.SP), the implicit/hidden connection is made to RAMPSOAK.OPX to provide value/status data.)

Output ranges and limits

- CVEUHI and CVEULO define the full range of CV in engineering units.
 - If the RAMPSOAK block has a secondary, it brings the secondary's input range through BACKCALC and sets its CV range to that.
 - If the RAMPSOAK block has no secondary, you can configure CVEUHI and CVEULO to specify the desired range values. The default values are 100 and 0, respectively, for a default range of 0 to 100.
- OPHILM and OPLOLM define the normal high and low limits for OP as a percent of the CV range. You can also configure values for these limits. The default limits are 105% and –5%, respectively.
 - 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.
- OPEXHILM and OPEXLOLM define the extended high and low limits for OP as a percent of the CV range. You can also configure values for these limits. The default limits are 106.9% and –6.9%, respectively.
 - The operator is prevented from storing an OP value that exceeds these limits.

(Note that the RAMPSOAK block does not apply a floating bias to the output.)

Mode handling

The RAMPSOAK block supports the AUTOMATIC and MANUAL modes.



ATTENTION

You must select MANUAL as the configuration setting for the MODE parameter on the RAMPSOAK block's configuration form in the Control Builder Project tree. Control Builder generates an error if you try to load a RAMPSOAK block with a MODE configuration of AUTOMATIC to the Controller. The MODE of the RAMPSOAK block must be MANUAL after it is loaded to the Controller.

- You set the mode to AUTOMATIC to start a ramp/soak profile. When the profile is running, you can not adjust the output (OP) or the profile variables such as ramp rate, soak value, and soak time.
- You set the mode to MANUAL to stop a ramp/soak profile, including all timers. When a profile is stopped, you can change the output (OP) and adjust the profile variables

including the current segment (CURSEGID) and the remaining soak time (REMSOAKTIME), if the current segment is a soak. If you change the current segment, the profile starts at the new segment when you change from MANual to AUTOMatic mode. If the reset timer (RESETTIMER) function is ON, note that all timers are reset when the profile starts regardless of any change in the remaining soak time (REMSOAKTIME). You can **not** add or delete profiles, ramp/soak pairs or events once a configuration is loaded into the Controller. Also, Control Builder does not allow online changes in profile variables such as Rate, Soak Value, and Soak Time

Hold command

The hold command (HOLDCMD) parameter allows another function block or user program to stop the profile until some user defined condition is met.

- When the HOLDCMD changes from OFF to ON, the profile stops, including all timers.
- When the HOLDCMD changes from ON to OFF, the profile starts where it left off.

CEE idle or Control Module inactivate command

When you change the CEE from Run to Idle or the Control Module from Active to Inactive, the contained RAMPSOAK block does the following.

- Sets mode to MANual.
- Sets CV to NaN.
- Resets internal ramp/soak timers.
- Sets current profile ID to 1 (first profile).
- Sets current segment ID to 1 (first ramp segment).

Profile statistics

Since the profile may be stopped or held for several reasons, the actual profile execution may be quite different from the configured profile definition. The RAMPSOAK block maintains the following execution profile statistic parameters.

- ACTRAMPRATE[n,s] – The actual rate for each ramp segment in engineering units per minute.
- ACTSOAKVAL[n,s] – The actual end value for each ramp segment in engineering units.
- ACTSOAKTIME[n,s] – The actual duration of each soak segment in minutes.
- ACTSTARTSEG[n] – The actual starting segment number for each profile.
- ACTSTARTOP[n] – The actual starting output (OP) value for each profile.

Where “n” is the profile number and “s” is the segment number.

You can also compare the graphical representation of the configured profile and the actual profile through the Profile Graph and Active Profile Graph tabs in the block configuration form, when monitoring operation through the Monitoring tab in Control Builder.

Guaranteed ramp rate

If you configure a maximum ramp deviation (MAXRAMPDEV[n]) value for a given profile, the RAMPSOAK block makes sure that the calculated output (CV) value does not deviate from the input (PV) by more than the configured deviation value. If it does deviate, the block stops the ramping action until PV catches up with CV. The RAMPSOAK block will stop the ramping action for the following condition.

- The Absolute Value of CV–PV is greater than the maximum ramp deviation (MAXRAMPDEV[n]. Where “n” is the number of the current profile.

If the maximum ramp deviation (MAXRAMPDEV[n]) value is NaN, the RAMPSOAK block ignores the above condition.

(Note that you can also stop the ramping by setting the hold command (HOLDCMD) to ON. This lets an operator, a user program, or a logic type function block stop the ramping until some other condition is satisfied.)

Guaranteed soak time

If you configure the maximum high soak deviation (MAXHISOAKDEV[n]) and/or the maximum low soak deviation (MAXLOSOAKDEV[n]) value, the RAMPSOAK block makes sure the calculated output (CV) value is at the proper value before it starts the soak timer. The RAMPSOAK block verifies that the CV and input PV are within the configured deviation limits and it will not start the soak timer for the following conditions.

- If the input (PV) is greater than the CV.
- If the PV is less than the CV.

If the deviation exceeds the limit during a soak, the block stops the soak timer until the deviation returns to within limits and then it automatically restarts the timer.

If the MAXHISOAKDEV[n] and/or the MAXLOSOAKDEV[n] value is NaN, the RAMPSOAK block ignores the above condition or conditions, as applicable.

(Note that you can also keep the soak timer from starting by setting the hold command (HOLDCMD) to ON. This lets an operator, a user program or a logic type function block put a hold on the stop timer until some other condition is satisfied.)

Event timer functions

You can configure up to 16 event flags (EVENTFL[n,e]) to provide Boolean outputs for a specified time during a given ramp or soak segment in a given profile. This means you can have up to 16 events per profile or a total of 160 events in 10 profiles.

The following parameters are associated with each event flag.

EVENTSEGID[n,e] – Identifies the segment in a given profile to which the event applies.

EVENTBGNTIME[n,e] – The user-configured time in minutes measured from the start of the segment when the given event turns ON. This is also called the start time.

EVENTENDTIME[n,e] – The user-configured time in minutes measured from the start of the segment when the given event turns OFF. This is also called the stop time.

Note that you can configure the start time (EVENTBGNTIME[n,e]) to be greater than or equal to the stop time (EVENTENDTIME[n,e]), but such a configuration results in no event action.

Control initialization

The RAMPSOAK 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 RAMPSOAK block:
 - initializes its output
CV = initialization value from the secondary
 - sets initialization request parameters for its primary

Override feedback processing

The RAMPSOAK block does not propagate override feedback data. It ignores any override feedback requests it receives.

RAMPSOAK parameters



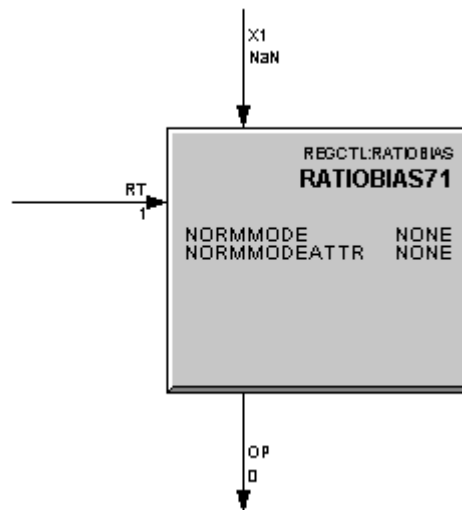
REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the RAMPSOAK block.

RATIOBIAS Block

Description

The RATIOBIAS block accepts a ratio value input (RT) and an input value (X1) to provide a calculated output based on the ratio of the input variables plus a fixed and/or a floating bias. It looks like this graphically:



Each RATIOBIAS 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. - Ratio (RT) – Lets you specify a value between 0.001 and 100 to be used for the RT input when the block is in its AUTOMATIC mode. The default value is 1. - 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 0 to 65535. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. - X1 High Limit (XEUHI) – Lets you specify the high input range value in engineering units that represents 100% full scale input for the block. The default value is 100. - X1 Low Limit (XEULO) – Lets you specify the low input range value in engineering units that represents the 0 full scale input for the block. The default value is 0 (zero). - Ratio High Limit (RTHILM) – Lets you specify the high ratio limit value in engineering units. The default value is 100. - Ratio Low Limit (RTLLOLM) – Lets you specify the low ratio limit value in engineering units. The default value is 0.001. - 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, NONE, and NORMAL. All selections

Configuration Tab	Description
	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, PROGRAM, and NORMAL. 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

Configuration Tab	Description
	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 selection is NO_SHED. • 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.
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 (NaN), which means no rate limiting is

Configuration Tab	Description
	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. - 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.
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)

Configuration Tab	Description
	– 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: – 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.

Configuration Tab	Description
	- 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. 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

Configuration Tab	Description
	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. – 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 Rate (not applicable to 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.

Regulatory Control

RATIOBIAS Block

Configuration Tab	Description
Block Preferences	Lets you change several block-viewing preferences including the color of the block's faceplate.

Function

Lets you implement a form of ratio control by using this block between two PID blocks. In this case, the output from one PID block is used as the X1 input to the RATIOBIAS block and the output from the RATIOBIAS block is used as the SP input to the second PID block.

Configuration example

Figure 30 and its companion callout description table show a sample configuration that uses a RATIOBIAS block to form a ratio control loop for quick reference. The view in Figure 30 depicts a configuration in Project mode.

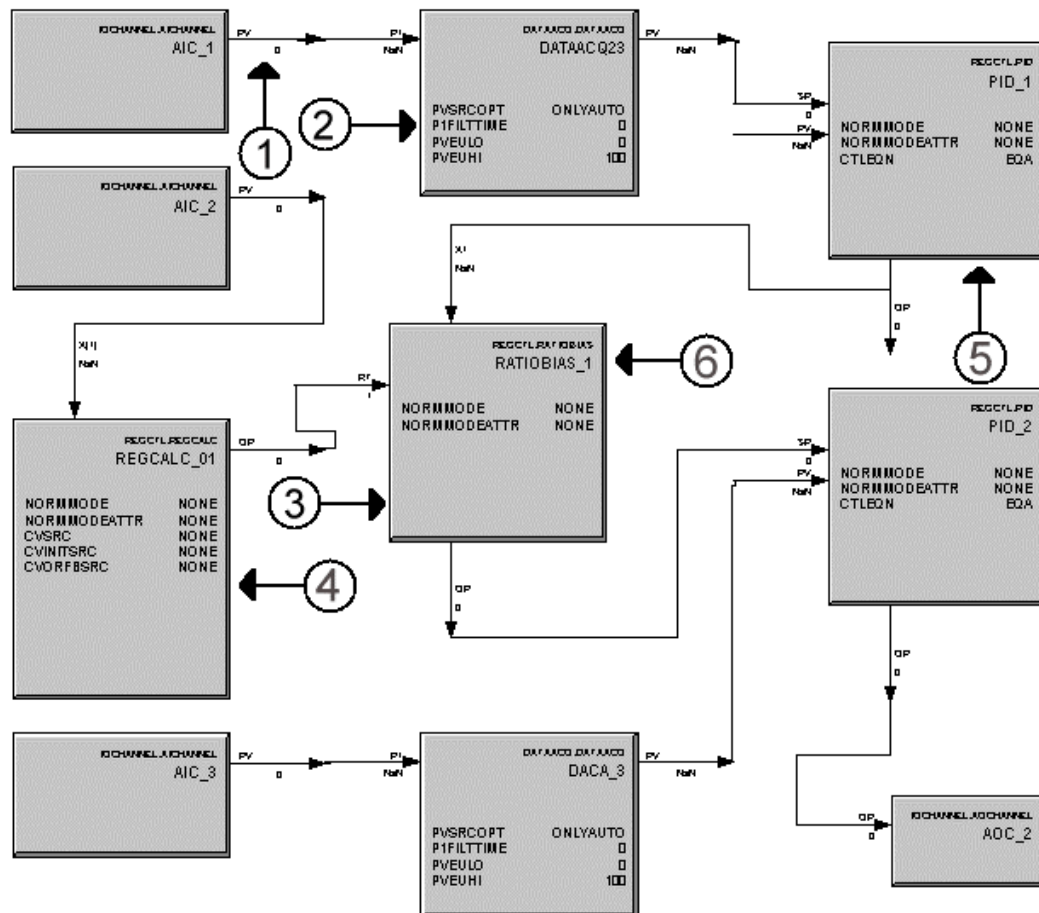


Figure 30 Example CB configuration using RATIOBIAS block.

The following table includes descriptions of the callouts in Figure 30.

Callout	Description
1	Use the PV parameter connection to carry data from the analog input to the other 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 DATAACQ block to define input range values and provide alarm monitoring on the analog input.
3	Use the RATIOBIAS block in cascade mode to accept X1 and RT primary inputs from other blocks.
4	Use the REGCALC block output (OP) to provide the RT input based on assigning expression 1 as its CV source. 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.
5	Use the PID block output (OP) to provide the X1 input. 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.
6	Control Builder creates the X1BACKCALOUT and RTBACKCALOUT hidden connections to carry BACKCAL (secondary) data from the RATIOBIAS block to the BACKCALCIN connections on X1 and RT primary blocks, respectively. The individual BACKCALCIN/BACKCALCOUT connections for each output used are automatically built by Control Builder as implicit/hidden connections.

Operating modes and mode handling

The RATIOBIAS 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 X1 and RT inputs are ignored. The block continually initializes both primaries, while in this mode.
Automatic (AUTO)	the X1 input comes from another function block and the RT input can be set by the operator or a user program. The block continually initializes the RT primary, while in this mode.
Cascade (CAS)	both X1 and RT inputs come from other function blocks.

This block requests both primaries to initialize when the mode changes from CAScade to MANual. This block requests only one primary to initialize when the mode changes from CAScade to AUTOMATIC. This block requests no primary to initialize when the mode changes from MANual to CAScade. However, it always requests the X1 primary to initialize first, and then initializes the RT based on whether or not the X1 initialization was successful.

Required inputs

A RATIOBIAS block requires one or two inputs depending on the block's Mode, as follows.

If Mode is. . .	Then, block requires. . .
Cascade	both X1 and RT inputs.
Auto	only X1 input.

- Both X1 and RT are initializable inputs. This means the block can have one or two primaries depending upon whether the RT input is required or not. There is one primary for each initializable input.
- The X1 input must come from another function block. You cannot set this value.
- The RT input must come from another function block, if the Mode is Cascade. If the Mode is Auto, you can set the value for RT or it can come from a user program.

Input ranges and limits

- You must specify an X1 engineering unit range, XEUHI and XEULO.
 - XEUHI and XEULO define the full range of X1 in engineering units. XEUHI represents the 100% of full scale value. XEULO represents the 0% of full scale value.
- This block assumes X1 is within XEUHI and XEULO – it applies no range check
- You must specify RTHILM and RTLOLM to define the ratio limits in engineering units. RT cannot exceed these limits. The maximum RTHILM value is 100.0 and the minimum RTLOLM value is 0.001, so the RT range must be between 0.001 and 100.0.
 - The operator is prevented from storing a RT 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 RT primary's windup status.

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 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 RATIONBIAS block to SP on a PID block, Control Builder automatically creates the BACKCALCOUT to BACKCALCIN connection.

- The RATIOBIAS 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 RATIOBIAS block

(RATIOBIAS.OP) to the set point of a PID block (PIDA.SP), the implicit/hidden connection is made to RATIOBIAS.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 RATIOBIAS 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, CVEUHI and CVEULO track the X1 input range (XEUHI and XEULO).



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. 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.

OPHILM and OPLOLM 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.

This block calculates CV using this equation:

- $CV = X1 * RT + OPBIAS.FIX + OPBIAS.FLOAT$

Control initialization

The RATIOBIAS 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 RATIOBIAS block:

- initializes its output:
 - CV = initialization value from the secondary
- calculates initialization values for the X1 and RT primaries:
 - $INITVAL[1] = (CV - OPBIAS.FIX) / RT$
 - $INITVAL[2] = (CV - OPBIAS.FIX) / INITVAL[1]$
(If the calculated $INITVAL[2]$ value exceeds either the high or low ratio limit (RTHILM or RTLOLM), it is clamped to the limit.)
- requests both primaries to initialize:
 - $INITREQ[1] = ON$
 - $INITREQ[2] = ON$

Where:

$OPBIAS.FIX$ = fixed output bias

$INITREQ[2]$ = initialization request flag for the RT primary

$INITVAL[2]$ = initialization value for the RT primary

$INITREQ[1]$ = initialization request flag for X1 primary

$INITVAL[1]$ = initialization value for X1 primary

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 RATIOBIAS 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.

$$\text{OPBIAS.FLOAT} = \text{CV}_{\text{INIT}} - (\text{CV}_{\text{UNBIASED}} + \text{OPBIAS.FIX})$$

Where:

CV_{INIT} = initialization value received from the secondary

$\text{CV}_{\text{UNBIASED}}$ = 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 $\text{CV} + \text{OPBIAS.FIX}$ should be the same as CV_{INIT} 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 ramp rate 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.

$$\text{OPBIAS.FLOAT} = \text{OPBIAS.FLOAT} - (\text{OPBIAS.RATE} / \text{cycles\_per\_Min})$$

Where:

$$\text{cycles\_per\_min} = \text{number of times the function block executes per minute (calculated)}$$

– 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.

Timeout monitoring

If mode is CASCADE, the RATIOBIAS block performs time-out monitoring on X1 and RT – if good X1 and RT values are not received within a predefined time (TMOUTTIME), the RATIOBIAS 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.

- If RT times out, the RATIOBIAS block does the following:
 - Holds RT at its last good value.
 - Changes the mode to a user-specified TMOUTMODE.
 - Requests the RT primary to initialize.
- If X1 times out, the RATIOBIAS block does the following:
 - Sets the X1 value to NaN. This causes CV to go to NaN, which initializes the RT and X1 primaries.

If RT times out and the block sheds to AUTO mode, it sets the Cascade Request Flag (CASREQFL). When CASREQFL is set, it means the block is waiting to return to the Cascade mode as soon as it gets a good RT value. You can disable the return to Cascade mode by manually clearing the CASREQFL or changing the mode.



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.

Override feedback processing

If the RATIOBIAS 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 RATIOBIAS block does the following:

- Computes a feedback value for the X1 and RT primaries:

$$\text{feedback value for X1 primary} = (\text{ORFBVAL} - \text{OPBIAS.FIX} - \text{OPBIAS.FLOAT}) / \text{RT}$$

$$\text{feedback value for RT primary} = (\text{ORFBVAL} - \text{OPBIAS.FIX} - \text{OPBIAS.FLOAT}) / \text{override feedback value for X1 primary}$$

Where:

ORFBVAL = override feedback value received from secondary

OPBIAS.FIX = fixed output bias

OPBIAS.FLOAT = floating output bias



ATTENTION

You can use SECINITOPT to ignore override requests from the secondary.

Windup handling

The RATIOBIAS block computes these three anti-reset windup status parameters.

- ARWOP
- ARWNET[1]
- ARWNET[2]

The ARWOP parameter indicates if OP is woundup. OP is woundup, if it is clamped or the secondary is in windup. ARWOP is computed as follows. (The secondary's windup status comes through BACKCALC.)

If OP is . . .	And Secondary's Windup = Normal; then, ARWOP = . . .	And Secondary's Windup = Lo; then, ARWOP = . . .	And Secondary's Windup = Hi; then, ARWOP = . . .
not clamped	NORMAL	Lo	Hi
clamped at its high limit	Hi	HiLo	Hi
clamped at is low limit	Lo	Lo	HiLo

The ARWNET[1] parameter indicates if X1 is woundup. This is a copy of the ARWOP, which means; if OP is woundup, then X1 is also woundup.

The ARWNET[2] parameter indicates if RT input is woundup. RT winds up, if it is clamped or OP is woundup. ARWNET[2] is computed as follows.

If RT is . . .	And ARWOP = Normal; then, ARWNET[2] = . . .	And ARWOP = Lo; then, ARWNET[2] = . . .	And ARWOP = Hi; then, ARWNET[2] = . . .
not clamped	NORMAL	Lo	Hi
clamped at its high limit	Hi	HiLo	Hi
clamped at is low limit	Lo	Lo	HiLo

Regulatory Control

RATIOBIAS Block

RATIOBIAS parameters



REFERENCE - INTERNAL

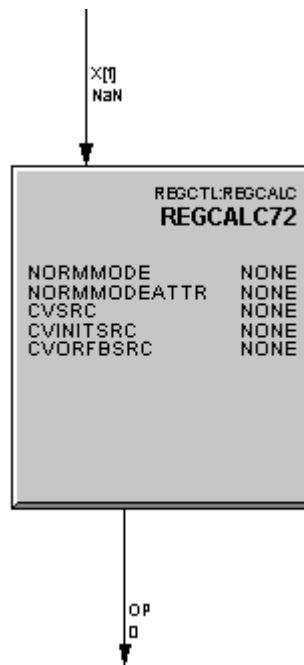
Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the RATIOBIAS block.

REGCALC (Regulatory Control Calculator) Block

Description

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.



Each REGCALC 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 0 to 65535. 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, NONE, and NORMAL. 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, PROGRAM, and NORMAL. 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

Configuration Tab	Description
	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).
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..6]). 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..6]). The default value is 0 (zero). • 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

Configuration Tab	Description
	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. • XK (XK[1..6]) – Lets you specify an individual gain value for each of the six X inputs. The default value is 1. • XB (XB[1..6]) – Lets you specify an individual bias value for each of the six X inputs. The default value is 0.00, which means no bias is added.
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

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 -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 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. 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 REGCALC 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,

Configuration Tab	Description
	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 REGCALC 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 REGCALC 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 Value (ORFBVALSRC) – Lets you assign an input or expression result as the source

Configuration Tab	Description
	of the override feedback value for the primary. If desired, you can leave this parameter unassigned. The default selection is NONE. The REGCALC 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. • Priority – Lets you set the desired priority level individually for each alarm type (SIALM.PR, BADCTLALM.PR, OPHIALM.PR, OPLOALM.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 (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. 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
SCM	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 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 (not applicable to this block). – CAS - Set MODEREQ = CASCADE.

Configuration Tab	Description
	– FIXEDOP - Set OPREQ = Configured Value. – 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 Rate (not applicable to 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 OPEXL0LM to OPEXHILM and within SPLOLM to SPHILM, for set point. The default value is NaN (Not a Number).
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.

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.

Operating modes and mode handling

The REGCALC block supports the Manual and Cascade modes of operation.

If Mode is . . .	Then,
Manual (MAN)	the output can be set by the operator or a user program. The X1 input is ignored.
Cascade (CAS)	the X1 input comes from another function block.

The initialization request occurs when the MODE changes from CAScade to MANual, but not from MANual to CAScade.

Inputs

The REGCALC block can function without any inputs. The following inputs are optional and they only accept real (Float 64) data types.

- X[1] - An initializable input that must come from another block, an operator can not set it.
- X[2] through X[6] general purpose inputs.
- XK[1..6] individually configurable gain value for each input.
- XB[1..6] individually configurable bias value for each input.
- XKB[1..6] individual inputs with gain and bias values applied to them.
- XWHIFL – An external windup high flag.
- XWLOFL – An external windup low flag.

Since X[1] is an initializable input, the block can have one primary. There is one primary for each initializable input.

Input ranges and limits

- If this block has a primary, you must specify an X[1] engineering unit range, XEUHI and XEULO. These only apply to initializable input.
 - XEUHI and XEULO define the full range of X1 in engineering units.
XEUHI represents the 100% of full scale value.
XEULO represents the 0% of full scale value.
- This block assumes X[1] is within XEUHI and XEULO – it applies no range check. If this function is required, you must write an expression for it.

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 REGCALC block to SP on a PID block, Control Builder automatically creates the BACKCALCOUT to BACKCALCIN connection.

- The REGCALC 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 REGCALC block (REGCALC.OP) to the set point of a PID block (PIDA.SP), the implicit/hidden connection is made to REGCALC.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 REGCALC 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.
- ORFBSTSRC – 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..6] – Inputs

XSTS[1..6] – Input Status

Output assignment rules



ATTENTION

The REGCALC 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 REGCALC 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 REGCALC 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.

Regulatory Control

REGCALC (Regulatory Control Calculator) Block

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 REGCALC 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 REGCALC 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 REGCALC runs, its CV increments by 10.0, producing seemingly wrong results.

You can configure a REGCALC block to work like an AUTOMAN block by:

- Connecting X[1] for input from the primary.
- Assigning CVSRC to X[1] input.
- Configuring all other parameters like OPBIAS.RATE the same as you would for an AUTOMAN block.

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 REGCALC 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.

$$\text{OPBIAS.FLOAT} = \text{CV}_{\text{INIT}} - (\text{CV}_{\text{UNBIASED}} + \text{OPBIAS.FIX})$$

Where:

- CV_{INIT} = initialization value received from the secondary
- $\text{CV}_{\text{UNBIASED}}$ = unbiased calculated value (based on input from the primary)
- OPBIAS.FIX = fixed bias (user-specified)

Regulatory Control

REGCALC (Regulatory Control Calculator) Block

- 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.

$$\text{OPBIAS.FLOAT} = \text{OPBIAS.FLOAT} - (\text{OPBIAS.RATE} / \text{cycles\_per\_Min})$$

Where:

$$\text{cycles\_per\_min} = \text{number of times the function block executes per minute (calculated)}$$

- 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 REGCALC block calculates the floating bias using the following equation.

$$\text{OPBIAS.FLOAT} = \text{CVINIT} - (\text{CV}_{\text{unbiased}} + \text{OPBIAS.FIX})$$

Where:

$\text{CV}_{\text{unbiased}}$ = 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 REGCALC block performs timeout monitoring on $X[1]$ – if good $X[1]$ value is not received within a predefined time (TMOUTTIME), the REGCALC 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.

- If $X[1]$ times out, the REGCALC block does the following:
 - Sets the input timeout flag (TMOUTFL).
 - Sets the input value to Bad (NaN).
 - Requests the $X[1]$ 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.

Regulatory Control

REGCALC (Regulatory Control Calculator) Block

Override feedback processing

If the REGCALC 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 REGCALC block does the following:

- initializes its output:

$$CV = CVORFB \text{ (an assignable output)}$$

- Computes a feedback value for its primary:

$$\text{feedback value for primary} = ORFBVAL \text{ (an assignable output)}$$

$$\text{feedback status for primary} = ORFBSTS \text{ (an assignable output)}$$

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:

$$\text{feedback value for primary} = \text{feedback value received from secondary.}$$



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 REGCALC 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 REGCALC 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 REGCALC 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 REGCALC 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 REGCALC 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.

Regulatory Control

REGCALC (Regulatory Control Calculator) Block

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 REGCALC 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.

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)
Assignment	:=
Parenthesis	()

Operators		Description	
Array Syntax		[]	
Unary Functions			
ABS	absolute value	LOG	natural Logarithm of a number
ATN	arc tangent	RND	round value
COS	cosine	SGN	sign of value (returns -1,0 or +1)
EXP	e to the power of x	SIN	sine
INT	convert to integer	SQR	square of a number
ISFIN	is finite	SQRT	square root
ISNAN	is Not a Number	TAN	tangent
LN	Log to the base of e		
Multiple Argument Functions			
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

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 except Time 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.REGCALC.X[1], CM1.REGCALC.X[2], CM1.REGCALC.X[3]))

REGCALC parameters



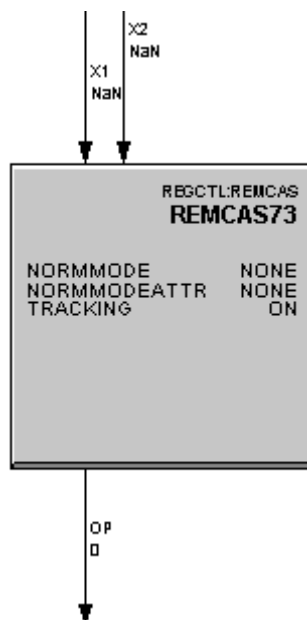
REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the REGCALC block.

REMCAS (Remote Cascade) Block

Description

The REMCAS block provides automatic switching between a remote cascade and a backup cascade. It is typically used with a PID that normally gets its set point from a remote source, but sheds to a local source if there is communication failure. It looks like this graphically:



If this block can communicate with both sources, it always selects the remote source. If it loses communications with the remote, it switches to the backup source; and when communications are restored, it automatically switches back to the remote.

You may force the unselected input to track the selected input through the TRACKING option:

- If TRACKING is On, this block continually initializes the unselected input. That is, on each cycle, it requests the unselected primary to initialize and set its output to the selected input value.
- If TRACKING is Off, this block does not initialize the unselected input.

This block provides bumpless switching by applying a floating bias to the output, regardless of whether TRACKING is On or Off.

Each REMCAS 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 0 to 65535. 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, NONE, and NORMAL. 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, PROGRAM, and NORMAL. 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

Configuration Tab	Description
	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). • 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).

Configuration Tab	Description
	• X1 (XDESC[1]) – X1 input descriptor of up to 15 characters long. • X2 XDESC[2]) – X2 input descriptor of up to 15 characters long. • Enable Tracking Option (TRACKING) – Lets you select if the unselected input is to track the selected input or not. The default selection is box checked, which means TRACKING is ON. – When TRACKING is ON, the block only propagates to the selected input. – When TRACKING is OFF, the block propagates changes in the windup status and override feedback data to all inputs. • 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. 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

Configuration Tab	Description
	default selection is NO_SHED.
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 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

Configuration Tab	Description
	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. - 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.
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

Configuration Tab	Description
	(BADCTL OPT). 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: – 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

Configuration Tab	Description
	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 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

Configuration Tab	Description
	– 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). – CAS - Set MODEREQ = CASCADE. – FIXEDOP - Set OPREQ = Configured Value. – 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 Rate (not applicable to 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).

Configuration Tab	Description
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.

Function

This block receives two input values (X1 and X2), as shown in Figure 31. X1 comes from the remote source and X2 comes from the backup or local source. The block performs timeout monitoring on both inputs, and the function block normally operates in the Cascade mode. Under normal conditions, this block passes input from the remote source to the output, without change. When the remote input times out, this block automatically switches to the backup source, and changes the mode to Backup Cascade. If both inputs timeout, this function block sets CV to NaN, which forces “Bad Control” processing.

It does not matter where the sources for X1 and X2 reside.

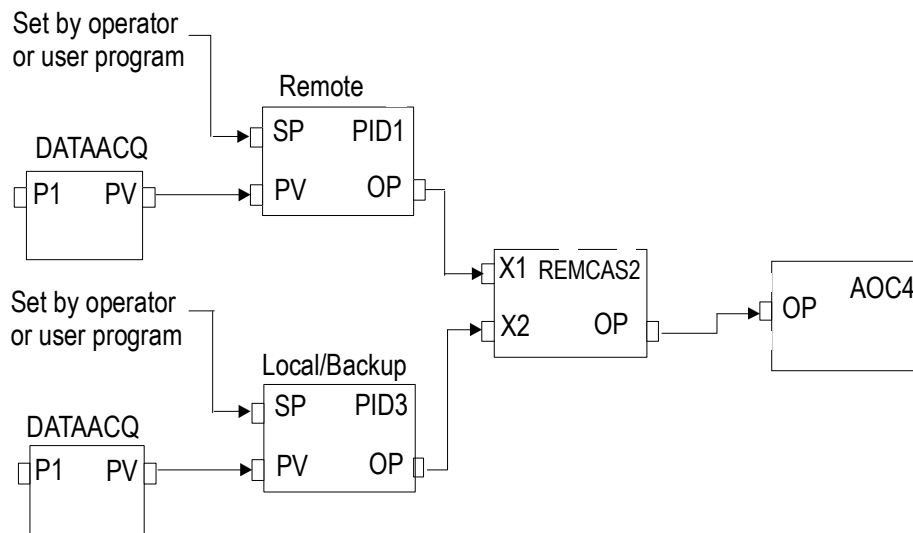
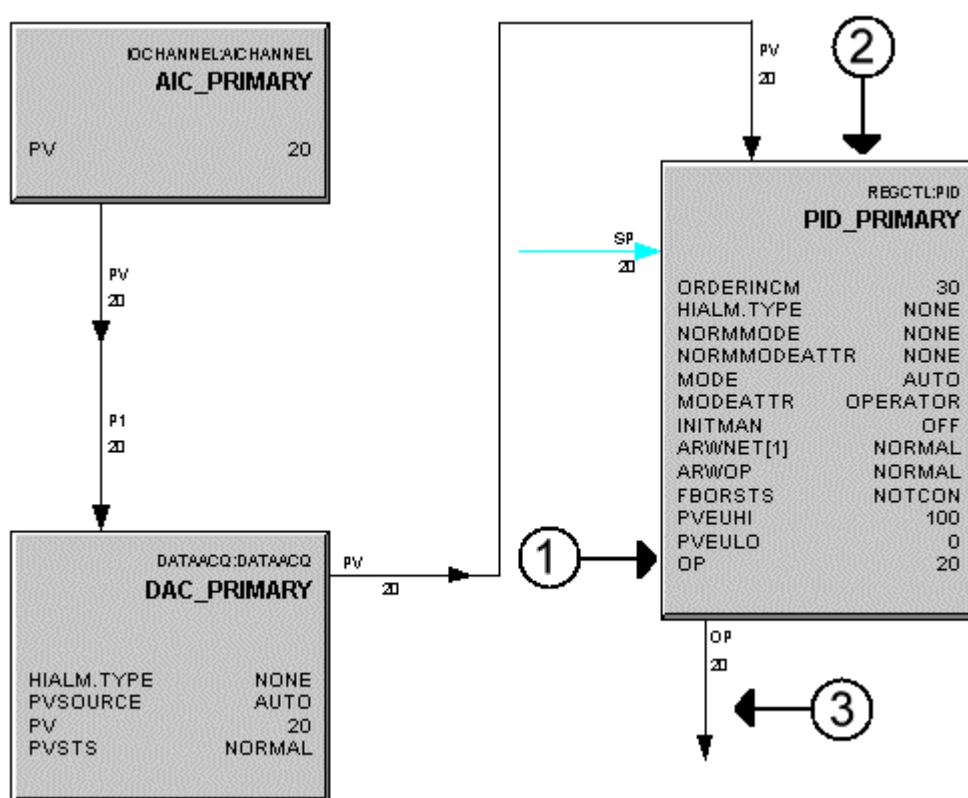


Figure 31 Functional block diagram of typical remote cascade operation.

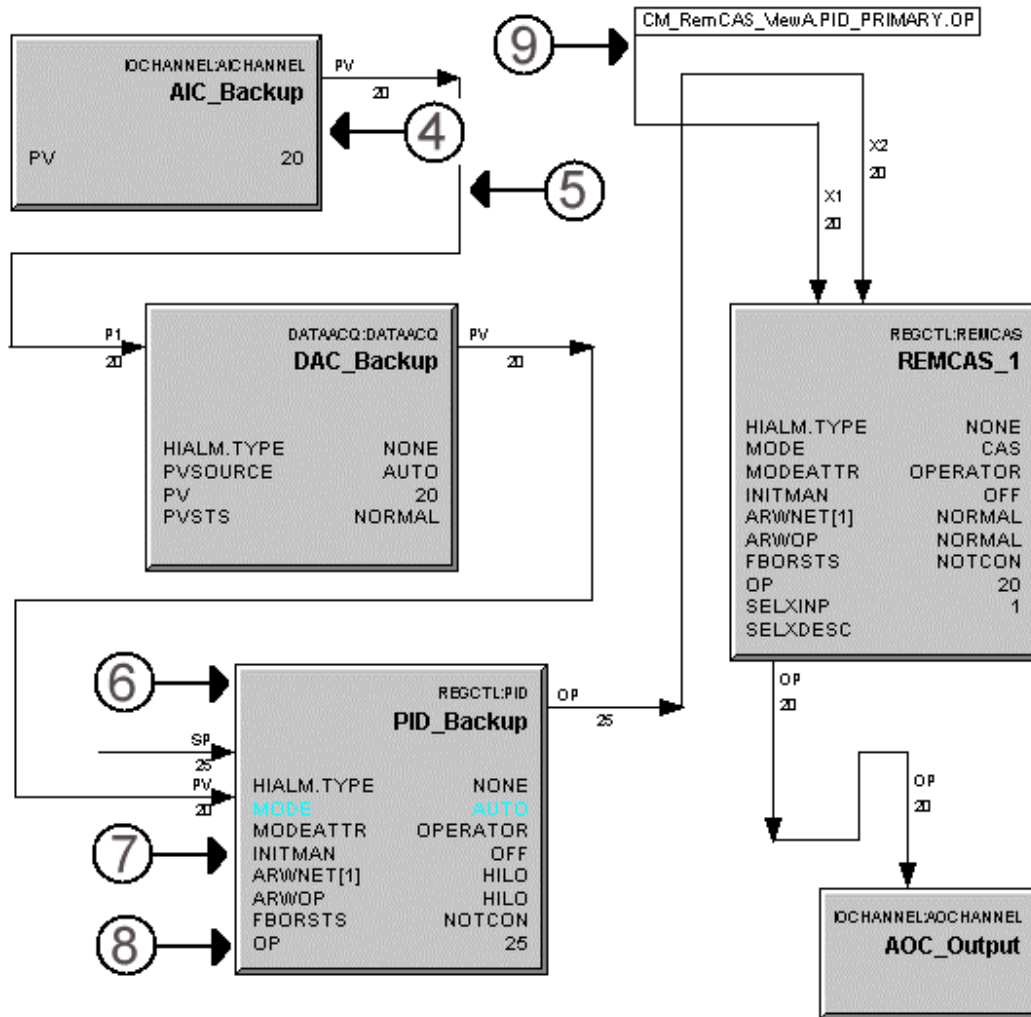
Configuration example

Figure 32 (Views A and B) and its companion callout description table show a sample configuration that uses a REMCAS block to form a cascade control loop with a backup primary loop for quick reference. The views in Figure 32 depict loaded configurations in Monitoring mode.



View A – Remote Primary Control Loop

Regulatory Control
REMCAS (Remote Cascade) Block



View B – Backup Primary Loop

Figure 32 Example of CB configuration using REMCAS block

The following table includes descriptions of the callouts in Figure 32, Views A and B.

Callout	Description
1	Control Builder connects the X1BACKCALOUT parameter of the REMCAS_1 block to the BACKCALCIN parameter for the PID_PRIMARY block in another Control Module (CM). The individual BACKCALCIN/BACKCALCOUT connections for each output used are automatically built by Control Builder as implicit/hidden connections.
2	This control loop represents a remote cascade primary located in a different CM. Typically, this loop is also located in a different controller. This loop serves as the remote primary for the REMCAS_1 block located in another CM. The REMCAS_1 block uses the output from this loop as its primary input (X1), as long as this loop provides a valid output value. If a communication or some other problem interrupts this loop's output, the REMCAS_1 block switches to the output from the backup/local primary loop.
3	Use the parameter connector function to connect the output (OP) from this loop to the input (X1) of the REMCAS_1 block in another Control Module (CM). See callout 9 in View B for the parameter connector to the X1 pin. 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.
4	Typically, the Analog Input Channel (AIC) block supplying the input for the backup/local primary loop (PID_BACKUP) is field wired to the same location as the AIC for the remote primary loop (PID_PRIMARY).
5	Use the PV parameter connection to carry data and status 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.
6	The PID_BACKUP block serves as a backup/local primary source for the remote primary source (PID_PRIMARY) located in another CM. The REMCAS_1 block switches to this backup source, if there is a problem with the output from the PID_PRIMARY.
7	The INITMAN function remains ON in the PID_BACKUP block.
8	With the Tracking option enabled, the output (OP) of the PID_BACKUP block always equals the value being sent by the PID_PRIMARY, while the PID_PRIMARY is being used for control.

Regulatory Control

REMCAS (Remote Cascade) Block

Callout	Description
9	Use the parameter connector function to connect the OP parameter of the PID_PRIMARY block in another Control Module (CM) to the X1 parameter on this REMCAS_1 block. See callout 3 in View A for the location of the OPEUX pin. 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.

Inputs

The REMCAS block requires two inputs – X1 and X2. X1 comes from the remote source and X2 is from the backup or local source.

- X1 and X2 are both initializable inputs.
- X1 and X2 must be pulled from other function blocks; they cannot be stored manually.
- This block has two primaries. (There is one primary per initializable input.)

Input ranges and limits

- You must specify an X-input engineering unit range, XEUHI and XEULO.
XEUHI and XEULO define the full range of the inputs; XEUHI is the value that represents 100% of full scale, and XEULO is the value that represents 0%.
- XEUHI and XEULO apply to both inputs (X1 and X2).
- This block assumes both inputs are within XEUHI and XEULO; it applies no range-checks.

Input descriptors

You can define a 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.

SELXDESC is automatically updated when SELXINP changes.

Outputs

The REMCAS 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 REMCAS block (REMCAS.OP) to the set point of a PID block (PIDA.SP), the implicit/hidden connection is made to REMCAS.OPX to provide value/status data.)

Output ranges and limits

- CVEUHI and CVEULO define the full range of CV, in engineering units.
If this block has a secondary, it brings the secondary's input range through BACKCALCIN and sets its CV range to that. If it has no secondary, CVEUHI and CVEULO track the X-input range (XEUHI and XEULO).
This block brings 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. 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.

Mode handling

This block supports the Cascade, Backup Cascade, and Manual modes:

- If the remote source (X1) is the currently selected input, the MODE is CAScade
- If the backup source (X2) is the currently selected input, the MODE is Backup CAScade
- If the MODE is MANual, an operator or user program may store OP. In this case, X1 and X2 are ignored.

Regarding mode-changes:

- This block requests both primaries to initialize after any mode-change except MANual to CAScade and CAScade to Backup CAScade.

Timeout monitoring

If the MODE is CAScade or Backup CAScade, this block performs timeout monitoring on both inputs (X1 and X2). If either input value is not updated within a predefined time, the block invokes timeout processing as outlined in the following paragraph.

Timeout processing

If the MODE is CAScade and an input times out, this block does the following :

- If X1 times out, but X2 is good, the block:
 - sets the “input timeout” flag (TMOUTFL)
 - sets the MODE to Backup Cascade
 - sets the currently selected input (SELXINP) to X2
 - requests the X1 primary to initialize
- If X2 times out, but X1 is good, the block:
 - requests the X2 primary to initialize

If X1 is good, then the MODE is CAScade and X1 is already the currently selected input.

- If both inputs timeout, the block:
 - sets CV to NaN, which forces a “Bad Control” condition. You specify what actions to take on Bad Control through the BADCTLOPT parameter.
 - sets the currently selected input (SELXINP) to None
 - requests both primaries to initialize

If X1 times out, and the block sheds to Backup Cascade, it sets the Cascade Request flag (CASREQFL). When CASREQFL is set, it means the block is waiting to return to the Cascade mode, and will do so as soon as it gets a good X1 value.

Processing notes on CASREQFL:

- This block only sets CASREQFL if the original mode was Cascade, the X1 input times out, and TMOUTMODE = Backup Cascade.
- You **cannot** set the CASREQFL. However, it can be cleared by setting the block's MODE to MANUAL.

If the MODE was Cascade and it changed due to timeout, the block does the following the next time it receives data from a primary:

- If SELXINP is X2, and X1 is good, (i.e., X1 just changed from bad to good) , the block:
 - sets SELXINP to X1
 - changes MODE to Cascade

Input switching

You may force the unselected input to track the selected input through the TRACKING option:

- If TRACKING is On, this block continually initializes the unselected input. That is, on each cycle, it requests the unselected primary to initialize and set its output to the selected input value.
- If TRACKING is Off, this block does not initialize the unselected input.

When TRACKING is Off, this block propagates changes in windup status and override feedback data to both inputs. When TRACKING is On, it only propagates to the selected input (because the unselected input is in the initialized state).

This block provides bumpless switching by applying a floating bias to the output, regardless of whether TRACKING is On or Off.

Equations

The REMCAS block computes CV as follows:

$$CV = X(n) + OPBIAS.FIX + OPBIAS.FLOAT$$

Where:

OPBIAS.FIX = fixed output bias

OPBIAS.FLOAT = floating bias, calculated using the following equation:

OPBIAS.FLOAT = (CV(from last cycle) – X(n)) – OPBIAS.FIX

X(n) = the currently-selected input (n = 1 or 2)

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 REMCAS 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 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. You can only set the OPBIAS.FIX value when the function block is Inactive or Mode equals Manual. The following occurs when you set the OPBIAS.FIX 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, OPBIAS and OPBIAS.FLOAT are recomputed.

- The OPBIAS.FLOAT is calculated as follows.

$$\text{OPBIAS.FLOAT} = \text{CV}_{\text{INIT}} - (\text{CV}_{\text{UNBIASED}} + \text{OPBIAS.FIX})$$

Where:

CV_{INIT} = initialization value received from the secondary

$\text{CV}_{\text{UNBIASED}}$ = 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 $\text{CV} + \text{OPBIAS.FIX}$ should be the same as CV_{INIT} 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.

$$\text{OPBIAS.FLOAT} = \text{OPBIAS.FLOAT} - (\text{OPBIAS.RATE} / \text{cycles\_per\_Min})$$

Where:

$$\text{cycles\_per\_min} = \text{number of times the function block executes per minute (calculated)}$$

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.

Control Initialization

This block brings initialization requests from its secondary through BACKCALCIN. In addition, the secondary may propagate one-shot initialization requests to this block.

You may use SECINITOPT to ignore initialization requests from the secondary.

If the secondary is requesting initialization, this block:

- initializes its output:

$$\text{CV} = \text{initialization value from the secondary}$$

- builds an initialization request for the X1 primary as follows:

$$\text{INITREQ}[1] = \text{On}$$

$$\text{INITVAL}[1] = \text{CV} - \text{OPBIAS.FIX}$$

- Parameters are defined below.
- builds an initialization request for the X2 primary:

INITREQ[2] = On
INITVAL[2] = CV - OPBIAS.FIX

Where:

INITREQ[1] = initialization request flag for the X1 primary
INITVAL[1] = initialization value for the X1 primary
INITREQ[2] = initialization request flag for the X2 primary
INITVAL[2] = initialization value for the X2 primary

Override feedback processing

If this block is in a cascade strategy with a downstream Override Selector block, it will receive 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 Selector block). The offset flag only applies to PID-type function blocks.

You may use SECINITOPT to ignore override requests from the secondary.

When the override status changes from selected to unselected, this block does the following:

- Computes a feedback value for the selected primary:

feedback value for selected primary = BACKCALOUT.ORFBVAL –
OPBIAS.FIX – OPBIAS.FLOAT

- The unselected primary is propagated with the “not connected” status.

REMCAS parameters



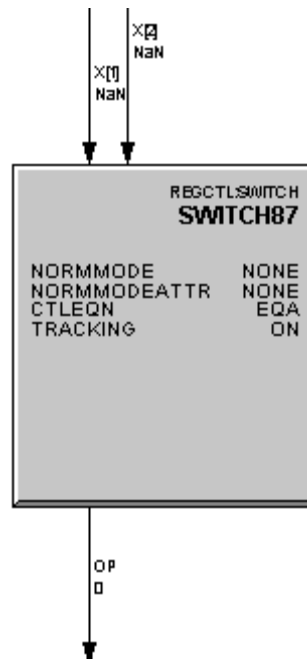
REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the REMCAS block.

SWITCH Block

Description

The SWITCH block accepts up to eight initializable inputs and operates as a single-pole, eight-position rotary switch. The switch position may be changed by the operator, a user program, or another function block. It looks like this graphically.



You may force the unselected inputs to track the selected input through the TRACKING option:

- If TRACKING is On, this block continually initializes the unselected inputs. That is, on each cycle, it requests the unselected primaries to initialize and set their output to the selected input value.
- If TRACKING is Off, this block does not initialize the unselected inputs.

This block provides bumpless switching by applying a floating bias to the output, regardless of whether TRACKING is On or Off.

Each SWITCH 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 0 to 65535. 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, B, or C to define how the block is to select an input. The default selection is EQA, which means you must store the number of the input to be selected to the SELXINP parameter. See the Function and Equation sections for this block for more details about the equations. • Enable Tracking Option (TRACKING) – Lets you select if the unselected input is to track the selected input or not. The default selection is box checked, which means TRACKING is ON. – When TRACKING is ON, the block continually initializes the unselected inputs. This means the block requests the unselected primaries to initialize and set their output to the selected input value. – When TRACKING is OFF, the block does not initialize the unselected inputs.

Configuration Tab	Description
	• 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). • 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, NONE, and NORMAL. 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, PROGRAM, and NORMAL. 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

Configuration Tab	Description
	(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 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 NOSLED.
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..8]). 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..8]). The default value is 0 (zero). - Time (sec) (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

Configuration Tab	Description
	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..8]). The description is stored in the XDESC[1..8] 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..8]) – 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).
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

Configuration Tab	Description
	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. • 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: – 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: – 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,

Configuration Tab	Description
	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 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

Configuration Tab	Description
	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 (not applicable to this block).. - CAS - Set MODEREQ = CASCADE. - FIXEDOP - Set OPREQ = Configured Value. - 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 Rate (not applicable to 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.

Function

This block lets you select one input from as many as eight, and outputs the selected value. It provides these three methods for selecting an input:

- Equation A. You store the number of the input to be selected to SELXINP.
- Equation B. You set one of the selection flags (SELXFL[1..8]) to On. Each flag corresponds to an input. The block turns all of the other flags Off and updates SELXINP.
- Equation C. You set or reset one of the selection flags (SELXFL[1..8]). The block does not change any of the other flags. Instead, it scans all flags in ascending order (1 to 8) and selects the first one that is On.

You can use this block to assign a different primary to a secondary. The example configuration shown in Figure 33 has five primary PID blocks connected to a SWITCH block. The active primary is selected by turning ON the corresponding SELXFL[1..5] input or storing the appropriate number to the SELXINP input, depending on the SWITCH block equation selected. The SELXINP parameter requires an integer data type and is usually set by an operator. The default SELXINP value is 1 and you cannot change it until the Control Module containing the SWITCH and primary blocks is activated at least once in Monitoring mode.

Please note that the configuration shown in Figure 33 is incomplete and is intended to only give you an idea of the general construction of a typical SWITCH block configuration.

Regulatory Control SWITCH Block

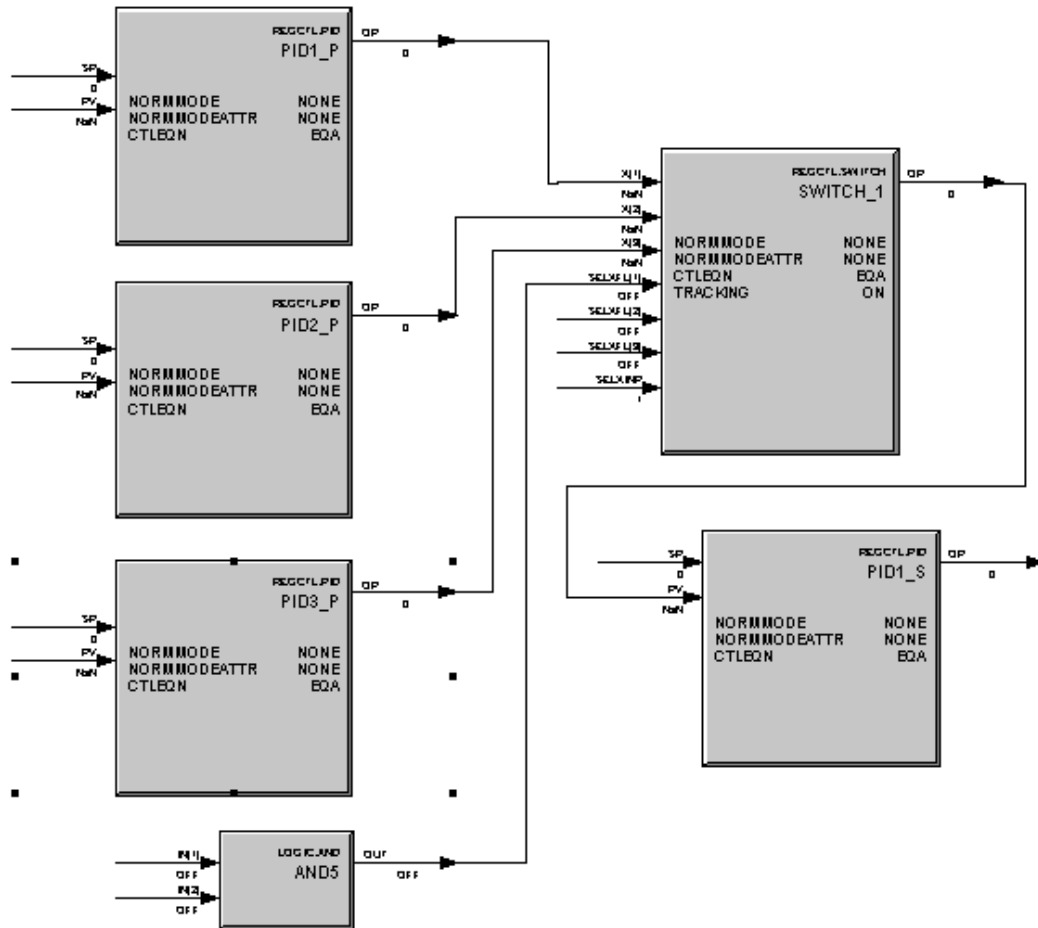


Figure 33 Example CB configuration using a SWITCH block to assign a different primary to a secondary.

You can also use multiple SWITCH blocks to assign a primary to a different secondary. The example configuration shown in Figure 34 uses a FANOUT block to provide the output from a primary PID block to two SWITCH blocks. One SWITCH block for each secondary. To select one of the secondaries, you must turn ON the same SELXFL input or store the same number to the SELXINP input on each SWITCH block.

Please note that the configuration shown in Figure 34 is incomplete and is intended to only give you an idea of the general construction of a typical multiple SWITCH blocks configuration.

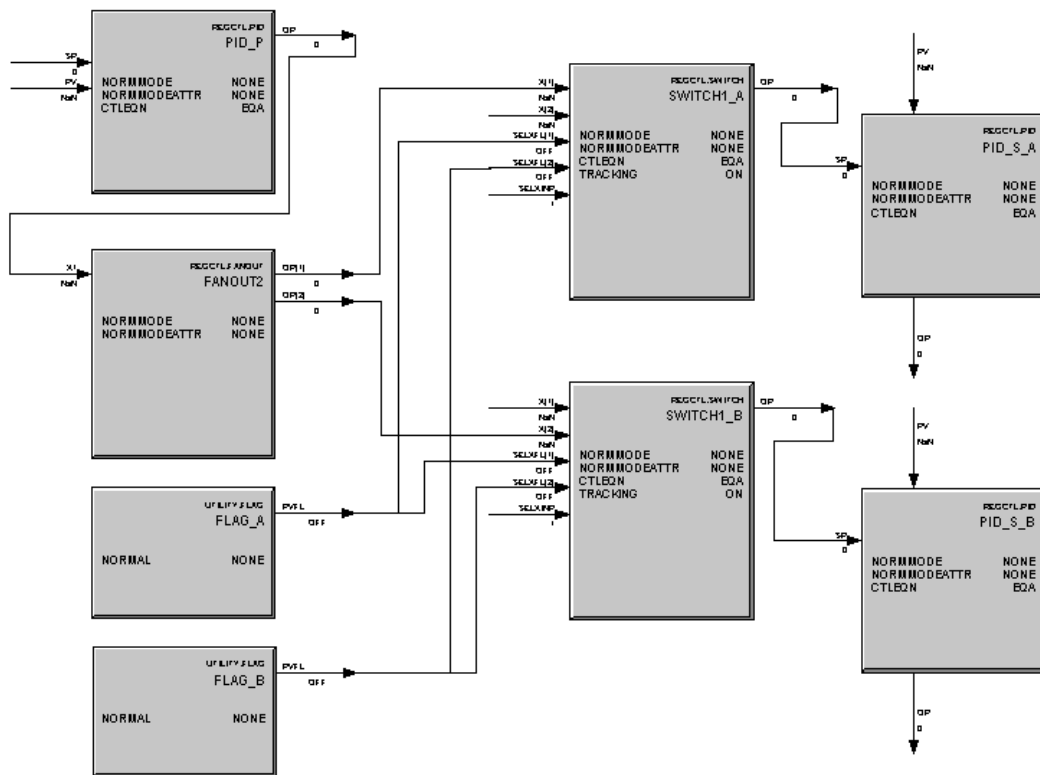


Figure 34 Example CB configuration using multiple SWITCH blocks to assign a primary to a different secondary.

Inputs

The SWITCH block accepts up to eight inputs – X[1] through X[8]. At least two inputs are required; the others are optional.

- X[1] through X[8] are initializable inputs.
- X[1] and X[2] are required; X[3] through X[8] are optional.
- The inputs must be pulled from other function blocks; you cannot store to them.
- This block may have two to eight primaries, depending on the number of inputs that are configured. (There is one primary per initializable input.)

Input ranges and limits

- You must specify an engineering unit range for the X inputs, by entering values for XEUHI and XEULO.
- XEUHI and XEULO define the full range of the inputs. XEUHI is the value that represents 100% of full scale, and XEULO is the value that represents 0%.
- XEUHI and XEULO apply to all of the X inputs.
- This block assumes all of the X inputs are within XEUHI and XEULO; it applies no range checks.

Input descriptors

This block lets you define a 15-character descriptor (name) for each X-input. The descriptors reside in the XDESC parameter, and when an input is selected, the corresponding descriptor is copied to SELXDESC.

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 SWITCH block to SP on a PID 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 SWITCH block (SWITCH.OP) to the set point of a PID block (PIDA.SP), the implicit/hidden connection is made to SWITCH.OPX to provide value/status data.)

Output ranges and limits

- CVEUHI and CVEULO define the full range of CV, in engineering units.
If this block has a secondary, it brings the secondary's input range through BACKCALCIN and sets its CV range to that. If it has no secondary, CVEUHI and CVEULO track the X-input range (XEUHI and XEULO).
This block brings 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. 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.

Mode handling

This block supports the Cascade and Manual modes:

- If MODE = Cascade, all inputs are pulled from other function blocks.
- If MODE = Manual, OP may stored by the operator or user program; inputs are ignored.

Regarding mode-changes:

- This block requests all primaries to initialize when mode changes from CASCade to MANual.

Timeout monitoring

If MODE is Cascade, this block performs timeout monitoring on all inputs (X[1..8]). If an input value is not updated within a predefined time, the block invokes timeout processing as described in the next section.

Timeout processing

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



ATTENTION

This block does not support mode shedding on timeout.

Equations

The SWITCH block supports three methods for selecting an input – Equation A, B or C. You configure this method through the parameter CTLEQN:

- Equation A: You select an input by storing to SELXINP. (SELXINP identifies the input to be selected.) When SELXINP is updated, equation A:
 - updates all selection flags (SELXFL[1..8]) accordingly. That is, it sets the flag for the selected input to On, and turns all others Off.
 - copies the selected input's descriptor to SELXDESC.
 - calculates CV as follows:

$$CV = X(n) + OPBIAS.FIX + OPBIAS.FLOAT$$

Where:

OPBIAS.FIX = fixed output bias

OPBIAS.FLOAT = floating bias

X(n) = currently selected input (n = 1 to 8)

Equation A prevents you from storing to the selection flags (SELXFL[1..8]).

- Equation B: You select an input by setting one of the selection flags (SELXFL[1..8]) to On. When this occurs, equation B turns all of the other flags Off. Following a store to any selection flag, equation B:
 - turns all other selection flags Off,
 - updates SELXINP and SELXDESC, and
 - calculates CV as noted above for Equation A.

Equation B prevents you from storing to SELXINP.

- Equation C: You can set one selection flag On without causing the others to be turned Off. You may store On or Off to any flag and the others are not affected.
Following a store to any selection flag, equation C:

- scans all flags in ascending order – from SELXFL[1] to SELXFL[8],
- selects the first input whose flag is On depending on following conditions:

if BADINPTOPT(i) = IgnoreBad, a “bad” input is not selected, it will be ignored.

if BADINPTOPT(i) = IncludeBad, a “bad” input may be selected.

- updates SELXINP and SELXDESC.
- calculates CV as noted above for Equation A.

The input selection is changed by storing On or Off to the selection flags as follows:

SELX FL[1]	SELX FL[2]	SELX FL[3]	SELX FL[4]	SELX FL[5]	SELX FL[6]	SELX FL[7]	SELX FL[8]	Given Selection Flag States Select Input...
On	NA	NA	NA	NA	NA	NA	NA	X[1]
Off	On	NA	NA	NA	NA	NA	NA	X[2]
Off	Off	On	NA	NA	NA	NA	NA	X[3]
Off	Off	Off	On	NA	NA	NA	NA	X[4]
Off	Off	Off	Off	On	NA	NA	NA	X[5]
Off	Off	Off	Off	Off	On	NA	NA	X[6]
Off	Off	Off	Off	Off	Off	On	NA	X[7]
Off	Off	Off	Off	Off	Off	Off	On	X[8]

“NA” means On or Off does not affect the input selection

Equation C prevents you from storing to SELXINP.

Bad input handling

The BADINPTOPT[1..8] parameter specifies the Bad Input handling option (IncludeBad or IgnoreBad) on a per input basis. The block uses BADINPTOPT[1..8] in a consistent manner, regardless of the configured equation - A, B, or C.

If the selected input “i” goes Bad (either the value being fetched is NaN, or the Switch sets the value to NaN because of a timeout), then the Switch does the following, based on the value of BADINPTOPT(i):

- BADINPTOPT(i) = IncludeBad:
CV is set to NaN,
the selected input does not change (no automatic switching).
- BADINPTOPT(i) = IgnoreBad
An attempt is made to automatically switch to the next input. If a good input is found, then the Switch selection changes to this input; if a good input is not found, then CV is set to NaN and the selected input does not change.

Based on the configured equation, the SWITCH block automatically switches to the next input as follows:

- Equations A and B: The next input is the next highest-input according to input number. For example, the next input for input # 1, X[1], is input #2, X[2]; the next input for X[2] is X[3], and so on; the next input for the last used input of the block is X[1] - if 5 inputs are used with the Switch, then the next input for X[5] is X[1].
- Equation C: The Switch block will only automatically switch to an input whose SELXFL(i) is On. The same “next” order is used as with Equations A and B.

Bypass processing

You may explicitly ignore bad inputs when Equation C is selected as the control equation. The following parameter supports this:

BADINPTOPT - Bad Input Option

Indicates if the block should include bad inputs (NaN) in the selection process.

BADINPTOPT has the following options:

- IgnoreBad (Ignore bad inputs)
- IncludeBad (Include bad inputs)

For this block, a bad input will cause CV to go bad. This means Bad Control.

Input switching

You may force the unselected inputs to track the selected input through the TRACKING option:

- If TRACKING is On, this block continually initializes the unselected input. That is, on each cycle, it requests the unselected primary to initialize and set its output to the selected input value.
- If TRACKING is Off, this block does not initialize the unselected input.

When TRACKING is Off, this block propagates changes in windup status and override feedback data to all inputs. When TRACKING is On, it only propagates to the selected input (because the unselected input is in the initialized state).

This block provides bumpless switching by applying a floating bias to the output, regardless of whether TRACKING is On or Off.

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 SWITCH 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 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.

$$\text{OPBIAS.FLOAT} = \text{CV}_{\text{INIT}} - (\text{CV}_{\text{UNBIASED}} + \text{OPBIAS.FIX})$$

Where:

CV_{INIT} = initialization value received from the secondary

$\text{CV}_{\text{UNBIASED}}$ = 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 $\text{CV} + \text{OPBIAS.FIX}$ should be the same as CV_{INIT} 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 ramp rate 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.

$$\text{OPBIAS.FLOAT} = \text{OPBIAS.FLOAT} - (\text{OPBIAS.RATE} / \text{cycles\_per\_Min})$$

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.

Error handling

If a selected input is bad, this block sets the CV to Bad (NaN), and leaves the Mode unchanged.

When the selected input is again good, this block recalculates CV, and requests the primary to initialize.

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 may use SECINITOPT to ignore initialization requests from the secondary.

If the secondary is requesting initialization, this block:

- initializes its output:

$CV = \text{initialization value from the secondary}$

- builds an initialization request for the selected primary as follows:

$INITREQ(s) = \text{On}$

$INITVAL(s) = CV - OPBIAS.FIX$

where:

(s) = identifies the selected input

$INITREQ(s) = \text{initialization request flag for the selected input}$

$INITVAL(s) = \text{initialization value for the selected input}$

- If TRACKING is On, this block also builds an initialization request for the unselected primaries as follows:

$INITREQ(n) = \text{On}$

$INITVAL(n) = CV - OPBIAS.FIX$

where:

(n) = identifies the unselected inputs

$INITREQ(n) = \text{initialization request flag for the unselected inputs}$

$INITVAL(n) = \text{initialization value for the unselected inputs}$

Override feedback processing

If this block is in a cascade strategy with a downstream Override Selector block, it will receive 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.

You may use SECINITOPT to ignore override requests from the secondary.

When the override status changes from selected to unselected, this block does the following:

- Computes a feedback value for the selected primary:

$$\text{feedback value for selected primary} = \text{BACKCALCOUT.ORFBVAL} - \text{OPBIAS.FIX} - \text{OPBIAS.FLOAT}$$

- Propagates the unselected primaries with “not connected” status.

If this block and a primary are on the same node, this block propagates the override data to the primary. If a primary is on a different node, this block stores the data in the BACKCALC packet for that primary, which the primary brings on its next execution.

SWITCH parameters



REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the SWITCH block.

UCN Interface

Universal Control Network (UCN) Interface Block

Functional overview

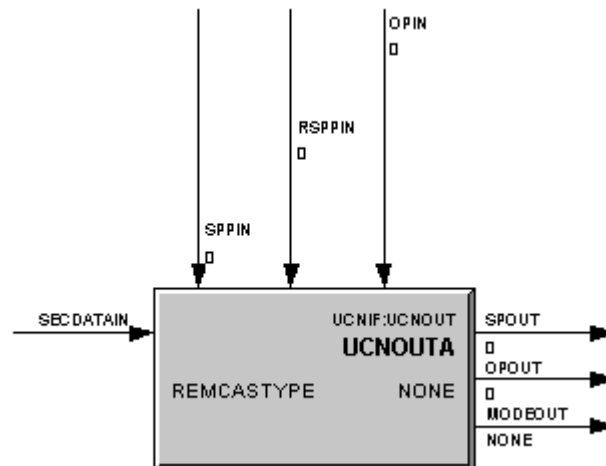
The UCN Interface block provides a configurable function for creating regulatory control cascade strategies between the Application Control Environment (ACE) supervisory controller and Process Manager controllers residing on a Universal Control Network in a connected TPS system.

The only UCN Interface block is the UCNOUT function block that provides configurable connections and compatible data mapping between controllers. The following subsection provides a functional description of the UCNOUT block

UCNOUT

Description

The UCNOUT function block supports Setpoint Control (SPC), Direct Digital Control (DDC), Remote Setpoint Control (RSP) and Direct Digital Control with Remote Setpoint (DDCRSP) remote cascade types between the regulatory control function blocks included in an ACE supervisory controller control strategy and the regulatory control points included in a Process Manager controller. It looks like this graphically in a Control Module:



The following table lists the major functions the UCNOUT block performs along with a brief description of the function.

Function	Description
Translates secondary data (SECDATA) from Process Manager regulatory control points to ACE controller compatible back calculation (BACKCALC) data.	The data structures for SECDATA and BACKCALC pass information back up the control path from secondaries to primaries. They contain data like initialization request, initialization value, and anti-reset windup status. Since SECDATA does not provide override feedback propagation data, the UCNOUT block cannot use BACKCALC to forward this data to its primary. This means override strategies are not possible between the ACE supervisory controller and the UCN based Process Manager controller.
Participates in Remote Cascade Request protocol for Process Manager regulatory control point MODE changes.	If Process Manager's regulatory control point is configured for Remote Cascade and the MODE is changed to Cascade, the MODE does not change immediately. The UCNOUT block receives the Remote Cascade request and then stores Cascade to the MODE for the Process Manager point to complete the formation of the cascade strategy.
Forwards inputs from primary regulatory control blocks in ACE supervisory controller to Process Manager regulatory control point.	The Process Manager point uses the Engineering Units obtained from the SECDATA fetch to convert stores to its regulatory control point setpoint to Engineering Units. The BACKCALC structure supports the same Engineering Units information function for ACE regulatory control blocks.

About remote cascade

The following table provides an overview of the four types of remote cascades that the UCNOUT block supports. Please refer to applicable Knowledge Builder documents for more information about Experion PKS system and TPS system regulatory control functions.

Remote Cascade Type	Description
SPC - Setpoint Control	Used for Supervisory to Level 1 controller cascade - UCNOUT block writes to the setpoint (SP) of the Process Manager regulatory control point, when the point is in Cascade (CAS) mode.
DDC - Direct Digital Control	Used for Supervisory store to Output directly - UCNOUT block writes to the output (OP) of the Process Manager regulatory control point, when the point is in Cascade (CAS) mode.
RSP - Remote Setpoint Control	Used for Supervisory to Level 1 controller cascade with a backup primary also existing in Level 1 - UCNOUT block writes to the setpoint (SP) of the local backup Process Manager regulatory control point, when the point is in Automatic Mode and being initialized by its secondary, which is in either SPC or DDC control by ACE supervisory controller.
DDCRSP - Direct Digital Control with Remote Setpoint	Used for Supervisory store to Output directly - UCNOUT block writes to output (OP) of the Process Manager regulatory control point, when the point is in Cascade (CAS) mode. UCNOUT block also writes to the setpoint (SP) of the same Process Manager regulatory control point to supply a backup SP.

Configuration form overview

The following table identifies the tabs and parameters associated with each one for quick reference.

Configuration Tab	Description
Main	- Name – Block (Tag) name of up to 16 characters long. Must be unique within the CM block containing it. - 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 0 to 65535. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. - Remote Cascade Type (REMCASTYPE) – Lets you select the type of remote cascade function the UCNOUT block is to support. See the previous section About remote cascade for more 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.

Input/Output

The remote cascade type (REMCATYPE) selection determines which UCNOUT block inputs to use as summarized in the following table.



ATTENTION

It is possible to configure and load a Control Module that includes an UCNOUT block with connections that are not consistent for the selected Remote Cascade Type. In this case, you will not be permitted to activate the Control Module in the Monitoring mode.

If a warning prompt appears about inconsistent UCNOUT input and output connections when saving a Control Module, be sure you have configured the correct inputs and outputs for the selected Remote Cascade Type before you close the Control Module.

If Remote Cascade Type Is . . .	Then, Connect SP Input (SPPIN) . . .	Or, Connect RSP Input (RSPPIN) . . .	Or, Connect OP Input (OPIN). . .
SPC - Setpoint Control	Yes	No	No
DDC - Direct Digital Control	No	No	Yes
RSP - Remote Setpoint Control	No	Yes	No
DDCRSP - Direct Digital Control with Remote Setpoint	Yes	No	Yes

The secondary data input (SECDAIN) and the mode output (MODEOUT) connections are required for all Remote Cascade Types. The following table summarizes the UCNOUT block inputs and outputs needed for the given Remote Cascade Type selection.

If Remote Cascade Type Is . . .	Then, Connect SECDAIN . . .	And/or, Connect SPOUT . . .	And/or, Connect OPOUT . . .	And/or, Connect MODEOUT . . .
SPC - Setpoint Control	Yes	Yes	No	Yes

UCN Interface

UCNOUT

If Remote Cascade Type Is . . .	Then, Connect SECDATAIN . . .	And/or, Connect SPOUT . . .	And/or, Connect OPOUT . . .	And/or, Connect MODEOUT . . .
DDC - Direct Digital Control	Yes	No	Yes	Yes
RSP - Remote Setpoint Control	Yes	Yes	No	Yes
DDCRSP - Direct Digital Control with Remote Setpoint	Yes	Yes	Yes	Yes

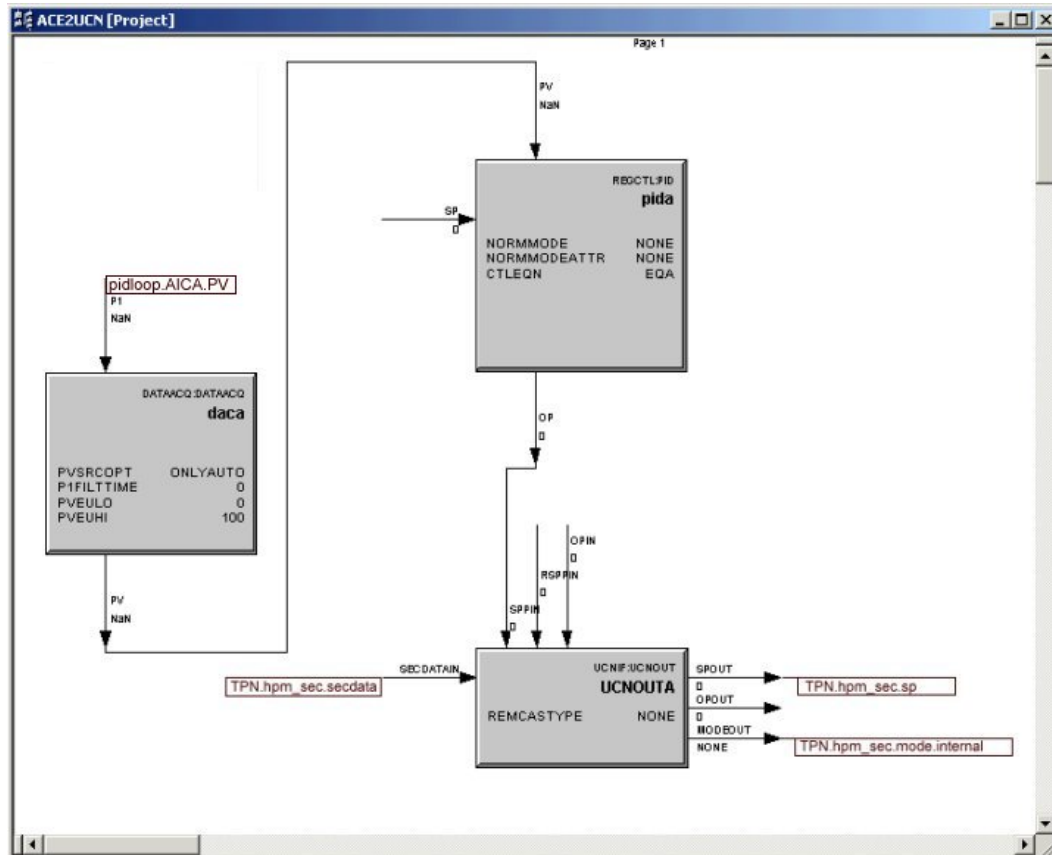


TIP

You must configure the MODE connection to the MODE.INTERNAL parameter to store the enumeration ordinal value instead of the enumeration member name. A reference to just MODE returns (on Get) or expects (on Store) the enumeration member name string. The TPN Server in the TPS system APP node provides this capability to access enumeration values as either ordinals or strings.

Configuration example

The following figure shows a configured Control Module assigned to the ACE supervisory controller with cascade connections to the UCN. The connections shown are for a Remote Cascade Type selection of SPC (Setpoint Control). Please note that the system automatically creates a BACKCALC connection between the UCNOUTA block and the primary (pida) based on the forward connection from pida.OP to UCNOUTA.SPPIN.



The SPPIN, RSPPIN, and/or OPIN input to the UCNOUT block is updated as part of each execution of the Control Module. The SECDATAIN input is gathered at the rate specified by periodic update rate configured for the referenced OPC server peer environment through the CEEACE block configuration of the peer subscription period. The SPOUT and/or OPOUT outputs are stored at the rate of the UCNOUT block execution, while the MODEOUT output is stored whenever necessary (normally a one-time store) to form the cascade with the UCN point.

Exchange Functions

Exchange Function Blocks

Functional overview

Exchange function blocks provide a variety of configurable functions for storing and retrieving selected control data.

The following table presents the various functions that can be performed through the configuration of the associated Exchange function blocks. Functional descriptions for each block are given in the following subsections.

Function	Block	Description
Initiate read/write of multiple two-state values	REQFLAGARRAY Block	Used to define up to 512 two separate states (Off/On) to indicate status of a particular input. CPM initiates read/write transactions to other connected devices using either the PCCC or CIP communications protocol.
Initiate read/write of multiple floating point values	REQNUMARRAY Block	Used to store up to 64 floating point values for use in a control strategy. CPM initiates read/write transactions to other connected devices using either the PCCC or CIP communications protocol.
Initiate read/write of multiple text strings	REQTEXTARRAY Block	Used to store up to 64 ASCII characters for use in a control strategy. CPM initiates read/write transactions to other connected devices using either the PCCC or CIP communications protocol.
Respond to read/write of multiple two-state values	RSPFLAGARRAY Block	Used to define up to 512 two separate states (Off/On) to indicate status of a particular input. CPM responds to read/write transactions from other connected devices using either the PCCC or CIP communications protocol.
Respond to read/write of multiple floating point values	RSPNUMARRAY Block	Used to define up to 64 floating point values for use in a control strategy. CPM responds to read/write transactions from other connected devices using either the PCCC or CIP communications protocol.

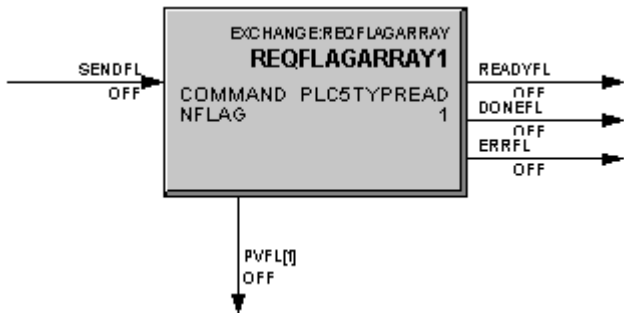
Exchange Functions
Exchange Function Blocks

Function	Block	Description
Respond to read/write of multiple text strings	RSPTXTARRAY Block	Used to define up to 64 ASCII characters for use in a control strategy. CPM responds to read/write transactions from other connected devices using either the PCCC or CIP communications protocol.

REQFLAGARRAY Block

Description

The REQFLAGARRAY function block provides storage for up to 512 2-state values. The value can be accessed as a simple Boolean (Off or On) using the PVFL[n] parameter. Where “n” is the number of the flag. It looks like this graphically:



Configuration Tab	Description
Main	- Name – Block (Tag) name of up to 16 characters long. Must be unique within the CM block containing it. - 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 0 to 65535. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. - Command (COMMAND) – Lets you set the block's current MODE. The selections are PLC5TYPREAD, PLC5TYPWRITE, CIPREAD, and CIPWRITE. (Note that the corresponding communications protocol for a PLC5TYPREAD or PLCTYPWRITE selection is PCCC (Programmable Controller Communications Commands) and the CIP (Control and Information Protocol) communications protocol is used for a CIPREAD or CIPWRITE selection.) - Number of Flag Values (NFLAG) – Lets you set the amount of Flags you want to control (1 .. 512).

Configuration Tab	Description
Communications	- Path to Device (PATH) - The relative path from the CPM to the target device (or the DHRIO module when the target device is on DH+) is specified as a comma-separated list of path segments. Each path segment is specified as an X, Y pair where X can either be 1 (indicating that this is an ICP Backplane Segment) or 2 (indicating that this is a ControlNet Segment). When X is 1, Y denotes a slot number and is in the range 0 to 16. When X is 2, Y denotes a ControlNet MAC ID and is in the range 1 to 99. The first path segment must be an ICP Backplane Segment. The path segment types, MAC IDs and slot numbers must be specified in decimal values. Binary, octal or hexadecimal values are not supported. IP Addresses are not supported. The following are examples of valid paths: e) 1, 5 (Path from the CPM to a device in slot 5 of the same chassis) f) 1, 0, 2, 3 (Path from the CPM through the CNI in slot 0 to a device that is on ControlNet with MAC ID 3) g) 1, 5, 2, 7, 1, 3 (Path from the CPM through the CNI in slot 5 to a CNI with MAC ID 7 to a device in slot 3) When the target device is on the DH+ network, the path is the relative path from the CPM to the DHRIO module. Additional values are used to specify the address of the target device on the DH+ network. See the following DHFL parameter for details. - File Name in Target Device (FILENAME) –The File Name in the target device is specified as a Logical ASCII Symbolic address without the Logical ASCII Identifier, when COMMAND is PLC5TYPREAD or PLC5TYPWRITE. For example, a valid PCCC File Name and Offset in a target device value might be N7:0. When COMMAND is CIPREAD or CIPWRITE, this is then the name of the file in the target device. - Use DH+ through DHRIO? (DHFL) – Lets you specify whether or not the target device is on an Allen-Bradley Data Highway and is connected through a DHRIO

Configuration Tab	Description
	module. The default selection is UNCHECKED (target device is not on DH+). The following parameters are only active when the DHFL parameter is selected (CHECKED). • DH+ Channel A/B (DHCHANNEL) – Lets you specify which DH+ channel on the DHRIO to use for this connection. The default selection is A for Channel A. • DH+ Source Link (DHSRCLINK) – Lets you specify the source link for the DH+ route when using DH+ Remote Messaging. This parameter should be set to zero when using DH+ Local Messaging. The default selection is 0. • DH+ Destination Link (DHDESTLINK) – Lets you specify the destination link for the DH+ route when using DH+ Remote Messaging. This parameter should be set to zero when using DH+ Local Messaging. The default selection is 0. • Target Node Address (octal) (DHNODE) – Lets you specify the node number (0 to 77 octal) of the target node on the DH+ network. The default value is 0 (octal).
Status/Data	• Process Value, from device (PVFL) - The Status/Data Tab in Configuration Form contains the initial values of the configured number of flag values. When this is a "read" block, we cannot write to these values (they will be obtained from the target device when the block starts executing and a send is triggered).
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.

Function

Used to define two separate states (Off/On) to indicate status of a particular input.

- Number of flag values (NFLAG) is user configurable.
- Current state of flags can be changed/read using flag value (PVFL[n] (Boolean)).

Input/Output

The block has up to 512 output flags (PVFL[n]). But, all block pin parameters are available to be exposed and connected to using Control Builder graphical connections.

REQFLAGARRAY parameters



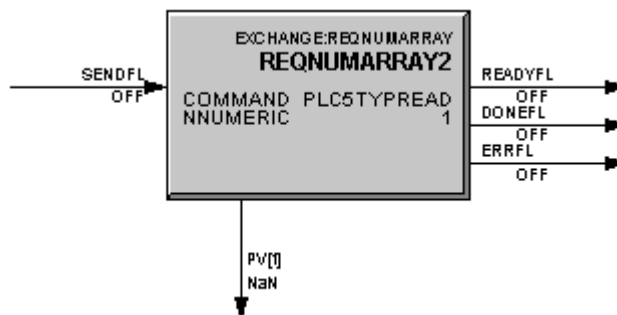
REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the REQFLAGARRAY function block.

REQNUMARRAY Block

Description

The REQNUMARRAY block provides storage for up to 64 floating point values which are accessible through the corresponding PV configuration parameter (PV[n]). Where “n” is the number of the numeric. It looks like this graphically:



Configuration Tab	Description
Main	- Name – Block (Tag) name of up to 16 characters long. Must be unique within the CM block containing it. - 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 0 to 65535. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. - Command (COMMAND) – Lets you set the block's current MODE. The selections are PLC5TYPREAD, PLC5TYPWRITE, CIPREAD, and CIPWRITE. (Note that the corresponding communications protocol for a PLC5TYPREAD or PLCTYPWRITE selection is PCCC (Programmable Controller Communications Commands) and the CIP (Control and Information Protocol) communications protocol is used for a CIPREAD or CIPWRITE selection.)

Configuration Tab	Description
	- Number of Numeric Values (NNUMERIC) – Lets you set the amount of floating point integers you want to control (1 .. 64). - Data Type in Target device (TGTDATATYPE) - Lets you specify the type of data in the target device. The selections are FLOAT32, SIGNEDINT8, SIGNEDINT16 AND SIGNEDINT 32.
Communications	- Path to Device (PATH) - The relative path from the CPM to the target device (or the DHRIO module when the target device is on DH+) is specified as a comma-separated list of path segments. Each path segment is specified as an X, Y pair where X can either be 1 (indicating that this is an ICP Backplane Segment) or 2 (indicating that this is a ControlNet Segment). When X is 1, Y denotes a slot number and is in the range 0 to 16. When X is 2, Y denotes a ControlNet MAC ID and is in the range 1 to 99. The first path segment must be an ICP Backplane Segment. The path segment types, MAC IDs and slot numbers must be specified in decimal values. Binary, octal or hexadecimal values are not supported. IP Addresses are not supported. The following are examples of valid paths: a) 1, 5 (Path from the CPM to a device in slot 5 of the same chassis) b) 1, 0, 2, 3 (Path from the CPM through the CNI in slot 0 to a device that is on ControlNet with MAC ID 3) c) 1, 5, 2, 7, 1, 3 (Path from the CPM through the CNI in slot 5 to a CNI with MAC ID 7 to a device in slot 3) When the target device is on the DH+ network, the path is the relative path from the CPM to the DHRIO module. Additional values are used to specify the address of the target device on the DH+ network. See the following DHFL parameter for details. - File Name in Target Device (FILENAME) –The File Name in the target device is specified as a Logical ASCII Symbolic address without the Logical ASCII Identifier, when COMMAND is PLC5TYPREAD or

Configuration Tab	Description
	PLC5TYPWRITE. For example, a valid PCCC File Name and Offset in a target device value might be N7:0. When COMMAND is CIPREAD or CIPWRITE, this is then the name of the file in the target device. • Use DH+ through DHRIO? (DHFL) – Lets you specify whether or not the target device is on an Allen-Bradley Data Highway and is connected through a DHRIO module. The default selection is UNCHECKED (target device is not on DH+). The following parameters are only active when the DHFL parameter is selected (CHECKED). • DH+ Channel A/B (DHCHANNEL) – Lets you specify which DH+ channel on the DHRIO to use for this connection. The default selection is A for Channel A. • DH+ Source Link (DHSRCLINK) – Lets you specify the source link for the DH+ route when using DH+ Remote Messaging. This parameter should be set to zero when using DH+ Local Messaging. The default selection is 0. • DH+ Destination Link (DHDESTLINK) – Lets you specify the destination link for the DH+ route when using DH+ Remote Messaging. This parameter should be set to zero when using DH+ Local Messaging. The default selection is 0. • Target Node Address (octal) (DHNODE) – Lets you specify the node number (0 to 77 octal) of the target node on the DH+ network. The default value is 0 (octal).
Status/Data	• Process Value, from device (PV) - The Status/Data tab contains the initial data values to be written to the target. Control strategies in the CPM can modify these values at run-time. The operator at run-time can also modify these values.
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 Tab	Description
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.

Function

The REQNUMARRAY block outputs (PV[n]) can be used as source parameters to provide predefined analog constants to other function blocks. A bad numeric output parameter typically has the value NaN (Not-a-Number).

The block supports these user configurable attributes.

- A configurable Number of Numeric Values (NNUMERIC) which lets you specify the desired number of numeric values to be supported.

Input/Output

The block has up to 64 outputs (PV[n]), depending on the number of numeric values (NNUMERIC) configured. But, all block pin parameters are available to be exposed and connected to using Control Builder graphical connections.

REQNUMARRAY parameters



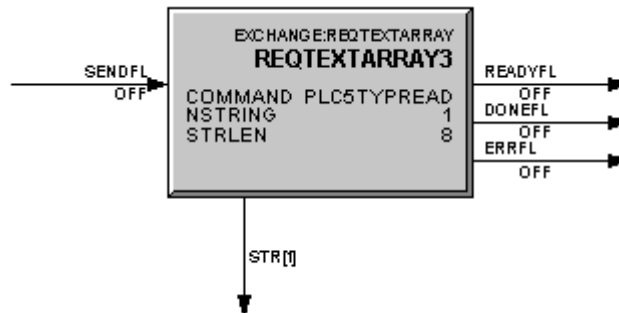
REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the REQNUMARRAY function block.

REQTEXTARRAY Block

Description

The REQTEXTARRAY block provides storage for up to 64 ASCII characters which are accessible through the corresponding string configuration parameter (STR[n]). Where “n” is the number of the text string. The length of the text strings is user configurable. It looks like this graphically:



Configuration Tab	Description
Main	- Name – Block (Tag) name of up to 16 characters long. Must be unique within the CM block containing it. - 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 0 to 65535. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. - Command (COMMAND) – Lets you set the block's current MODE. The selections are PLC5TYPREAD, PLC5TYPWRITE, CIPREAD, and CIPWRITE. (Note that the corresponding communications protocol for a PLC5TYPREAD or PLCTYPWRITE selection is PCCC (Programmable Controller Communications Commands) and the CIP (Control and Information Protocol) communications protocol is used for a

Configuration Tab	Description
	CIPREAD or CIPWRITE selection.)
	- Number of String Values (NSTRING) – Lets you set the amount of ASCII strings needed (1..8). - Char Length of String Values (STRLEN) - Lets you select the length of strings (1..64) needed. This is based on the NSTRING value. I.E. NSTRING = 4 then STRLEN = 16. (The product of NSTRING and STRLEN can not exceed 64.)
Communications	- Path to Device (PATH) - The relative path from the CPM to the target device (or the DHRIO module when the target device is on DH+) is specified as a comma-separated list of path segments. Each path segment is specified as an X, Y pair where X can either be 1 (indicating that this is an ICP Backplane Segment) or 2 (indicating that this is a ControlNet Segment). When X is 1, Y denotes a slot number and is in the range 0 to 16. When X is 2, Y denotes a ControlNet MAC ID and is in the range 1 to 99. The first path segment must be an ICP Backplane Segment. The path segment types, MAC IDs and slot numbers must be specified in decimal values. Binary, octal or hexadecimal values are not supported. IP Addresses are not supported. The following are examples of valid paths: a) 1, 5 (Path from the CPM to a device in slot 5 of the same chassis) b) 1, 0, 2, 3 (Path from the CPM through the CNI in slot 0 to a device that is on ControlNet with MAC ID 3) c) 1, 5, 2, 7, 1, 3 (Path from the CPM through the CNI in slot 5 to a CNI with MAC ID 7 to a device in slot 3) - When the target device is on the DH+ network, the path is the relative path from the CPM to the DHRIO module. Additional values are used to specify the address of the target device on the DH+ network. See the following DHFL parameter for details. - File Name in Target Device (FILENAME) –The File Name in the target device is specified as a Logical ASCII Symbolic address without the Logical ASCII

Configuration Tab	Description
	Identifier, when COMMAND is PLC5TYPREAD or PLC5TYPWRITE. For example, a valid PCCC File Name and Offset in a target device value might be N7:0. When COMMAND is CIPREAD or CIPWRITE, this is then the name of the file in the target device. - Use DH+ through DHRIO? (DHFL) – Lets you specify whether or not the target device is on an Allen-Bradley Data Highway and is connected through a DHRIO module. The default selection is UNCHECKED (target device is not on DH+). The following parameters are only active when the DHFL parameter is selected (CHECKED). - DH+ Channel A/B (DHCHANNEL) – Lets you specify which DH+ channel on the DHRIO to use for this connection. The default selection is A for Channel A. - DH+ Source Link (DHSRCLINK) – Lets you specify the source link for the DH+ route when using DH+ Remote Messaging. This parameter should be set to zero when using DH+ Local Messaging. The default selection is 0. - DH+ Destination Link (DHDESTLINK) – Lets you specify the destination link for the DH+ route when using DH+ Remote Messaging. This parameter should be set to zero when using DH+ Local Messaging. The default selection is 0. - Target Node Address (octal) (DHNODE) – Lets you specify the node number (0 to 77 octal) of the target node on the DH+ network. The default value is 0 (octal).
Status/Data	Process Value, from device (STR) - The Status/Data tab contains the initial data values to be written to the target. Control strategies in the CPM can modify these values at run-time. The operator at run-time can also modify these values.
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 Tab	Description
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.

Function

The REQTEXTARRAY block outputs (STR[n]) can be used to provide predefined text strings to other function blocks.

The block supports these user configurable attributes.

- A configurable Number of String Values (NSTRING) which lets you specify the desired number of string values (up to 64) to be supported.
- A configurable Character Length of String Values which lets you specify the number of characters (8, 16, 32, or 64) allowed in the strings.

The REQTEXTARRAY block supports a maximum size of 64 two-byte characters. The following table shows the maximum data combinations that you can configure through NSTRING and STRLEN values. Illegal combinations of NSTRING and STRLEN values, those requiring more than 64 two-byte characters of data, will be rejected.

NSTRING Value	STRLEN Value	STR[n] Range
1	64	[0]
2	32	[0. .1]
4	16	[0. .3]
8	8	[0. .7]

Input/Output

The block has up to 64 ASCII characters (STR[n]), depending on the number of string values (NSTRING) configured. But, all block pin parameters are available to be exposed and connected to using Control Builder graphical connections.

REQTEXTARRAY parameters



REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the REQTEXTARRAY function block.

RSPFLAGARRAY Block

Description

The RSPFLAGARRAY function block provides storage for up to 512 2-state values. The value can be accessed as a simple Boolean (Off or On) using the PVFL[n] parameter. Where “n” is the number of the flag. It looks like this graphically:



Configuration Tab	Description
Main	• Name – Block (Tag) name of up to 16 characters long. Must be unique within the CM block containing it. • PCCC File Number (FILENUM) – Lets you specify the file number (0 to 999) to be emulated when accessing this block through the PCCC (Programmable Controller Communications Commands) protocol from a remote device. The default value is 0. • CIP Tag Name (CIPNAME) – Lets you specify the tag name of this array when accessing this block through the CIP protocol from a remote device. The default entry is blank (no name). • Number of Flag Values (NFLAG) – Lets you set the amount of Flags you want to control (1 .. 512). The default value is 1. • Process Value, from device (PVFL) – Contains the values of the configured number of flag values that are the “target” for a remote device initiated read or write message request.
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 Tab	Description
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.

Function

Used to define two separate states (Off/On) to indicate status of a particular input.

- Number of flag values (NFLAG) is user configurable.
- Current state of flags can be read using flag value (PVFL[n] (Boolean).



ATTENTION

The Process Values (PVFL[N]) can be overwritten by operators or other programs (SCMs), when the value is also being written by a remote device as part of a write request type operation. Be sure your control strategy design does not allow write conflicts.

Input/Output

The block has up to 512 output flags (PVFL[n]). But, all block pin parameters are available to be exposed and connected to using Control Builder graphical connections.

RSPFLAGARRAY parameters



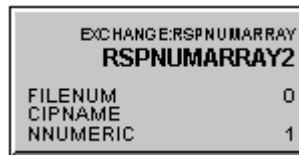
REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the RSPFLAGARRAY function block.

RSPNUMARRAY Block

Description

The RSPNUMARRAY block provides storage for up to 64 floating point values which are accessible through the corresponding PV configuration parameter (PV[n]). Where “n” is the number of the numeric. It looks like this graphically:



Configuration Tab	Description
Main	• Name – Block (Tag) name of up to 16 characters long. Must be unique within the CM block containing it. • PCCC File Number (FILENUM) – Lets you specify the file number (0 to 999) to be emulated when accessing this block through the PCCC (Programmable Controller Communications Commands) protocol from a remote device. The default value is 0. • CIP Tag Name (CIPNAME) – Lets you specify the tag name of this array when accessing this block through the CIP protocol from a remote device. The default entry is blank (no name). • Number of Numeric Values (NNUMERIC) – Lets you set the amount of floating point integers you want to control (1 .. 64). The default value is 1. • Data Type for PCCC/CIP access (DATATYPE) - Lets you specify the type of data that can be read from or written to by a remote device. The selections are FLOAT32, SIGNEDINT8, SIGNEDINT16 and SIGNEDINT 32. The default value is FLOAT32. • Process Value, from device (PV) – Contains the data values to be read from or written to by the remote device.

Configuration Tab	Description
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.

Function

The RSPNUMARRAY block outputs (PV[n]) can be used as source parameters to provide predefined analog constants to other function blocks. A bad numeric output parameter typically has the value NaN (Not-a-Number).

The block supports these user configurable attributes.

- A configurable Number of Numeric Values (NNUMERIC) lets you specify the desired number of numeric values to be supported.

Input/Output

The block has up to 64 outputs (PV[n]), depending on the number of numeric values (NNUMERIC) configured. But, all block pin parameters are available to be exposed and connected to using Control Builder graphical connections.

RSPNUMARRAY parameters



REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the RSPNUMARRAY function block.

RSPTEXTARRAY Block

Description

The RSPTEXTARRAY block provides storage for up to 64 ASCII characters which are accessible through the corresponding string configuration parameter (STR[n]). Where “n” is the number of the text string. The length of the text strings is user configurable. It looks like this graphically:



Configuration Tab	Description
Main	• Name – Block (Tag) name of up to 16 characters long. Must be unique within the CM block containing it. • PCCC File Number (FILENUM) – Lets you specify the file number (0 to 999) to be emulated when accessing this block through the PCCC (Programmable Controller Communications Commands) protocol from a remote device. The default value is 0. • CIP Tag Name (CIPNAME) – Lets you specify the tag name of this array when accessing this block through the CIP protocol from a remote device. The default entry is blank (no name). • Number of String Values (NSTRING) – Lets you set the amount of ASCII strings needed (1..8). • Character Length of String Values (STRLEN) - Lets you select the length of strings (1..64) needed. This is based on the NSTRING value. I.E. NSTRING = 4 then STRLEN = 16. (The product of NSTRING and STRLEN can not exceed 64.) • Process Value, from device (STR) – Contains the array of strings to be read from or written to by the

Configuration Tab	Description
	remote device.
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.

Function

The RSPTEXTARRAY block outputs (STR[n]) can be used to provide predefined text strings to other function blocks.

The block supports these user configurable attributes.

- A configurable Number of String Values (NSTRING) which lets you specify the desired number of string values (up to 8) to be supported.
- A configurable Character Length of String Values which lets you specify the number of characters (8, 16, 32, or 64) allowed in the strings.

The RSPTEXTARRAY block supports a maximum size of 64 two-byte characters. The following table shows the maximum data combinations that you can configure through NSTRING and STRLEN values. Illegal combinations of NSTRING and STRLEN values, those requiring more than 64 two-byte characters of data, will be rejected.

NSTRING Value	STRLEN Value	STR[n] Range
1	64	[0]
2	32	[0. .1]
4	16	[0. .3]
8	8	[0. .7]

Exchange Functions

RSPTTEXTARRAY Block

Input/Output

The block has up to 64 ASCII characters (STR[n]), depending on the number of string values (NSTRING) configured. But, all block pin parameters are available to be exposed and connected to using Control Builder graphical connections.

RSPTTEXTARRAY parameters



REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the RSPTTEXTARRAY function block.

Auxiliary Functions

Auxiliary Function Blocks

Functional Overview

Auxiliary function blocks provide a variety of configurable functions for conditioning, calculating, and compensating PV data in support of regulatory control functions.

The following table presents the various functions that can be performed through the configuration of the associated Auxiliary function block.

Block	Function	Description
AUXCALC (Auxiliary Calculation) Block	Compute a PV value	Lets you create custom algorithms by writing up to eight expressions. Each expression can contain any valid combination of inputs, operators and functions; and may perform arithmetic or logic operations, test conditions, etc. Optionally, it can accept up six inputs.
DEADTIME Block	Delay processing of input value changes	Lets you specify a fixed or a variable "dead-time" before a change in input value is calculated as a corresponding change in PV.
GENLIN (General Linearization) Block	Generate a linearized output for an input with non-linear characteristics.	Computes an output value that is a user-defined function of the input. You may define 2 to 13 coordinates, which, together, approximate a continuous non-linear function. Typically used to provide a linearized PV (in engineering units) for a sensor with nonlinear characteristics.
LEADLAG Block	Provide lead and lag compensation	Provides lead and lag compensation to a change in input value calculation for corresponding change in PV.
TOTALIZER Block	Accumulate total flows	Periodically adds an input value to an accumulator value, sets status flags to indicate when accumulator value is near user-specified target values ("near", "nearer", "actual target value"). Typically used to accumulate total flows. Block also supports warm restarts.

Common auxiliary block functions

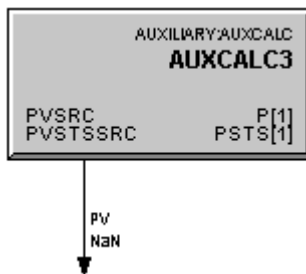
Listed below are the major functions performed by auxiliary function blocks along with a brief functional description for each. Functional descriptions for each block are given in the following subsections.

Major Function	Description
Input Processing	Auxiliary blocks get input data from other function blocks. Input processing gets this data, checks that its valid, and updates the appropriate block parameters.
Algorithm Calculation	This involves calculations that are unique to each block. The result or output is stored in PV. Depending of the particular Auxiliary block, additional functions may be included.

AUXCALC (Auxiliary Calculation) Block

Description

The AUXCALC (Auxiliary Calculation) block lets you write up to eight expressions for computing a PV value. Each expression may perform arithmetic or logic operations, test conditions, etc. Status information is made available for both the inputs, as well as the expression results. Through configuration, you can assign the result of an expression, a status, or an input to PV and PVSTS parameters. It looks like this graphically.



Function

The AUXCALC block evaluates user-defined expressions and conditions to compute the desired output and status for the control strategy.

As shown in Figure 35, the block may bring values from up to six inputs and determines their statuses in every execution cycle of the Control Module. It evaluates up to eight expressions and determines their statuses. It derives values for PV and PV status based on the configuration choices for the PVSRC and PVSTSSRC block parameters.

You can enter expression strings and configure PV and PV status selections at build time before the CM is loaded. The block performs syntax checking and conversion of the expression string during entry. If any errors are detected, they are displayed to inform you of the problem. You must re-enter the string to correct the error. You can only enter an expression in the Project tab during block configuration. You can **not** change an expression online in Monitoring tab.

The block checks and accepts other configuration parameters when the Control Module is active. If there are any invalid entries, it generates appropriate error messages to help identify the cause.

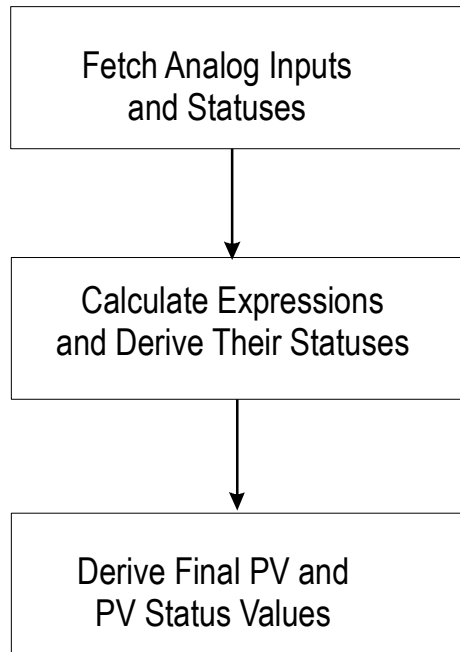


Figure 35 AUXCALC block major functions

Configuration example

Figure 36 shows a sample configuration that uses an AUXCALC block to provide square root characterization for the analog input. The AIC block always provides values in the range of 0 to 100. You can use the AUXCALC block to provide range conversion, if required. In this example, expression number 1 is configured as follows and C[1] is assigned to the PV output. The view in Figure 36 depicts a loaded configuration in Monitoring mode.

- exprn# 1 is: `SQRT(PIDLOOP1.AUXCALC2.P[1])`

Auxiliary Functions

AUXCALC (Auxiliary Calculation) Block

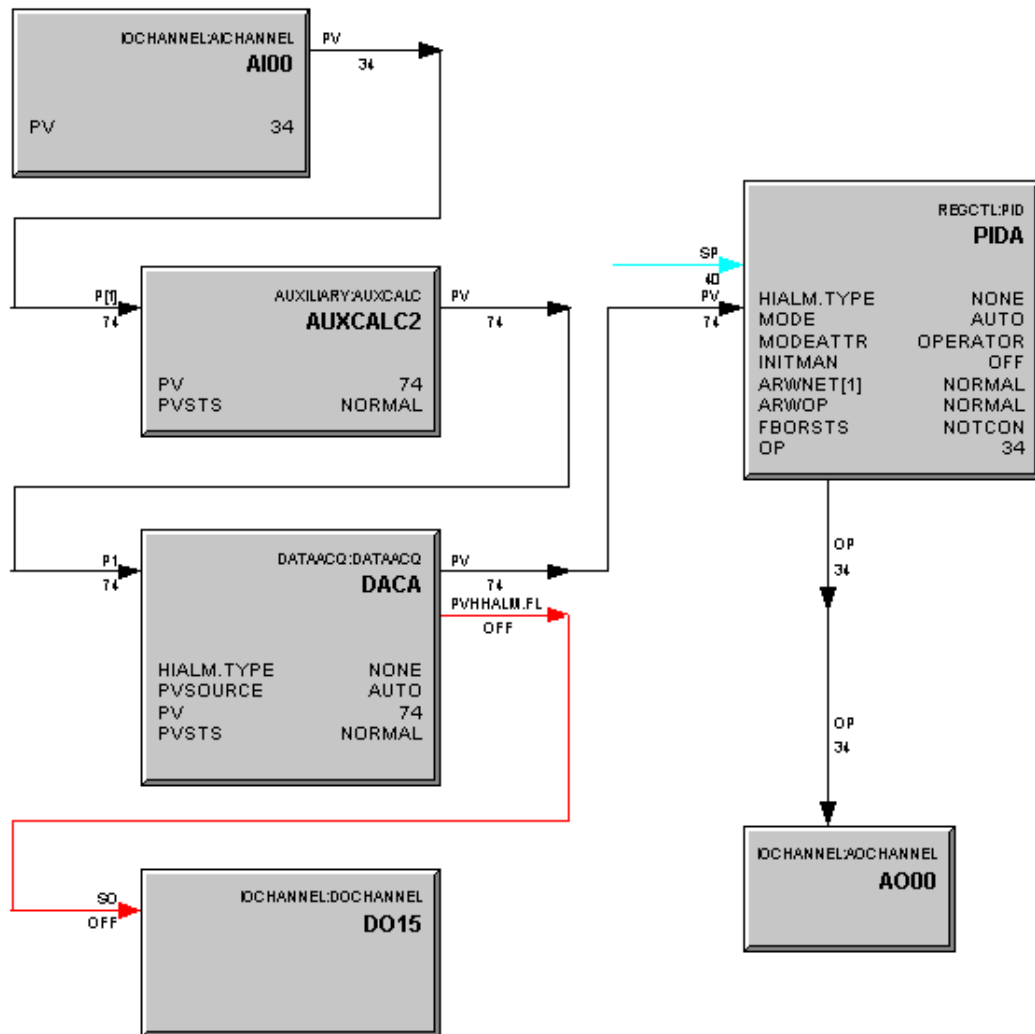


Figure 36 Example CB configuration using AUXCALC block for range conversion.

Input

This function block accepts as many as six inputs (P[1..6]):

- All inputs are optional.
- Must fetch all inputs from other function blocks.
- The number of process input connections are equal to the number of inputs; the default is 1.

Output

This block produces the following outputs:

- PV and its status, PVSTS
- As many as eight expression results (C[1] through C[8]) and their statuses

Expressions

You can write up to eight expressions, each expression can contain any valid combination of inputs, operators, and functions. Table 1([Expressions](#)) in the REGCALC block section lists the expression operators and functions supported by this block for reference.

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 P inputs in the expression and enclose identification number in brackets instead of parentheses. For example, CM151.AUXCALC BLOCK.P[1] * CM151.AUXCALC BLOCK.P[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 except Time 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.

Auxiliary Functions

AUXCALC (Auxiliary Calculation) Block

Assignable Outputs

Produces these outputs according to the values you assign to them.

- PV and its status PVSTS
- Up to eight expression results (C[1] to C[8]) and their statuses

You can assign an input, expression, result, or status value to PV and PVSTS through block configuration. For example, you may assign the result of the second expression(C[2]) to PV. You may also assign inputs directly to outputs; for example, P[1] can be assigned to PV, and P[2] can be assigned to PVSTS.

AUXCALC parameters



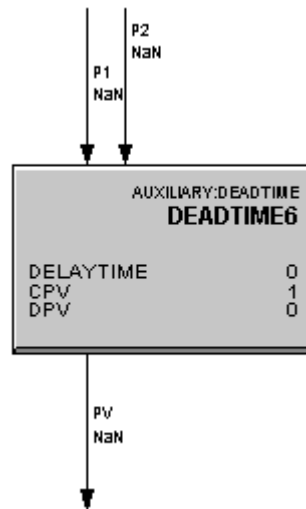
REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the AUXCALC block.

DEADTIME Block

Description

The DEADTIME block provides a user configurable fixed or variable dead-time delay in processing changes in its input (P1). The variable dead-time function varies as the inverse of a second input (P2) to the block. The block looks like this graphically:



Each DEADTIME 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. - 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 0 to 65535. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. - Overall Scale Factor (CPV) – Lets you specify the overall-scaling factor for the PV output. The default value is 1. - Factor for P1 (C1) – Lets you specify the scaling factor for the P1 input. The default value is 1. - Factor for P2 (C2) – Lets you specify the scaling factor for the P2 input. This only applies for the Variable delay type selection. The default value is 1. - Overall Bias (DPV) – Lets you specify an overall bias for the PV output. The default value is zero (0). - Bias for P1 (D1) – Lets you specify a bias for the P1 input. The default value is zero (0). - Bias for P2 (D2) – Lets you specify a bias for the P2 input. This only applies for the Variable delay type selection. The default value is zero (0). - Delay Type (DELAYTYP) – Lets you select the delay type as either Fixed or Variable. The default selection is Fixed. - Delay Time (minutes) (DELAYTIME) – Lets you specify the fixed delay time in minutes. This only applies for the Fixed selection. The default value is zero (0). - Delay Table Size (NUMLOC) – Lets you specify the number of locations to be used in the delay table. This only applies for the Variable selection. The default value is 60.

Configuration Tab	Description
	Cutoff Limit (CUTOFF.LM) – Lets you specify the zero-flow cutoff limit for the P2 input. When the P2 input is below the limit, the block sets the delayed P1 value to 0.0. The default value is NaN (Not-a-Number), which means there is no cutoff limit.
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.

Function

The DEADTIME block is typically used in a feedforward control loop. It provides its delayed PV output as the input to a LEADLAG block which feeds its output to the feedforward (FF) input of the PIDFF block. This helps condition the control response to the actual process characteristics.

The cutoff feature with the variable dead time lets you simulate conditions like the stopping of a conveyor belt. If the flow or speed value the P2 input represents drops below the value you configured for the CUTOFF.LM parameter, the value of the delayed P1 input goes to zero. When P2 again exceeds the Cut Off Limit value, the delayed P1 input resumes a value.

Input

The block requires one or two inputs depending on the type of delay action selected.

- If delay type is Fixed or Variable, P1 must be brought from another block.
- If delay type is Variable, P2 must also be brought from another block.

Output

The block produces an output value (PV), a status (PVSTS), and a status flag (PVSTSFL).

PV status

PV status (PVSTS) may have one of the following values:

- Bad - which means that PV is NaN (Not a Number)
- Normal - which means PV is OK.
- Manual – which means P1 source (for example, DATAACQ block) is in manual PV source.
- Uncertain - which means that PV is OK but P1 or P2 status is uncertain.

The following Boolean flags (typically used with Logic and Alarm blocks) also reflect the value of PVSTS:

- PVSTSFL.BAD – if PVSTS = Bad, this flag is on; otherwise it is off.
- PVSTSFL.NORM – if PVSTS = Normal, this flag is on; otherwise it is off.
- PVSTSFL.MAN – if PVSTS = Manual, this flag is ON; otherwise it is off.
- PVSTSFL.UNCER – if PVSTS = Uncertain, this flag is on; otherwise it is off.

Error handling

If the P1 input status (P1STS) or the P2 input status (P2STS) is Uncertain, this block sets PV status (PVSTS) to Uncertain.

If the P1 input status (P1STS) or the P2 input status (P2STS) is Bad, this block sets the PV status (PVSTS) to Bad and the PV output to NaN.

Delay type

The DEADTIME block gives you choice of either a Fixed or Variable delay type.

- For the Fixed delay, a change in the input value (P1) is delayed by the user configured delay time (DELAYTIME) as follows.

$$\begin{aligned}DP_t &= P1_{(t - \text{DELAYTIME})} \\PV &= CPV * DP_t + DPV\end{aligned}$$

Where:

$$\begin{aligned}CPV &= \text{Overall scale factor for PV} \\ \text{DELAYTIME} &= \text{Fixed delay time in minutes} \\ DPV &= \text{Overall bias for PV} \\ DP_t &= \text{Delayed P1 value (internal variable, not user accessible)} \\ t &= \text{Present time notation only (not a parameter)}\end{aligned}$$

- For the Variable delay, a change in the P1 input value is delayed by a time period, which varies as the inverse of the P2 input value. A combination of the P2 value, the scaling factors (C1, C2) and the bias values (D1, D2) determines the variable time period as follows.

If CUTOFF.LM is not NaN and P2 is less than CUTOFF.LM:

$$DP_t = 0$$

Otherwise:

$$\begin{aligned}\text{DELAYTIME} &= [C1 / (C2 * P2 + D2)] + D1 \\ DP_t &= P1_{(t - \text{DELAYTIME})}\end{aligned}$$

And:

$$PV = CPV * DP_t + DPV$$

Where

$$\begin{aligned}C1 &= \text{Scaling factor in the calculation of the DELAYTIME} \\ C2 &= \text{Scaling factor for P2}\end{aligned}$$

Auxiliary Functions

DEADTIME Block

CPV	= Overall scale factor for PV
CUTOFF.LM	= Cutoff limit, for example, corresponding to zero flow or zero conveyor belt speed.
D1	= Bias in the calculation of the DELAYTIME, equivalent to a fixed delay
D2	= Bias for P2
DELAYTIME	= Fixed delay time in minutes
DPV	= Overall bias for PV
DP _t	= Delayed P1 value (internal variable, not user accessible)
P1	= Input value to which the delay is applied
P2	= Input value that changes the variable delay
t	= Present time notation only (not a parameter)

Delay table

The block uses a delay table (DELAYTABLE) to produce the desired delays in the P1 input. It stores and shifts P1 values through the table at a rate that is calculated to produce the desired deadtime. The following information is used to derive the table-shift rate.

- The sample rate of the P1 input value. This is the execution rate of the block.
- The delay time (DELAYTIME). For Fixed delay, delay time is user configured. For Variable delay, the delay time is derived from the P2 input.
- The number of entries (NUMLOC) to use in the delay table. The table has a maximum of 60 entries. You can change the number of entries by configuring the desired smaller value through the Delay Table Size (NUMLOC) entry in the block's configuration form.

The following relationship exists between DELAYTIME, Period (FB execution period in minutes) and NUMLOC.

$$\text{DELAYTIME} \geq \text{Period} * \text{NUMLOC}$$

$$\text{DELAYTIME} \leq \text{Period} * 3200$$

In the simplest case, where the scaling factors C1 and C2 equal 1 and the bias factors D1 and D2 equal 0, the variable delay time input signal P2 has the following limits.

$$P2 \leq 1 / (\text{Period} * \text{NUMLOC})$$

$$P2 \geq 1 / (\text{Period} * 3200)$$

In all other cases, use the scaling and bias factors to make sure the calculated delay time remains within the range defined above.



ATTENTION

- Using delays greater than two minutes or reducing the delay table size, will distort the input signal as it appears at the PV output. Input signals with high frequency content will cause samples to be missed, even at the maximum sample rate, resulting in reduced output fidelity.
 - When the delay time exceeds the product of the sample rate and the delay table size, the input value, which lies between other sampled inputs, is interpolated. This means the PV output is either a true sampled value or an interpolated value.
 - You can connect DEADTIME blocks in series to achieve longer delays.
-

Restart condition

When this block experiences a Restart condition, all the entries in the delay table are set equal to P1. The PV status is set to Normal and the PV is calculated as follows.

$$PV = CPV * P1 + DPV$$

When the INITREQ parameter is True, the block's algorithm produces the same result as the Restart condition.

DEADTIME parameters



REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the DEADTIME block.

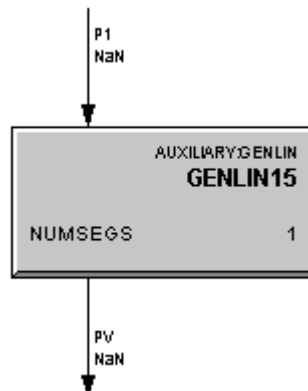
Auxiliary Functions

GENLIN (General Linearization) Block

GENLIN (General Linearization) Block

Description

The GENLIN (General Linearization) block calculates an output value (PV) as a function of the input value (P1) based on a separate function that can be represented by 2 to 13 user-defined coordinates. (You specify the IN and OUT values of each coordinate to make a segment.) The input value (P1) is then compared with the input range of each segment and the output is set at the intersection of the input with the appropriate segment. The GENLIN block looks like this graphically:



Each time the GENLIN block runs, it compares the input value (P1) with each segment based on a coordinate pair – starting with the first and continuing until it finds a segment that intersects with the input. When that segment is found, the block derives the output (PV) as follows:

- If P1 is exactly equal to the input value at the beginning of any segment (that is, $P1 = IN[i]$, for i in the range of 0 to NUMSEGS):

$$PV = OUT[i]$$

- If P1 intersects the first segment (that is, $P1 < IN[1]$):

$$PV = \frac{OUT(1) - OUT(0)}{IN(1) - IN(0)} * [P1 - IN(0)] + OUT(0)$$

- If P1 intersects the last segment (that is, $P1 > IN[i]$ for $i = NUMSEGS - 1$):

$$PV = \frac{OUT(NUMSEGS) - OUT(i)}{IN(NUMSEGS) - IN(i)} * [P1 - IN(i)] + OUT(i)$$

- If P1 intersects any other segment (that is, $IN[i] < P1 < IN[i + 1]$ for $i = 1$ to $NUMSEGS - 2$):

$$PV = \frac{OUT(i + 1) - OUT(i)}{IN(i + 1) - IN(i)} * [P1 - IN(i)] + OUT(i)$$

where:

$IN[i]$ = input value at the beginning of the intersecting segment.

$IN[i + 1]$ = input value at the end of the intersecting segment

$OUT[i]$ = output value at the beginning of the intersecting segment

$OUT[i + 1]$ = output value at the end of the intersecting segment

NUMSEGS = total number of segments in the curve based on 2 to 13 user defined coordinate pairs.

Auxiliary Functions

GENLIN (General Linearization) Block



ATTENTION

- The first and last segments are treated as if they are infinitely extended. So if P1 is less than IN[0] or greater than IN[NUMSEGS], PV is computed by assuming that the slope in the appropriate segment continues to the intersecting point.
 - The segment coordinate values (IN[i]) must be specified in ascending order, from smallest to largest value.
-

Function

The GENLIN block is typically used to provide a linearized PV (in engineering units) for a sensor with nonlinear characteristics. The GENLIN block can also be used to characterize functions of a single parameter, such as heat transfer versus flow rate, or efficiency as a function of load. It is particularly useful when the relationship of the input to engineering units is empirically determined.

Inputs

The GENLIN block requires one input value (P1):

- P1 must be brought from another function block.
- P1STS represents the status of P1.

Outputs

The GENLIN block produces the following output:

- PV and its status, PVSTS. It also sets Boolean flags PVSTSFL to reflect the status of PVSTS for logical use.

Error handling

- If P1STS is Uncertain, the GENLIN block sets PVSTS to uncertain.
- If P1STS is Bad, or if any of the segment coordinates (IN[i] or OUT[i]) contains NaN (Not a Number), this block sets PVSTS to Bad.
- If any of the segment coordinates (IN[i] or OUT[i]) contains NaN (not a Number, the Control Module that contains the GENLIN block will not be allowed to go Active (EXECSTATE = Active).

GENLIN parameters



REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the GENLIN block.

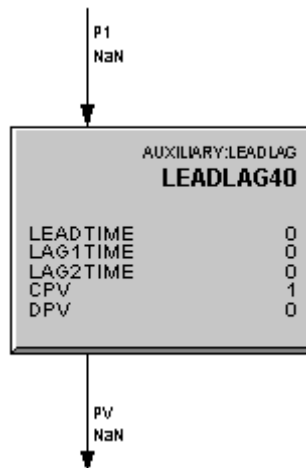
LEADLAG Block

Description

The LEADLAG block provides dynamic lead-lag compensation for changes in its input (P1). It subjects a change in the input value (P1) to one lead compensation and two lag compensation factors.

There is a user configurable time constant for each compensation factor. You can suppress a compensation factor by setting its corresponding time constant to zero (0).

The block looks like this graphically:



Each LEADLAG 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. • PV Format (PVFORMAT) – Lets you select the decimal format to be used to display the PV values. The selections are D0 for no decimal place (-XXXXXX.), D1 for one decimal place (-XXXXX.X), D2 for two decimal places (-XXXX.XX), and D3 for three decimal places (-XXX.XXX). The default selection is D1 for one decimal place. • 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 0 to 65535. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. • Lead Time (min) (LEADTIME) – Lets you specify the lead time constant in minutes. The default value is 0, which means the lead time compensation is suppressed. . • Lag 1 Time (min) (LAG1TIME) – Lets you specify the first order lag time constant in minutes. The default value is 0, which means the first order lag time compensation is suppressed. • Lag 2 Time (min) (LAG2TIME) – Lets you specify the second order lag time constant in minutes. The default value is 0, which means the second order lag time compensation is suppressed. • Overall Scale Factor (CPV) – Lets you specify the overall-scaling factor for the PV output. The default value is 1. • Overall Bias (DPV) – Lets you specify an overall bias for the PV output. The default value is zero (0).
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 Tab	Description
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.

Function

The LEADLAG block is typically used in a feedforward control loop. It provides its compensated PV output as the input to the feedforward (FF) input of the PIDFF block. This helps condition the control response to the actual process characteristics.

Input

The block requires one input. P1 must be brought from another block.

Output

The block produces an output value (PV), a status (PVSTS), and a status flag (PVSTSFL).

PV status

PV status (PVSTS) may have one of the following values:

- Bad - which means that PV is NaN (Not a Number)
- Normal - which means PV is OK.
- Manual - which means P1 source (for example, DATAACQ block) is in manual PV source.
- Uncertain - which means that PV is OK but P1 status is uncertain.

The following Boolean flags (typically used with Logic and Alarm blocks) also reflect the value of PVSTS:

- PVSTSFL.BAD – if PVSTS = Bad, this flag is on; otherwise it is off.
- PVSTSFL.NORM – if PVSTS = Normal, this flag is on; otherwise it is off.
- PVSTSFL.MAN – if PVSTS = Manual, this flag is ON; otherwise it is off.
- PVSTSFL.UNCER – if PVSTS = Uncertain, this flag is on; otherwise it is off.

Error handling

If the P1 input status (P1STS) is Uncertain, this block sets PV status (PVSTS) to Uncertain.

If the P1 input status (P1STS) is Bad, this block sets the PV status (PVSTS) to Bad and the PV output to NaN.

Equation

The LEADLAG block applies the following equation.

$$PV = L^{-1} [CPV * (1 + LEADTIME * s) / \{(1 + LAG1TIME * s) * (1 + LAG2TIME * s)\} * P1(s)] + DPV$$

Where:

CPV = Overall scale factor for PV

DPV = Overall bias for PV

L^{-1} = Inverse of the LaPlace transform

LAG1TIME = First order lag time constant (If 0, no first order lag.)

LAG2TIME = Second first order lag time constant (If 0, no second order lag.)

LEADTIME = Lead time constant (If 0, no lead time.)

P1 = Input value to which lead and lag compensation is applied

PV = Output of this block

s = LaPlace operator notation only (not a parameter)

Time constant recommendations

The execution rate of the LEADLAG block should be greater than the lowest break-point period of the block as follows.

- The first order lag time (LAG1TIME) should be greater than or equal to $2 * TS$.
Where TS is the sample time in minutes.
- The second order lag time (LAG2TIME) should be greater than or equal to $2 * TS$.
- The absolute lead time (|LEADTIME|) should be greater than or equal to $2 * TS$.
(Note that the absolute value of lead time is used, since both positive and negative lead times can be specified.)

Restart condition

When this block experiences a Restart condition, the lead-lag dynamics are set to a steady state and the PV is calculated as follows.

$$PV = CPV * P1 + DPV$$

When the INITREQ parameter is True, the block's algorithm produces the same result as the Restart condition.

LEADLAG parameters



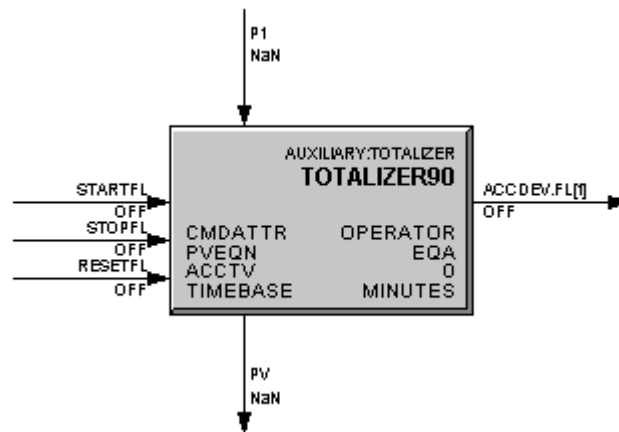
REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the LEADLAG block.

TOTALIZER Block

Description

The TOTALIZER block periodically adds an input value (P1) to an accumulator value (PV). It looks like this graphically:



You specify a target value for the accumulator, and up to four trip points, which are "near" and "nearer to" the target value. The TOTALIZER block sets status flags to indicate when the accumulator value is near (and nearer to) the user-specified target values. A trapezoidal-integration method of accumulation is used to improve accuracy. Accumulation proceeds even when the target value is exceeded. An external operator or program command is required to stop the block from further accumulating.

Function

The TOTALIZER block is typically used to accumulate total flows. For situations where the flow transmitter may not be precisely calibrated near the zero-flow value, a zero-flow cutoff feature is provided such that when P1 is below the cutoff value it clamps to zero.

Configuration example

Figure 37 and its companion callout description table show a sample configuration that uses a TOTALIZER block in a flow control loop for quick reference. The view in Figure 37 depicts a loaded configuration in Monitoring mode.

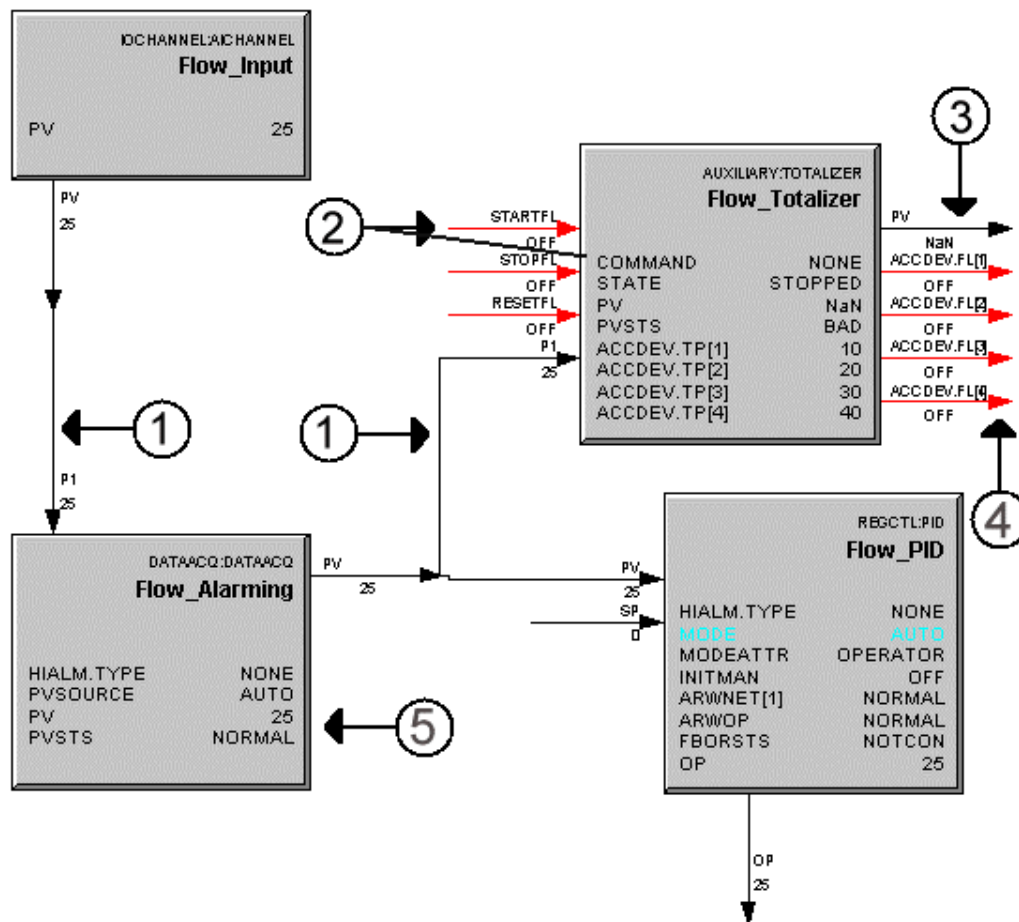


Figure 37 Example of CB configuration using a TOTALIZER block in a flow control loop.

The following table includes descriptions of the callouts in Figure 37.

Callout	Description
1	Use the PV parameter connection to carry data and status from the analog input, DATAACQ, and TOTALIZER blocks 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, you can use the COMMAND parameter on the block to issue Start, Stop, or Reset command. You must configure COMMAND as a monitoring parameter through the block configuration form. You can also use logic inputs to STARTFL, STOPFL, and RESETFL pins on the block to initiate Start, Stop, and Reset commands, respectively.
3	When the accumulated value (PV) reaches the accumulated target value (ACCTV), the accumulated target value flag (ACCTVFL) turns ON.
4	In this example, the following values were configured for the Trip Points 1 to 4 through the parameter configuration form based on a configured target value of 100. • Trip Point 1 (ACCDEV.TP[1] = 10 • Trip Point 2 (ACCDEV.TP[2] = 20 • Trip Point 3 (ACCDEV.TP[3] = 30 • Trip Point 4 (ACCDEV.TP[4] = 40 Based on these configured Trip Point values, the corresponding accumulated deviation flag will turn ON at the following accumulated values. • ACCDEV.FL[1] turns ON at PV = 90 • ACCDEV.FL[2] turns ON at PV = 80 • ACCDEV.FL[3] turns ON at PV = 70 • ACCDEV.FL[4] turns ON at PV = 60
5	Be sure to configure the ORDERINCM parameters for the DATAACQ block and the AICHANNEL block to be lower numbers than the ORDERINCM parameter for the TOTALIZER block, so the DATAACQ and AICHANNEL blocks execute before the TOTALIZER block. This configuration avoids possible TOTALIZER interruptions during a warm restart scenario.

Input

The TOTALIZER block requires one input (P1):

- P1 is the value to be accumulated – the input value may be real, integer, or Boolean, but is always stored as a real number.
- P1 must be brought from another block.

Outputs

The TOTALIZER block produces the following outputs:

- The accumulated value (PV) and its status (PVSTS).
- Flags, indicating if the accumulated value has reached the user-specified target value or one of the accumulator deviation trip points (ACCTVFL and ACCDEV.FL [1-4]).

TOTALIZER states

The TOTALIZER block has two possible states: Stopped and Running. The STATE parameter identifies the current state and the following parameters may be used to change the state:

- **COMMAND:** The operator or a user program may command the accumulator to Start, Stop, or Reset by storing to the COMMAND parameter. Since COMMAND is a write-only parameter, its displayed value does not reflect the last entered command.

Possible choices are:

- **Start** – requests the TOTALIZER to start the accumulation (change STATE to Running). The Totalizer block must be reset using the reset pin (RESETFL) prior to counting.
- **Stop** – requests the TOTALIZER to stop the accumulation (change STATE to Stopped).
- **Reset** – requests the TOTALIZER to reset the accumulated value (PV) with a user-specified reset value (RESETVAL). STATE will not change; if the accumulator is running, it continues from the reset value.

Totalizer must be reset using the reset pin before the totalizer can start counting. Otherwise P1 will have a good value, but PV will remain at zero.

When the TOTALIZER receives a reset command, it copies the current value of PV

to OLDAV (old accumulation value), and then sets PV equal to RESETVAL. This allows other system functions using the totaled value to reset the TOTALIZER without losing any "accumulation".

- CMDATTR: Specifies who may store to COMMAND (that is, either the operator or a user program). CMDATTR is used to prevent the operator from inadvertently changing the accumulator while it is under program control and allows the operator to override a program.

Possible choices are:

- Operator – only the operator may store to COMMAND.
- Program – only a program may store to COMMAND; the operator may override the program by setting CMDATTR = Operator.
- STARTFL (Start Flag): Allows either a Logic block or user-written program to store to COMMAND.
 - Off-to-On transitions cause the TOTALIZER state to change to Running.
- STOPFL (Stop Flag): Allows either a Logic block or user-written program to store to COMMAND.
 - Off-to-On transitions cause the TOTALIZER state to change to Stop.
- RESETFL (Reset Flag): Allows either a Logic block or user-written program to store to COMMAND.
 - Off-to-On transitions cause the TOTALIZER to be reset.

Accumulator target value

Prior to starting the TOTALIZER, you may specify a target value for the accumulator (ACCTV). The TOTALIZER block compares PV with ACCTV on each cycle and sets the target-value-reached flag (ACCTVFL) to ON when the accumulation is complete (that is, when PV is greater than or equal to ACCTV).

Deviation trip points

The TOTALIZER block provides trip points and flags to signal when the accumulated value is "getting close" to the target value. You may specify as many as four trip points, and the TOTALIZER block sets a corresponding flag when each trip point is reached. The flags are typically monitored by another function block that can initiate some sort of control action (for example, changing a valve position from full open to trickle when a TOTALIZER trip point is reached).

The trip point values (ACCDEV.TP[1-4]) are expressed as deviations from the target value. The TOTALIZER block compares the actual deviation (ACCTV - PV) with each trip point, and sets a flag (ACCDEV.FL[1-4]) when the deviation is less than or equal to a trip point. For example, if the user sets ACCTV = 50 and ACCDEV.TP[1] = 10, the TOTALIZER block sets ACCDEV.FL[1] to ON when PV is greater than or equal to 40.

Equations

PVEQN is a user-configured parameter, which specifies how the TOTALIZER should handle bad inputs and warm restarts. One of the following equations is specified using PVEQN:

Equation	Bad Input Handling	Warm Restart Handling
A	Use zero if input is bad.	Continue after a warm restart.
B	Use last good value if input is bad.	Continue after a warm restart.
C	Stop if the input is bad and set PV to NaN.	Continue after a warm restart.
D	Use zero if input is bad.	Stop after a warm restart.
E	Use last good value if input is bad.	Stop after a warm restart.
F	Stop if the input is bad and set PV to NaN.	Stop after a warm restart.

The following table summarizes block actions associated with a given PVEQN handling option relative to the accumulator state and the input status. .

If Accumulator is . . .	And Option is . . .	Then, block . . .
Running (STATE = RUNNING) and the input status (P1STS) is BAD	Use zero if input is bad	Sets the input value (P1) to zero, sets PVSTS to Uncertain, and continues the accumulation. When the input status (P1STS) returns to normal, PVSTS remains Uncertain until a Reset command is received.
Running (STATE = RUNNING) and the input status (P1STS) is BAD	Use last good value if input is bad	Sets the input value (P1) to its last good value, sets PVSTS to Uncertain, and continues the accumulation. When the input status (P1STS) returns to normal, PVSTS remains Uncertain until a Reset command is received.
Running (STATE = RUNNING) and the input status (P1STS) is BAD	Stop if the input is bad	Sets the input value (P1) to NaN (Not a Number), sets PVSTS to Bad, and stops the accumulation. When the input status (P1STS) returns to normal, PVSTS remains Bad until the operator restarts the accumulation. To restart the accumulator, the operator must estimate the accumulated value, issue a Reset command to establish that value, and then issue a Start command. The last accumulated value before the status went bad is designated as LASTGOOD.
Running (STATE = RUNNING)	Continue after a warm restart	Sets PVSTS to Uncertain and continues accumulation from last value of PV. PVSTS remains Uncertain until a Reset command is received.

Auxiliary Functions

TOTALIZER Block

If Accumulator is . . .	And Option is . . .	Then, block . . .
Running (STATE = RUNNING)	Stop after a warm restart	Sets the accumulated value (PV) to NaN (Not a Number), sets PVSTS to Bad, and stops the accumulation. The operator must intervene to restart the accumulator.

Accumulated value calculation

For equations A through F, the accumulated value (PV) is calculated as follows:

$$PV_i = PV_{(i-1)} + C1 \bullet \text{time\_scale} \bullet (P_{(i-1)} + [P_i - P_{(i-1)}] / 2)$$

Where:

PV_i = TOTALIZER block output from the current pass

$PV_{(i-1)}$ = accumulated value at the end of block's last processing pass

$C1$ = scale factor for P1; used to convert to different engineering units

P_i = input value from current pass

$P_{(i-1)}$ = input value from last pass

time_scale = (TS * 60) if TIMEBASE = seconds
(TS) if TIMEBASE = minutes
(TS / 60) if TIMEBASE = hours

where TS = TOTALIZER block's processing interval, in minutes

Error handling

- PVSTS is set to UNCERTAIN when:
 - The status of the input (P1STS) is Uncertain.
 - The input status is Bad and the "use zero" or "use last good value if input is bad" option is configured (Equation A, B, D, or E).
 - The TOTALIZER block is in warm restart and the "continue" option is configured (Equation A, B, or C).
- PV is set to NaN (Not a Number) and PVSTS is set to Bad, when:
 - The status of the input (P1STS) is Bad and the "stop if input is bad" option is configured (Equation C or F).
 - The TOTALIZER block is in warm restart and the "stop" option is configured (Equation D, E, or F).
- When PVSTS is Bad, the TOTALIZER block sets ACCTVFL and ACCDEV.FL[1-4] to Off.



ATTENTION

When the input status returns to normal, a Reset command is needed to return PVSTS to Normal.

Restart and activation

When a TOTALIZER block is activated:

- PV is set to NaN (Not a Number).
- PVSTS is set to Bad.
- The accumulator is stopped (that is, STATE = Stopped).

TOTALIZER parameters



REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the TOTALIZER block.

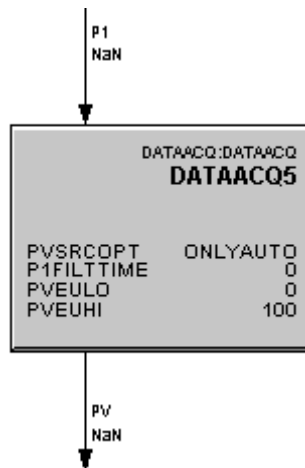
Auxiliary Functions
TOTALIZER Block

Data Acquisition Functions

DATAACQ (Data Acquisition) Block

Description

The DATAACQ (Data Acquisition) block processes a specified process input value (P1) into a desired output value (PV). It looks like this graphically.



Each DATAACQ 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 0 to 65535. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. - PV Source Option (PVSRCOPT) – Lets you select whether you want to limit the PV source to AUTO only or allow other PV source selections. The default selection is ONLYAUTO. - PV Source (PVSOURCE) – Lets you select the source of the PV as SUB for a user program, MAN for an operator, or AUTO for process input connection. Only applicable with PV Source Option selection of ALL. The default selection is AUTO. - PV Format (PVFORMAT) – Lets you select the decimal format to be used to display the PV values. The selections are D0 for no decimal place (-XXXXXX.), D1 for one decimal place (-XXXXX.X), D2 for two decimal places (-XXXX.XX), and D3 for three decimal places (-XXX.XXX). The default selection is D1 for one decimal place. - PV Character (PVCHAR) – Lets you select whether or not you want to apply Linear or Square Root PV characterization conversion to the input (P1). The default selection is NONE, which means no characterization conversion is applied. - PVEU Range High (PVEUHI) – Lets you specify the high input range value in engineering units that

Configuration Tab	Description
	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). • PV Limits Hi (PVEXHILM) – Lets you specify a high limit value for the PV in engineering units. If the PV value exceeds this limit, the block clamps the PV to the limit value and sets the PV high limit flag (PVEXHIFL). The default value is 102.9. • PV Limits Low (PVEXLOLM) – Lets you specify a low limit value for the PV in engineering units. If the PV value falls below this limit, the block clamps the PV to the limit value and sets the PV low limit flag (PVEXLOFL). The default value is –2.9. • Low Signal Cut Off (LOCUTOFF) – Lets you specify the low signal cutoff limit for the P1 input after filtering and clamping. When PVAUTO is below the limit, the block sets the PVAUTO value to the PVEULO value. Only applicable with PV character selection of Linear or Square Root. The default value is NaN (Not-a-Number), which means there is no cutoff limit. • Clamping Option (P1CLAMPOPT) – Lets you specify whether or not you want P1 to be clamped within the PV high (PVEXHILM) and low (PVEXLOLM) limits. The default setting is DISABLE, which means no clamping is applied. • Lag Time (P1FILTME) – Lets you specify a first order filter time in minutes for the P1 input. When time is non-zero (1 to 60 minutes), a first-order filter is applied to P1EU and the result is stored in an intermediate parameter called FilteredP1 (not a visible parameter). As long as FilteredP1 is within PV limits, it is copied to PVAUTO. See <i>Input Filtering</i> in this section for more details. The default value is 0.
Alarms	• Alarm Limits – Identifies the types of alarms this block supports. Of course, these alarms also interact with other block configuration values such as PVEU Range Hi and PVEU Range Lo. The types are:

Configuration Tab	Description
	– PV High High (PVHHALM.FL) – PV High (PVHIALM.FL) – PV Low (PVLOALM.FL) – PV Low Low (PVLLALM.FL) – Positive Rate of Change (ROCPOSALM.FL) – Negative Rate of Change (ROCNEGALM.FL) – Bad PV (BADPVALM.FL) – High Significant Change (PVHISIGCHG.TP) – Low Significant Change (PVLOSIGCHG.TP) • Trip Point – Lets you specify the following trip points for the given alarm. The default value is NaN, which disables the trip point. – PVHHALM.TP (PV High High Alarm Trip Point) – PVHIALM.TP (PV High Alarm Trip Point) – PVLOALM.TP (PV Low Alarm Trip Point) – PVLLALM.TP (PV Low Low Alarm Trip Point) – ROCPOSALM.TP (Positive Rate of Change Alarm Trip Point) – ROCNEGALM.TP (Negative Rate of Change Alarm Trip Point) – PVHISIGCHG.TP (High Significant Change Alarm Trip Point) – PVLOSIGCHG.TP TP (Low Significant Change Alarm Trip Point) • Priority – Lets you set the desired priority level individually for each alarm type (PVHHALM.PR, PVHIALM.PR, PVLOALM.PR, PVLLALM.PR, ROCPOSALM.PR, ROCNEGALM.PR, and BADPVALM.PR). The default value is LOW. The levels are: – NONE - Alarm is neither reported nor annunciated.

Configuration Tab	Description
	– 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 (PVHHALM.SV, PVHIALM.SV, PVLOALM.SV, PVLLALM.SV, ROCPOSALM.SV, ROCNEGALM.SV, and BADPVALM.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, PVHIALM.DB and PVLOALM.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, PVHIALM.TM and PVLOALM.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, PVHIALM.DBU and PVLOALM.DBU) when the CM is loaded. If you configure the individual alarm parameters as Monitoring

Data Acquisition Functions
DATAACQ (Data Acquisition) Block

Configuration Tab	Description
	Parameters for the block, you can change the individual alarm value while monitoring the loaded block in CB.
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.

Function

The DATAACQ block is normally configured to fetch an analog input from an AI Channel function block. As shown in Figure 38, it performs the following major functions:

- Input Processing - fetches input data from another block through process connections, checks its validity, and updates input parameters P1 and P1STS as appropriate.
- PV Characterization – converts input parameter P1 to Engineering Units, when the user configurable PV Characterization option is configured as Linear or Square Root. The converted P1 value is stored in a read-only parameter (P1EU).
- Filtering and Clamping – performs filtering and clamping on the read-only parameter P1EU and stores the result in PVAUTO. There are user configurable parameters associated with both the filtering (P1FILTIME) and clamping (P1CLAMPOPT) functions.
- Low Signal Cut Off – Applies a user configurable low signal cut off limit to the PVAUTO value after filtering and clamping.
- PV Source Selection - normally copies the filtered and clamped value of PVAUTO to the output PV, but also allows for instances where the operator or user program can store to PV, if the user configurable PV Source selection is configured for MAN or SUB, respectively.
- Alarm Processing - generates alarm flags when PV exceeds any of a number of user-specified alarm trip points for longer than a designated time interval.

These functions are discussed in more detail in the following paragraphs.

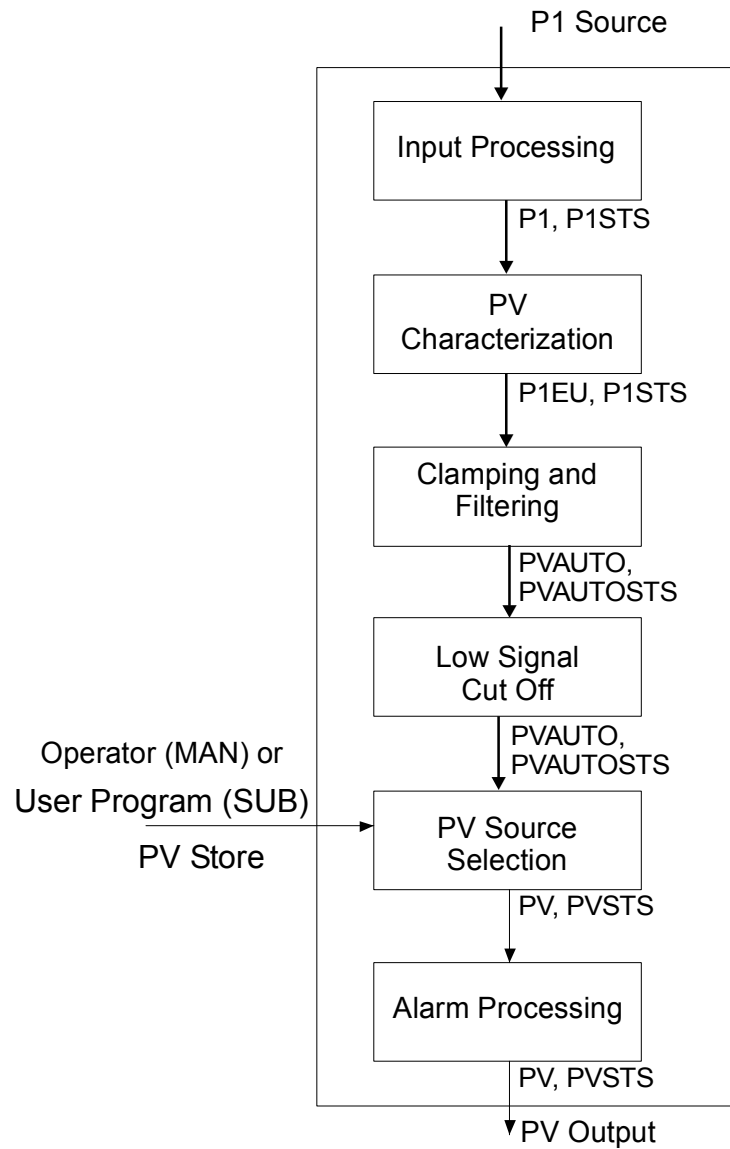


Figure 38 DATAACQ block major functions.

Input

The DATAACQ block requires one process input value – P1. P1 must be brought from another block.

Input ranges and limits

PVEUHI and PVEULO define the full range of P1 in engineering units.

- PVEUHI represents the 100% of full-scale value.
- PVEULO represents the 0% of full-scale value.

PVEXHILM and PVEXLOLM define the high and low limits of P1 in engineering units.

- If P1 clamping is desired (P1CLAMPOPT = Enable), the DATAACQ block clamps the input within the range defined by PVEXHILM and PVEXLOLM.

P1 status

You must configure the DATAACQ block to bring P1 from another block. Typically, the other block is an AI Channel block. If the P1 source provides a value and status, the DATAACQ block fetches both; otherwise it fetches the value only and derives a status from that.

- If the P1 source provides a value and status, the status (P1STS) may have one of the following values:
 - BAD – value is NaN (Not a Number)
 - Normal – value is OK.
 - Manual – value is OK, but was stored by an operator (at the source block)
 - Uncertain – value is OK, but was stored by a user-program (at the source block)
- If the P1 source provides a value only, the block derives P1STS as follows:
 - If P1 is NaN (Not a Number), then: P1STS = Bad.
 - Otherwise, P1STS = Normal.
- If P1 cannot be fetched (for example, due to a communications error), P1 is set to NaN and P1STS is set to Bad.

PV Characterization

You can configure the PV Characterization option to have the DATAACQ block provide one of the following conversion functions.

- **LINEAR:** Converts P1 to Engineering Units based on the 0 to 100 input span (100) and the configured PV span in Engineering Units (PVEUHI – PVEULO). The linear conversion is calculated as follows.

$$P1EU = (P1 / 100) * (PVEUHI - PVEULO) + PVEULO$$

where:

P1 = Process input value from another block

P1EU = P1 value in Engineering Units

PVEUHI = User configured PV high range value in Engineering Units for 100% full scale

PVEULO = User configured PV low range value in Engineering Units for 0% full scale

100 = Span for 0 to 100 input range

For example, If you want to convert the P1 input to a range of 0 to 1200 degrees, configure PVEULO as “0” and PVEUHI as “1200”. In this case, if P1 input is 50%, P1EU equals $(50 / 100) * (1200 - 0) + 0$ or $0.5 * 1200$ equals 600 degrees.

- **SQUARE ROOT:** Applies a square root calculation to the P1 input such that 100% of span equals 1.0. Then, converts the square root value to Engineering Units based on the configured PV span in Engineering Units (PVEUHI – PVEULO). The Square Root conversion is calculated as follows.

- For P1 input greater than or equal to zero (0):

$$P1EU = \text{SQRT}(P1 / 100) * (PVEUHI - PVEULO) + PVEULO$$

- For P1 input less than zero (0):

$$P1EU = \text{SQRT}(-P1 / 100) * (PVEUHI - PVEULO) + PVEULO$$

For example, If you want to convert the P1 input to a range of 0 to 1200 gallons per hour, configure PVEULO as “0” and PVEUHI as “1200”. In this case, if P1 input is 40%, P1EU equals the square root of $(40 / 100) * (1200 - 0) + 0$ or $0.632 * 1200$ equals 758.4 gallons per hour.

- NONE: Applies no conversion to the P1 input.

Input filtering

The P1 FILTTIME parameter indicates if P1 should be filtered. If a non-zero filter time (P1FILTTIME) is specified, a first-order filter is applied to P1EU and the result is stored in an intermediate parameter called FilteredP1 (not a visible parameter). As long as FilteredP1 is within PV limits, it is copied to PVAUTO.

- FilteredP1 is computed as follows:

$$\text{FilteredP1} = \text{FilteredP1}_{\text{LAST}} + (\text{P1} - \text{FilteredP1}_{\text{LAST}}) * \text{Ts} / (\text{Ts} + \text{P1FILTTIME})$$

where:

$\text{FilteredP1}_{\text{LAST}}$ = previous filtered value
 Ts = elapsed time in minutes
 P1FILTTIME = filter lag time in minutes

- Actual input value is stored in P1; the linear or square root converted P1 in EU is stored in P1EU, and the filtered and clamped result is stored in PVAUTO.
- Status of the filtered/clamped value is stored in PVAUTOSTS.
- If P1 is bad (NaN), the block stops filtering and sets PVAUTO to NaN. When P1 returns to good, the block sets $\text{FilteredP1}_{\text{LAST}}$ equal to the new P1EU, and starts filtering again.
- P1FILTTIME may have a value of 0 to 60 minutes (or fractions thereof). Given a single-step change in P1:
 - $\text{FilteredP1} = 63.2\%$ of P1EU after P1FILTTIME.
 - $\text{FilteredP1} = 86.5\%$ of P1EU after $2 * \text{P1FILTTIME}$.
 - $\text{FilteredP1} = 95.0\%$ of P1EU after $3 * \text{P1FILTTIME}$.
 - $\text{FilteredP1} = \text{approximately } 100\%$ of P1EU after $10 * \text{P1FILTTIME}$.

Input clamping

The P1CLAMPOPT parameter is used to clamp a filtered P1 within PV high/low limits (PVEXHILM and PVEXLOLM). If filtering is not configured, then P1CLAMPOPT is used to clamp P1 as follows:

- If P1CLAMPOPT = Enable, the block clamps the filtered P1 to the PV limits and stores the result in PVAUTO. If the filtered input is outside the PV limits:
 - P1 = Actual input value
 - P1STS = Normal
 - PVAUTO = Exceeded limit
 - PVAUTOSTS = Uncertain (because the value was clamped)
 - Appropriate "limit exceeded" flag is set (PVEXHIFL or PVEXLOFL)
- If P1CLAMPOPT = Disable and the filtered P1 is outside the limits, the block sets PVAUTO to Bad. If the filtered input is outside the PV limits:
 - P1 = Actual input value
 - P1STS = Normal
 - PVAUTO = NaN
 - PVAUTOSTS = Bad
 - Appropriate "limit exceeded" flag is set (PVEXHIFL or PVEXLOFL).

Low signal cut off

If you configure PV Characterization as LINEAR or SQUARE ROOT, you can configure a low cut off value to be applied to PVAUTO after filtering and clamping.

If the low cut off value is not NaN (Not-a-Number) and PVAUTO is less than the user configured low cut off value, PVAUTO is set to the PVEULO range value.

If the low cut off value is NaN, no cut off action is applied.

If you configure the PV Characterization as NONE, the low signal cut off function is not applicable.

Output

The DATAACQ block produces an output value (PV) and status (PVSTS) as well as a status flag (PVSTSFL).

PV source selection

PVSOURCE (which may be changed by the operator or user program) provides the following values to specify where the block's output should come from:

- AUTO (Automatic) – indicates that PVAUTO is used as the PV (where PVAUTO contains the clamped and filtered value of P1) and PVSTS tracks PVAUTOSTS.
- MAN (Manual) – indicates that the operator may enter the PV and:
 - sets PVSTS to Manual.
 - rejects any attempts by the operator to store a value that exceeds the PV limits (PVEXHILM and PVEXLOLM).
 - applies no filtering on operator-entered values.
- SUB (Substitution) – indicates that a user program may enter the PV and –
 - sets PVSTS to uncertain
 - if the program attempts to store a value that exceeds the PV limits (PVEXHILM and PVEXLOLM), the value is clamped to the appropriate limit and the "limit exceeded" flag (PVEXHIFL and PVEXLOFL) is set.
 - applies no filtering on program-entered values.

PV status

PV status (PVSTS) may have one of the following values:

- Bad - which means that PV is NaN (Not-a-Number)
- Normal - which means PV is OK.
- Manual - which means that PV is OK, but was stored by an operator.
- Uncertain - which means that PV is OK but was stored by a program.

The following Boolean flags (typically used with Logic and Alarm blocks) also reflect the value of PVSTS:

- PVSTSFL.BAD – if PVSTS = Bad, this flag is on; otherwise it is off.
- PVSTSFL.NORM – if PVSTS = Normal, this flag is on; otherwise it is off.
- PVSTSFL.MAN – if PVSTS = Manual, this flag is ON; otherwise it is off.
- PVSTSFL.UNCER – if PVSTS = Uncertain, this flag is on; otherwise it is off.

Alarm processing

The DATAACQ block may be configured to generate an alarm when PV exceeds one of the following trip points for more than a specified time:

- PV High trip point (PVHIALM.TP) - if PV exceeds this trip point for more than PVHIALM.TM seconds, a PV High alarm is generated and the PV High alarm flag (PVHIALM.FL) is set.

PV High alarming is enabled by setting PVHIALM.TP to a value which is not IEENaN, and disables it by setting PVHIALM.TP = NaN.

PVHIALM.TP must be $\leq$ PVHHALM.TP.

- PV High High trip point (PVHHALM.TP) - if PV exceeds this trip point for more than PVHHALM.TM seconds, a PV High High alarm is generated and the PV High High alarm flag (PVHHALM.FL) is set.

PV High High alarming is enabled by setting PVHHALM.TP to a value which is not IEENaN, and disabled by setting PVHHALM.TP = NaN.

PVHHALM.TP must be $\leq$ PVEUHI.

- PV Low trip point (PVLOALM.TP) - if PV falls below this trip point for more than PVLOALM.TM seconds, a PV Low alarm is generated and the PV Low alarm flag (PVLOALM.FL) is set.

PV Low alarming is enabled by setting PVLOALM.TP to a value which is not IEENaN, and disabled by setting PVLOALM.TP = NaN.

PVLOALM.TP must be $\geq$ PVLLALM.TP.

- PV Low Low trip point (PVLLALM.TP) - if PV falls below this trip point for more than PVLLALM.TM seconds, a PV Low Low alarm is generated and the PV Low Low alarm flag (PVLLALM.FL) is set.

PV Low Low alarming is enabled by setting PVLLALM.TP to a value which is not IEE NaN, and disabled by setting PVLLALM.TP = NaN.

- Positive Rate-of-Change trip point (ROCPOSALM.TP) – The rate-of-change trip point is specified by the user as EUs per minute, and the function block converts this to EUs per 4-second period. If PV changes in a positive direction by more than this amount for two consecutive 4-second periods, the function block will generate a Positive Rate-of-Change alarm and set the Positive Rate-of-Change alarm flag (ROCPOSALM.FL).

Positive Rate-of-Change alarming is enabled by setting ROCPOSALM.TP ≥ 0 , and disabled by setting ROCPOSALM.TP = NaN.



ATTENTION

- The rate-of-change trip point is specified in EUs per minute.
- ROCPOSALM.TP is expressed as a positive number in EUs per minute.

-
- Negative Rate-of-Change trip point (ROCNEGALM.TP) – The Rate-of-Change trip point is specified by the user in EUs per minute, and the function block converts this to EUs per 4-second period. If PV changes in a negative direction by more than this amount for two consecutive periods, the function block will generate a Negative Rate-of-Change alarm and set a Negative Rate-of-Change alarm flag (ROCPOSALM.FL).

Negative Rate-of-Change alarming is enabled by setting ROCNEGALM.TP ≥ 0 , and disabled by setting ROCNEGALM.TP = NaN.



ATTENTION

- The rate-of-change trip point is specified in EUs per minute.
 - ROCNEGALM.TP is expressed as a positive number in EUs per minute.
-

The following parameters also apply to each of the previously specified alarms:

- Alarm Filter Time (PVHIALM.TM, PVHHALM.TM, etc.) – Prevents input spikes from causing alarms. PV will only be alarmed if it consistently exceeds the trip point for more than xxxALM.TM seconds.

If xxxALM.TM = 0, the function block will generate an alarm as soon as PV exceeds the trip point.

Note: This parameter does not apply to the Rate-of-Change alarms (i.e., there is no ROCNEGALM.TM or ROCPOSALM.TM parameter).

- Alarm Deadband Value (PVHIALM.DB, PVHHALM.DB, etc.) – Note that alarm deadband is not supported for Rate-of-Change alarms.

Prevents recurring alarms and returns-to-normal due to a noise when PV is near the trip point. The deadband is applied to the return-to-normal. For example, if PV is in high alarm (PVHIALM.FL = On), it must return to a value of PVHIALM.DB below the high trip point before it is considered “normal”; and if it is in low alarm, it must return to a value of PVLOALM.DB above the low trip point.

- Alarm deadband units (PVHIALM.DBU, PVHHALM.DBU, etc.) - Indicates if the corresponding alarm deadband (xxxALM.DB) is in percent or engineering units.

This parameter does not apply to Rate-of-Change alarms (i.e., there is no ROCNEGALM.DBU or ROCPOSALM.DBU parameter). For Rate-of-Change alarms, the deadband is always expressed in EUs/minute.

- Alarm flag (PVHIALM.FL, PVHHALM.FL, ROCNEGALM.FL, etc.) - Indicates if the corresponding alarm condition exists.
- Alarm priority (PVHIALM.PR, PVHHALM.PR, ROCNEGALM.FL, etc.) - Indicates the relative priority of the alarm.
- Alarm severity (PVHIALM.SV, PVHHALM.SV, ROCNEGALM.SV, etc.) - Indicates the relative severity of the alarm (from 1 to 256).

PV significant-change alarming

If PV is between the high and high-high alarm trip points and continues to rise, the following parameters may be used to reannunciate the high alarm:

- PV High Significant-Change Trip Point (PVHISIGCHG.TP) – reannunciates the high alarm when PV is between the PV high and high-high limits (PVHIALM.TP and PVHHALM.TP) and keeps rising.

For example, consider a temperature input with PVHIALM.TP = 800 degrees, PVHHALM.TP = 850 degrees and PVHISIGCHG.TP = 10 degrees. When the temperature rises to 800 degrees, the PV high alarm is annunciated and, if the temperature continues to rise, the alarm is reannunciated at 810 degrees, 820 degrees, and so on.

- PV High Significant-Change Count (PVHISIGCHG.CT) – which is a count of the number of times PV has exceeded its high significant change trip point. Other blocks and user programs may monitor it.

When PV falls below the high alarm trip point (plus deadband), the count is reset to zero.

Similarly, if PV is between the low and low-low alarm trip points and continues to decrease, the following parameters may be used to reannunciate the low alarm:

- PVLOSIGCHG.TP – the PV Low Significant-Change Trip Point.
- PVLOSIGCHG.CT - the PV Low Significant-Change Count.

Bad PV alarm

The DATAACQ block may be configured to generate a "Bad PV" alarm if PV = NaN (Not a Number).

- The Bad PV alarm priority and severity parameters (BADPVALM.PR and BADPVALM.SV) are configurable.
- Setting BADPVALM.PR to No Action disables alarming.

Data Acquisition Functions

DATAACQ (Data Acquisition) Block

DATAACQ parameters



REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the DATAACQ block.

Pulse Input

Pulse Input Block

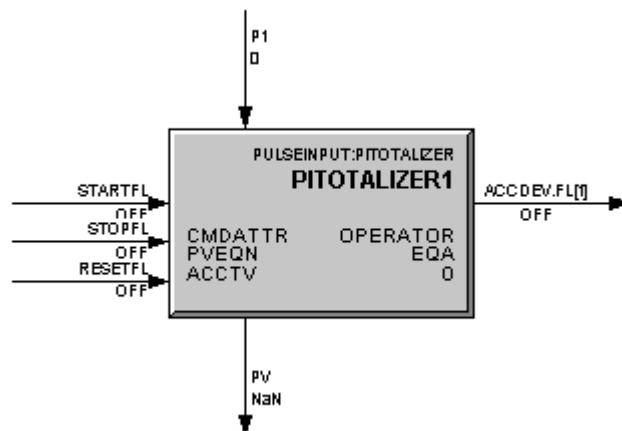
Functional overview

The Pulse Input Totalizer Block provides a flow totalization function to complement a Pulse Input Channel (PICHANNEL) or Pulse Input Counter Fast Cutoff (PICFASTCUROFF) block.

PITOTALIZER Block

Description

The PITOTALIZER block periodically adds the change of the input value (that is, the difference in P1) to an accumulator value (PV). It looks like this graphically:



You specify a target value for the accumulator, and up to four trip points, which are "near" and "nearer to" the target value. The PITOTALIZER block sets status flags to indicate when the accumulator value is near (and nearer to) the user-specified target values. Accumulation proceeds even when the target value is exceeded. An external operator or program command is required to stop the block from further accumulating.

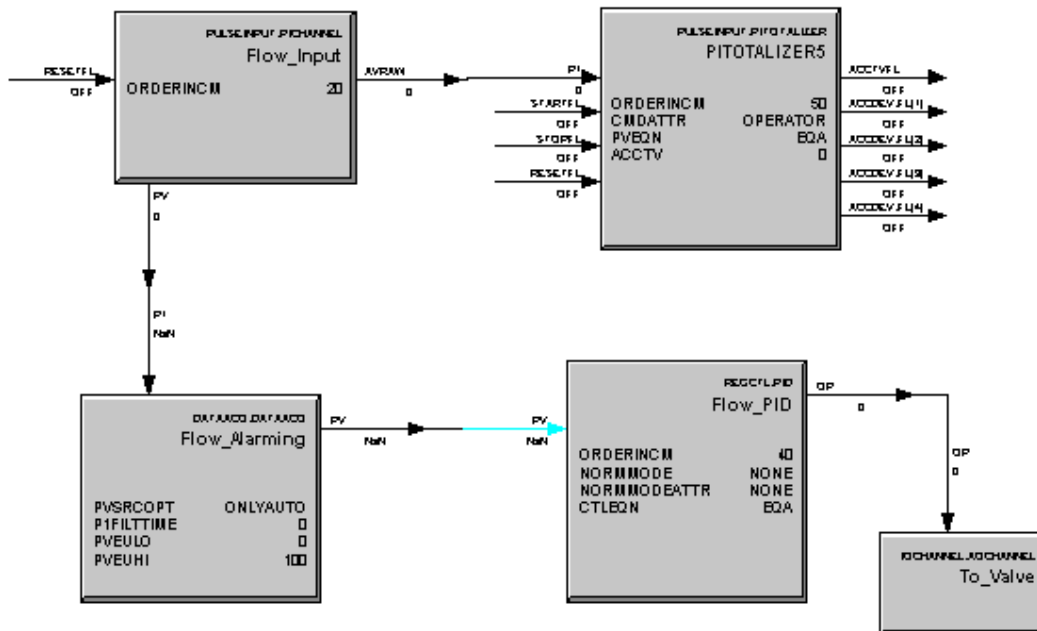
Pulse Input PITOTALIZER Block

Function

The PITOTALIZER block is typically used to accumulate total flows. For situations where the flow transmitter may not be precisely calibrated near the zero-flow value, a zero-flow cutoff feature is provided such that when P1 is below the cutoff value it clamps to zero.

Configuration example

In this example, the PITOTALIZER block is used to accumulate total flow in a flow control loop.



Note the following:

1. Use the PV parameter connection to carry data and status from the PICHANNEL and DATAACQ blocks 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 required.
2. When monitoring, you can use the COMMAND parameter on the PITOTALIZER block to issue Start, Stop or RESET commands. You must configure COMMAND as a monitoring parameter through the block configuration form.

You can also use logic inputs to STARTFL, STOPFL, and RESETFL pins on the block to initiate Start, Stop and Reset commands, respectively.

3. When the accumulated value (P1) reaches the accumulated target value (ACCTV), the accumulated target value flag (ACCTVFL) turns on.
4. In this example, the following values were configured for trip points 1 to 4 through the parameter configuration form based on a configured target value of 100.

Trip Point 1 (ACCDEV.TP[1]) = 10
Trip Point 1 (ACCDEV.TP[2]) = 20
Trip Point 1 (ACCDEV.TP[3]) = 30
Trip Point 1 (ACCDEV.TP[4]) = 40

Based on these configured trip point values, the corresponding accumulated deviation flag turns ON at the following accumulated values:

ACCDEV.FL[1] turns on at P1 = 90
ACCDEV.FL[2] turns on at P1 = 80
ACCDEV.FL[3] turns on at P1 = 70
ACCDEV.FL[4] turns on at P1 = 60

The DATAACQ block generates alarm flags when PV exceeds any of a number of user-specified alarm trip points for longer than the designated time interval.

Input

The PITOTALIZER block requires one input (P1):

- P1 is the value to be accumulated – the input value (P1) must be an integer value.
- P1 must be brought from another block (such as, PICHANNEL.AVRAW or PICFASTCUTOFF.AVRAW)

Outputs

The PITOTALIZER block produces the following outputs:

- The accumulated value (PV) and its status (PVSTS).
- Flags, indicating if the accumulated value has reached the user-specified target value or one of the accumulator deviation trip points (ACCTVFL and ACCDEV.FL [1-4]).

PITOTALIZER states

The PITOTALIZER block has two possible states: Stopped and Running. The STATE parameter identifies the current state and the following parameters may be used to change the state:

- **COMMAND:** The operator or a user program may command the accumulator to Start, Stop, or Reset by storing to the COMMAND parameter. Since COMMAND is a write-only parameter, its displayed value does not reflect the last entered command.

Possible choices are:

- Start – requests the PITOTALIZER to start the accumulation (change STATE to Running).
- Stop – requests the PITOTALIZER to stop the accumulation (change STATE to Stopped).
- Reset – requests the PITOTALIZER to reset the accumulated value (PV) with a user-specified reset value (RESETVAL). STATE will not change; if the accumulator is running, it continues from the reset value.

When the PITOTALIZER receives a reset command, it copies the current value of PV to OLDAV (old accumulation value), and then sets PV equal to RESETVAL. This allows other system functions using the totalized value to reset the PITOTALIZER without losing any "accumulation".

- **CMDATTR:** Specifies who may store to COMMAND (that is, either the operator or a user program). CMDATTR is used to prevent the operator from inadvertently changing the accumulator while it is under program control and allows the operator to override a program.

Possible choices are:

- Operator – only the operator may store to COMMAND.
- Program – only a program may store to COMMAND; the operator may override the program by setting CMDATTR = Operator.
- **STARTFL (Start Flag):** Allows either a Logic block or user-written program to store to COMMAND.
 - Off-to-On transitions cause the PITOTALIZER state to change to Running.
- **STOPFL (Stop Flag):** Allows either a Logic block or user-written program to store to COMMAND.
 - Off-to-On transitions cause the PITOTALIZER state to change to Stop.
- **RESETFL (Reset Flag):** Allows either a Logic block or user-written program to store to COMMAND.
 - Off-to-On transitions cause the PITOTALIZER to be reset.

Accumulator target value

Prior to starting the PITOTALIZER, you may specify a target value for the accumulator (ACCTV). The PITOTALIZER block compares PV with ACCTV on each cycle and sets the target-value-reached flag (ACCTVFL) to ON when the accumulation is complete (that is, when PV is greater than or equal to ACCTV).

Deviation trip points

The PITOTALIZER block provides trip points and flags to signal when the accumulated value is "getting close" to the target value. You may specify as many as four trip points, and the PITOTALIZER block sets a corresponding flag when each trip point is reached. The flags are typically monitored by another function block that can initiate some sort of control action (for example, changing a valve position from full open to trickle when a PITOTALIZER trip point is reached).

Pulse Input PITOTALIZER Block

The trip point values (ACCDEV.TP[1-4]) are expressed as deviations from the target value. The PITOTALIZER block compares the actual deviation (ACCTV - PV) with each trip point, and sets a flag (ACCDEV.FL[1-4]) when the deviation is less than or equal to a trip point. For example, if the user sets ACCTV = 50 and ACCDEV.TP[1] = 10, the PITOTALIZER block sets ACCDEV.FL[1] to ON when PV is greater than or equal to 40.

Equations

PVEQN is a user-configured parameter, which specifies how the PITOTALIZER should handle bad inputs and warm restarts. One of the following equations is specified using PVEQN:

Equation	Bad Input Handling	Warm Restart Handling
A	Stop accumulation while input is bad.	Continue after input turns valid.
B	Use last good value if input is bad.	Continue after input turns valid.
C	Stop if the input is bad and set PV to NaN.	Continue after input turns valid.
D	Stop accumulation while input is bad.	Stop after a warm restart.
E	Use last good value if input is bad.	Stop after a warm restart.
F	Stop if the input is bad and set PV to NaN.	Stop after a warm restart.

The following table summarizes block actions associated with a given PVEQN handling option relative to the accumulator state and the input status. .

If Accumulator is . . .	And Option is . . .	Then, block . . .
Running (STATE = RUNNING) and the input status (P1STS) is BAD	Use zero if input is bad	Sets the input value (P1) to zero, sets PVSTS to Uncertain, and continues the accumulation. When the input status (P1STS) returns to normal, PVSTS remains Uncertain until a Reset command is received.

If Accumulator is . . .	And Option is . . .	Then, block . . .
Running (STATE = RUNNING) and the input status (P1STS) is BAD	Use last good value if input is bad	Sets the input value (P1) to its last good value, sets PVSTS to Uncertain, and continues the accumulation. When the input status (P1STS) returns to normal, PVSTS remains Uncertain until a Reset command is received.
Running (STATE = RUNNING) and the input status (P1STS) is BAD	Stop if the input is bad	Sets the input value (P1) to zero, sets PVSTS to Bad, and stops the accumulation. When the input status (P1STS) returns to normal, PVSTS remains Bad until the operator restarts the accumulation. To restart the accumulator, the operator must estimate the accumulated value, issue a Reset command to establish that value, and then issue a Start command. The last accumulated value before the status went bad is designated as LASTGOOD.
Running (STATE = RUNNING)	Continue after a warm restart	Sets PVSTS to Uncertain and continues accumulation from last value of PV. PVSTS remains Uncertain until a Reset command is received.
Running (STATE = RUNNING)	Stop after a warm restart	Sets the accumulated value (PV) to NaN (Not a Number), sets PVSTS to Bad, and stops the accumulation. The operator must intervene to restart the accumulator.

Accumulated value calculation

For equations A through F, the accumulated value (PV) is calculated as follows:

$$PV_i = PV_{(i-1)} + C1/C2 * (P1_{(i)} - P1_{(i-1)})$$

Where:

- PV_i = PITOTALIZER block output from the current pass
- $PV_{(i-1)}$ = accumulated value at the end of block's last processing pass
- $C1$ = scale factor for P1; used to convert to different engineering units
- $C2$ = count factor in pulses per engineering units
- $P1_{(i)}$ = input value from current pass
- $P_{(i-1)}$ = input value from last pass

Error handling

- PVSTS is set to UNCERTAIN when:
 - The status of the input (P1STS) is Uncertain.
 - The input status is Bad and the "use zero" or "use last good value if input is bad" option is configured (Equation A, B, D, or E).
 - The PITOTALIZER block is in warm restart and the "continue" option is configured (Equation A, B, or C).
- PV is set to NaN (Not a Number) and PVSTS is set to Bad, when:
 - The status of the input (P1STS) is Bad and the "stop if input is bad" option is configured (Equation C or F).
 - The PITOTALIZER block is in warm restart and the "stop" option is configured (Equation D, E, or F).
- When PVSTS is Bad, the PITOTALIZER block sets ACCTVFL and ACCDEV.FL[1-4] to Off.



ATTENTION

When the input status returns to normal, a Reset command is needed to return PVSTS to Normal.

Restart and activation

When a PITOTALIZER block is activated:

- PV is set to NaN (Not a Number).
- PVSTS is set to Bad.
- The accumulator is stopped (that is, STATE = Stopped).

PITOTALIZER parameters



REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the PITOTALIZER block.

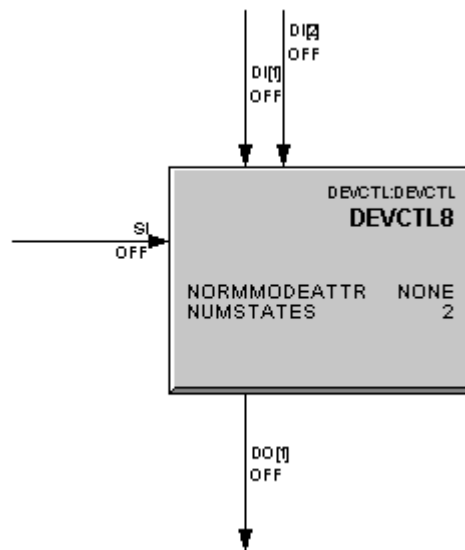
Pulse Input
PITOTALIZER Block

Device Control

DEVCTL (Device Control) Block

Description

The DEVCTL (Device Control) block is a multi-input, multi-output function that provides an interface to discrete devices, such as motors, solenoid valves, and motor-operated valves. This block provides built-in structures for handling interlocks and supports display of the interlock conditions in group, detail and graphic displays. It looks like this graphically.



Each DEVCTL 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 0 to 65535. 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. - 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 either MANual or AUTOMATIC. The default is OPERATOR. - 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. When MODEATTR is configured as Normal, it is actually set to the present value of NORMMODEATTR, if NORMMODEATTR is not None. Selections are NONE, OPERATOR, PROGRAM, and NORMAL. The default selection is NONE. - Enable PV Source Selection (PVSRCOPT) – Lets you enable or disable PV source selection. Check box to enable PV source (PVSOURCE) selection through the companion scroll window. Uncheck box to limit PVSOURCE to only AUTO. The default is enabled or box checked. When PVSRCOPT is ALL or enabled, you can select one of the following to be the source

Configuration Tab	Description
	(PVSOURCE) of the PV (GPV) value.
	– SUB – Provided by an SCM. – MAN – Operator stores value directly. – AUTO – Derived from the parameter PVAUTO (GPVAUTO) representing the assigned state of the actual inputs DI[1..4]. – TRACK – Derived directly for the commanded output state (OP). Use this online when a limit switch has failed, or as a debug mode.
	• Number Of Inputs (NUMDINPTS) – Lets you specify the number of digital inputs to be used with the block. The default is 2. • Number of Outputs (NUMDOUTS) – Lets you specify the number of digital outputs to be used with block. The default is 1. • Number of States (NUMSTATES) – Lets you define the number of settable states as two or three. The default is 2 states.
	– State Names (STATETEXT[0..6]) – Lets you specify a name of up to 12 characters to be used to identify the given state. The defaults are State1 for State 1 Name (STATETEXT[5]), State0 for State 0 Name (STATETEXT[4]), State2 for State 2 Name (STATETEXT[6]), Inbet for In Between (STATETEXT[1]), and Bad for Null (STATETEXT[0]), respectively. State 2 name is only applicable if number of states (NUMSTATES) is three.
Inputs	• Number of Digital Inputs (NUMINPTS) – Same as entry on Main tab. • Inputs 1, 2, 3, 4 (DI[1..4]) – Shows the input combinations to be associated with a given state. A Check in the box for the input represents its ON condition and no Check is for its OFF position. The default State is null (BAD). State 2 selections are only applicable if number of states (NUMSTATES) is three and a name was configured through the Main tab.
Output	• Outputs1,2,3 (DO[1..3] or PO[1..3]) – Lets you specify the output combinations to be associated with the

Configuration Tab	Description
	given state. Check the box for the output to associate its ON condition with the given state (State 1, State 0, State 2) or leave it unchecked for its OFF position. The default is OFF or unchecked. State 2 selections are only applicable if number of states (NUMSTATES) is three. • Safe (SAFEOP) – Lets you select the state that defines the DEVCTL block in a safe state. The default is S0 (State 0). State 2 (S2) selection is only applicable if number of states (NUMSTATES) is three. • Pulse Output (POCONNECTED[1..3]) – Lets you specify whether a given pulse output is to be enabled (ON) or not (OFF). A Check equals ON. When enabled, the corresponding output (PO[x]) pin will be exposed on the block. The default is no pulse outputs configured. The selectable outputs depend upon the configured number of outputs. • Pulsewidth (PULSEWIDTH[1..3]) – Lets you specify the width of a given output pulse as a value between 0.000 to 60 seconds. This is only configurable when the corresponding output is configured as a pulse output (POCONNECTED[x] = ON). The default value is 1 second for all configured pulse outputs. • Momentary State (MOMSTATE) – Lets you specify a given state or states operation as being momentary. See the Momentary State section for this block for more information. The default is NONE. No state is momentary. Note that Safe state (SAFEOP) can not be configured as Momentary state. The Seal-In Option and Momentary State are mutually exclusive. If Momentary state is not None, Seal-In Option will not be configurable. If Seal-In is enabled, Momentary state will not be configurable. • Seal-In Option (SEALOPT) – Lets you specify whether the Seal-In Option is to be enabled or disabled. See the Seal-In Option section for this block for information about this option. The default is an unchecked box or disabled. To enable the Seal-In Option, the Momentary state must be None. When the Seal-In Option is enabled, the Momentary State selection becomes void. • Enable Output Initialization (INITOPOPT) – Lets you

Configuration Tab	Description
	specify if the Output Initialization is to be enabled or disabled. If it is enabled, OP is set to SafeOP in initialization, if there is no active interlock, and the device is not in Local Manual condition. If disabled, INITOPOPT will not affect initialization. • Enable Permissive and Override Interlock Bypassing (BYPPERM) – Lets you specify if operators are permitted to bypass the Permissive and Override Interlocks or not. The default is Disabled (unchecked) or OFF. An operator cannot set or reset the BYPASS parameter. • Bypass Permissive and Override Interlocks (BYPASS) – When BYPPERM is ON, lets you change OP regardless of the state of the Override interlocks, if BYPASS is set ON. This does not affect the Safety Override Interlock (SI). When you reset the BYPASS parameter to OFF, any existing Override Interlocks (OI[0..2]) take effect immediately. The default is OFF (unchecked). Operator cannot bypass override interlocks to change OP.
Maintenance	• Enable Accumulation of Statistics (MAINTOPT) – Lets you specify if the collection of Maintenance Statistics for the DEVCTL block is to be enabled or not. When enabled or box checked, you can specify the maximum number of transitions of PV to each state (MAXTRANS[0..2]) and the maximum number of hours of PV accumulated in each state (MAXTIME[0..2]) for comparison purposes only. The default is OFF or box unchecked. The maintenance statistics are not collected. If statistics are collected, you can configure the following parameters to appear on the DEVCTL block during monitoring. An operator can only reset statistics while the block is red tagged, but a user program or another FB can turn ON the RESETFL parameter to reset statistics anytime. – NUMTRANS[0..2] – Accumulated number of transitions of PV to each state, since the last statistics reset. – NUMSIOVRD – Accumulated number of safety interlock trips that result in OP changing state, since the last statistics reset.

Configuration Tab	Description
SCM	• STATETIME[0..2] – Accumulated time of PV in each state, since the last statistics reset. • 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 • Abnormal State Options – 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. You can choose the NONE or SAFEOP selection for any of the previous options. If you select SAFEOP, the OPREQ is automatically set to the SAFEOP state and OPTYPE to default. You should set STOPOPT and/or HOLDOPT to NONE, if Stopping and/or Holding requires sequencing action. In this case, execute a STOP and/or HOLD HANDLER as part of the SCM. 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
Alarms	• PV Alarming – The following alarms are configurable to represent disagreements between the commanded state (OP) and the feedback state (PV). These alarms are disabled if there are no inputs or outputs. – Command Disagree (CMDDISALM.FL): This alarm is generated when the commanded output state (OP) changes and the feedback state (PV) does not change to the same state within the specified feedback time. This alarm returns to normal when the PV state becomes the same as the OP state. This alarm does not apply for momentary commanded states. – Command Fail (CMDFALALM.FL): This alarm checks to see if the PV state changed from its original state to any other state within a specified feedback time after the OP state is commanded. For slow responding devices, absence of this alarm indicates that the device responded to the command, even if it has not yet moved to its commanded position. – Uncommanded Change (UNCMDALM.FL): This alarm is configured in conjunction with the Command Disagree alarm function. This alarm is generated, if an OP state has not been commanded and the PV state changes for any reason except BADPV. – Bad PV (BADPVALM.FL): This alarm is generated whenever PV is detected in the Null state. The Null state can result from a BadPV condition for an input provided by a source block, or because input combinations represent a Null state as defined by the DIPVMAP[0..15] parameter. • Command Disagree – Lets you configure the following parameters for this alarm. – Time to State0 (or assigned State Name) (CMDDISALM.TM[0]): Lets you set the feedback time for State 0 in range of 0 to 1000 seconds. A time of 0 disables the alarm. The default setting is 0. – Time to State 1 (or assigned State Name) (CMDDISALM.TM[1]): Lets you set the feedback time for State 1 in range of 0 to 1000 seconds. A time of 0 disables the alarm. The default setting is 0.

Configuration Tab	Description
	– Time to State 2 (or assigned State Name) (CMDDISALM.TM[2]): Lets you set the feedback time for State 2 in range of 0 to 1000 seconds. A time of 0 disables the alarm. The default setting is 0. This can only be configured if the number of states is 3. – Priority (CMDDISALM.PR): Lets you select the priority level as NONE, JOURNAL, LOW, HIGH, or URGENT. The default setting is LOW. – Severity (CMDDISALM.SV): Lets you set the relative severity of the alarm on a scale of 0 to 15. Where 15 is the most sever. The default setting is 0. • Command Fail – Lets you configure the following parameters for the command fail alarm. – Time to State0 (or assigned State Name) (CMDFALALM.TM[0]): Lets you set the feedback time for State 0 in range of 0 to 1000 seconds. A time of 0 disables the alarm. The default setting is 0. This value must be less than the value set for CMDDISALM.TM[0]. – Time to State 1 (or assigned State Name) (CMDFALALM.TM[1]): Lets you set the feedback time for State 1 in range of 0 to 1000 seconds. A time of 0 disables the alarm. The default setting is 0. This value must be less than the value set for CMDDISALM.TM[1]. – Time to State 2 (or assigned State Name) (CMDFALALM.TM[2]): Lets you set the feedback time for State 2 in range of 0 to 1000 seconds. A time of 0 disables the alarm. The default setting is 0. This value must be less than the value set for CMDDISALM.TM[2]. This can only be configured if the number of states is 3. – Priority (CMDFALALM.PR): Lets you select the priority level as NONE, JOURNAL, LOW, HIGH, or URGENT. The default setting is LOW. – Severity (CMDFALALM.SV): Lets you set the relative severity of the alarm on a scale of 0 to 15. Where 15 is the most sever. The default setting is 0. • Uncommanded Change - Lets you configure the

Configuration Tab	Description
	following parameters for this alarm. – Priority (UNCMDALM.PR): Lets you select the priority level as NONE, JOURNAL, LOW, HIGH, or URGENT. The default setting is LOW. – Severity (UNCMDALM.SV): Lets you set the relative severity of the alarm on a scale of 0 to 15. Where 15 is the most severe. The default setting is 0. • Bad PV - Lets you configure the following parameters for this alarm. – Priority (BADPVALM.PR): Lets you select the priority level as NONE, JOURNAL, LOW, HIGH, or URGENT. The default setting is LOW. – Severity (BADPVALM.SV): Lets you set the relative severity of the alarm on a scale of 0 to 15. Where 15 is the most severe. The default setting is 0. • Override Alarms – The following alarms are configurable to represent override interlock conditions. – Safety Override Interlock (SIALM.FL): This alarm may be generated when the safety override interlock (SI) occurs, and has caused an OP state change. – Override Interlock (OIALM[0..2].FL): This alarm may be generated when an override interlock (OI[0..2]) occurs, and has caused an OP state change. – Off Normal Condition (OFFNRMALM.FL): If an interlock bypass becomes active when OPREQ is not Null, the OPREQ is transmitted to OP immediately upon activation of the bypass parameter. If bypass is activated after an interlock has been initiated, the OP and OFFNRMALM.FL will be corrected within one scan. • Override Alarms – Lets you configure the following parameters for the safety override interlock alarm. – Option (SIALM.OPT): Lets you specify whether the safety override interlock alarm is enabled or not. The default setting is DISABLED. – Priority (SIALM.PR): Lets you select the priority level as NONE, JOURNAL, LOW, HIGH, or URGENT. The

Configuration Tab	Description
	default setting is LOW.
	– Severity (SIALM.SV): Lets you set the relative severity of the alarm on a scale of 0 to 15. Where 15 is the most severe. The default setting is 0.
	• State 0 Override Interlock Alarm – Lets you configure the following parameters for this alarm. – Option (OIALM[0].OPT): Lets you specify whether the State 0 override interlock alarm is enabled or not. The default setting is DISABLED. – Priority (OIALM[0].PR): Lets you select the priority level as NONE, JOURNAL, LOW, HIGH, or URGENT. The default setting is LOW. – Severity (OIALM[0].SV): Lets you set the relative severity of the alarm on a scale of 0 to 15. Where 15 is the most severe. The default setting is 0.
	• State 1 Override Interlock Alarm – Lets you configure the following parameters for this alarm. – Option (OIALM[1].OPT): Lets you specify whether the State 0 override interlock alarm is enabled or not. The default setting is DISABLED. – Priority (OIALM[1].PR): Lets you select the priority level as NONE, JOURNAL, LOW, HIGH, or URGENT. The default setting is LOW. – Severity (OIALM[1].SV): Lets you set the relative severity of the alarm on a scale of 0 to 15. Where 15 is the most severe. The default setting is 0.
	• State 2 Override Interlock Alarm – Lets you configure the following parameters for this alarm. – Option (OIALM[2].OPT): Lets you specify whether the State 0 override interlock alarm is enabled or not. The default setting is DISABLED. – Priority (OIALM[2].PR): Lets you select the priority level as NONE, JOURNAL, LOW, HIGH, or URGENT. The default setting is LOW. – Severity (OIALM[2].SV): Lets you set the relative severity of the alarm on a scale of 0 to 15. Where 15

Configuration Tab	Description
	is the most severe. The default setting is 0. • Off Normal Condition – Lets you configure the following parameters for this alarm. – Priority (OFFNRMALM.PR): Lets you select the priority level as NONE, JOURNAL, LOW, HIGH, or URGENT. The default setting is LOW. – Severity (OFFNRMALM.SV): Lets you set the relative severity of the alarm on a scale of 0 to 15. Where 15 is the most severe. The default setting is 0.
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.

Function

The DEVCTL block allows manipulation of sets of digital outputs and interprets corresponding feedback of digital inputs. Operation consists of transmitting the commands represented by the state parameter OP (the Commanded Output State), monitoring PV (the Current Active State), and producing alarms based on various configurations such as whether or not the PV has achieved the state commanded in OP.



ATTENTION

Please refer to the *SCM Interface and CM interaction* section for more information on the DEVCTL block's batch level 1 driver interface function.

Device Control

DEVCTL (Device Control) Block

Figures 39 and 40 are a graphic representation of the DEVCTL block's major functions and associated parameters.

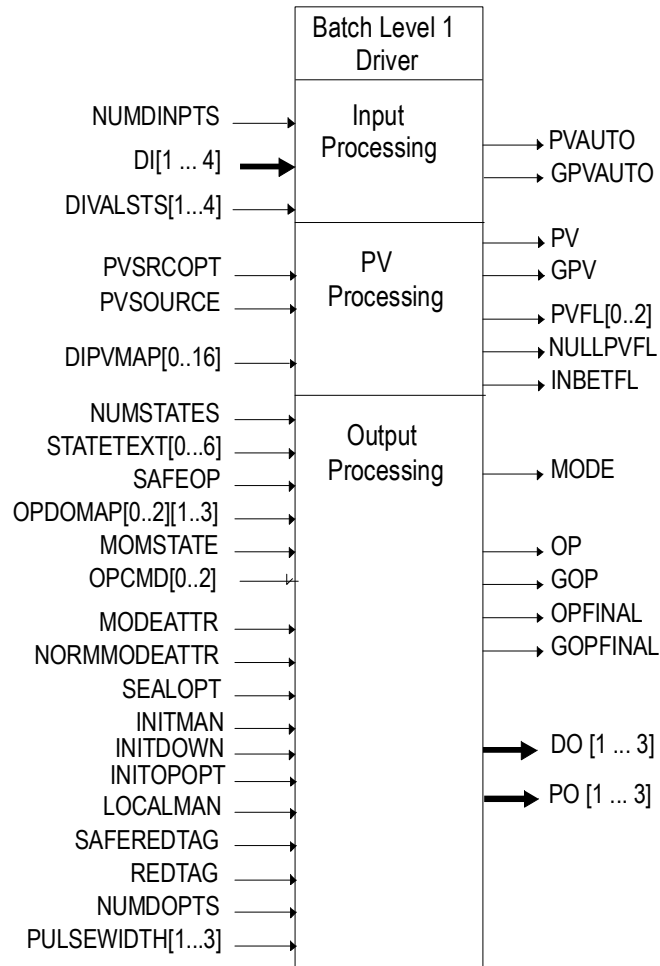


Figure 39 DEVCTL block major functions and parameters - See Figure 40 also.

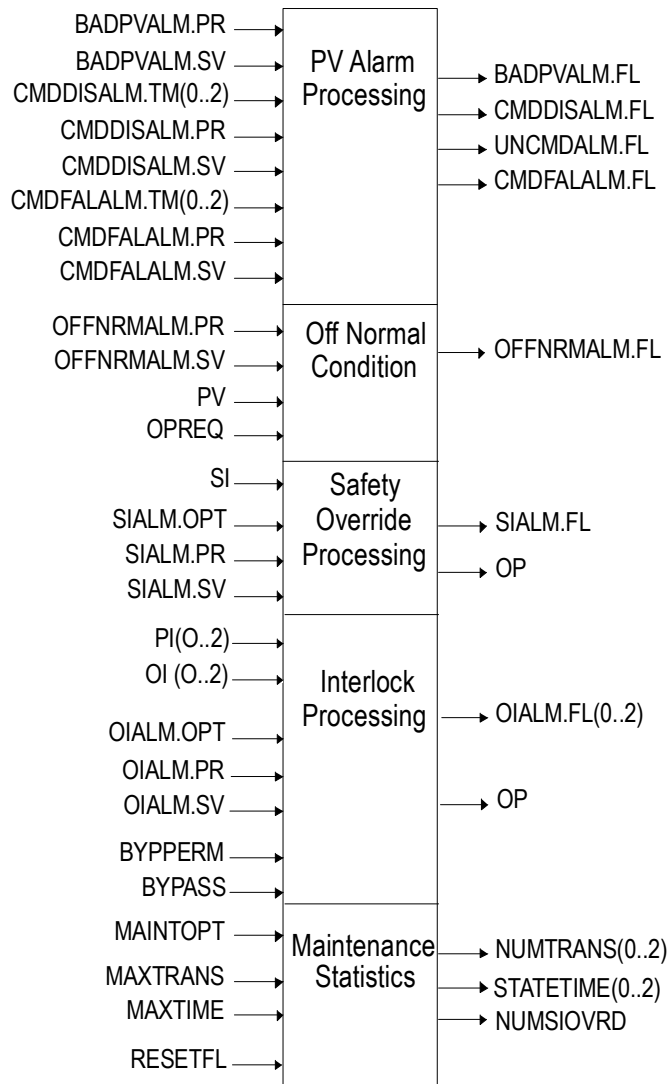


Figure 40 More DEVCTL block major functions and parameters.

Device Control

DEVCTL (Device Control) Block

In summary, the DEVCTL block provides these major features.

- Up to 4 inputs, 3 states, and 3 outputs.
- PV Source Selection, PV has 3 basic states plus in-between and faulty.
- Latched and pulsed outputs.
- Momentary states.
- Initialization, Local Manual and Redtagging.
- BADPV, Command Disagree, Uncommanded Change and Command Fail alarms.
- PV Change of state event.
- Permissive and Override Interlocks for each state.
- Interlock trip alarms.
- Seal In option.
- Maintenance statistics.
- The Safety Interlock enforces the defined safe state.
- Safe State explicitly configured. Can not be momentary.
- Generic State parameters defined as consistent data types.
- Initialization has OPFINAL based configuration.
- Boolean Command option
- Batch level 1 driver option.
- OFF Normal Alarm associated with requested OP.

Configuration examples

- Status Output – Figure 41 and its companion callout description table show a sample configuration that uses a DEVCTL block to command two status outputs. The view in Figure 41 depicts a loaded configuration in Monitoring mode.

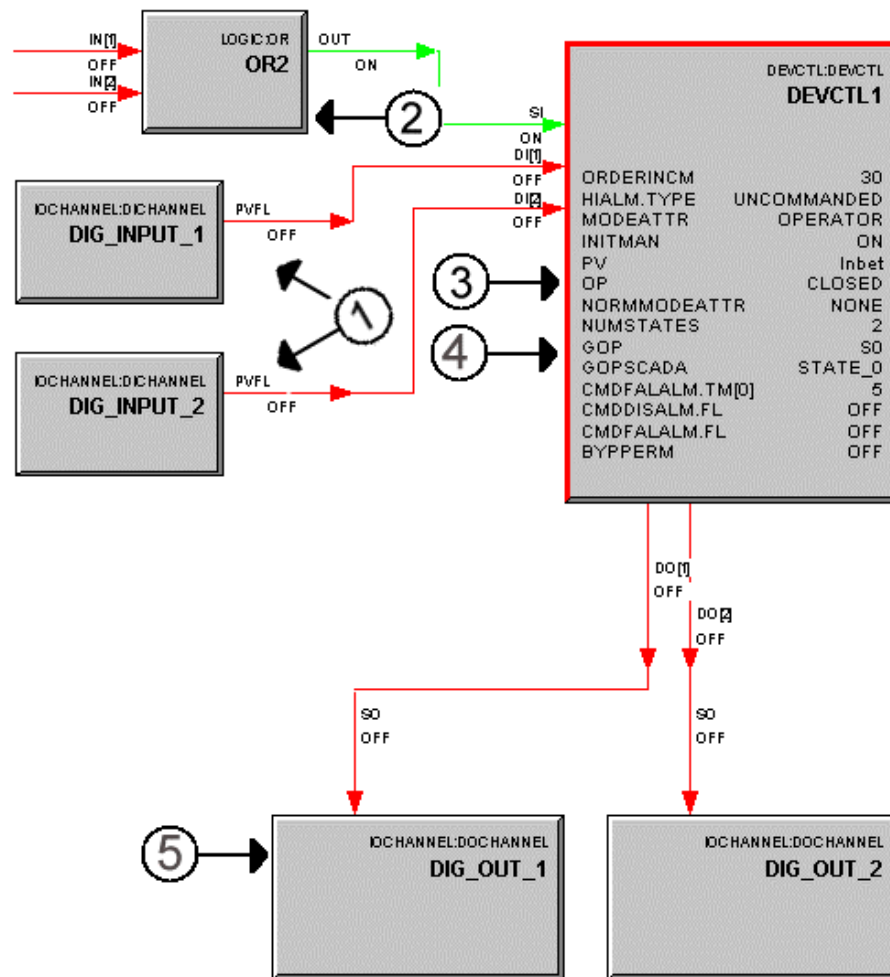


Figure 41 Example of CB configuration using a DEVCTL block to provide two status outputs.

Device Control

DEVCTL (Device Control) Block

The following table includes descriptions of the callouts in Figure 41.

Callout	Description
1	Use the PVVAL parameter connection to carry data from the DICHANNEL block to the DEVCTL block. In device control, the inputs provide the feedback that the commanded action has or has not taken place.
2	You can use an appropriate interlock logic to activate the safety interlock function.
3	You can command the device through the output (OP), which shows the state names you configured for the block through Control Builder.
4	You can have the device commanded by another block or Sequential Control Module through the generic output (GOP), which shows the state as S0 to S2. The GOPSCADA parameter provides a link to Station detail displays and custom schematics to show the state as STATE_0 to STATE_2.
5	Use the BACKCALCIN/BACKCALOUT connection to carry secondary data from the DOC block to the DEVCTL. (Note that the individual BACKCALCIN/BACKCALCOUT connections for each DEVCTL output used are automatically built by Control Builder as implicit/hidden connections.) The secondary data contains this information for DEVCTL blocks. • Initialization request flag – requests continuous initialization. If this flag is set and this block is configured to accept secondary initialization, this block goes to the initialized state and stays there until the flag is reset. • Initialization value – provides continuous and oneshot initialization.

- Pulse Output – Figure 42 and its companion callout description table show a sample configuration that uses a DEVCTL block to command two on pulse outputs. The view in Figure 42 depicts a loaded configuration in Monitoring mode.

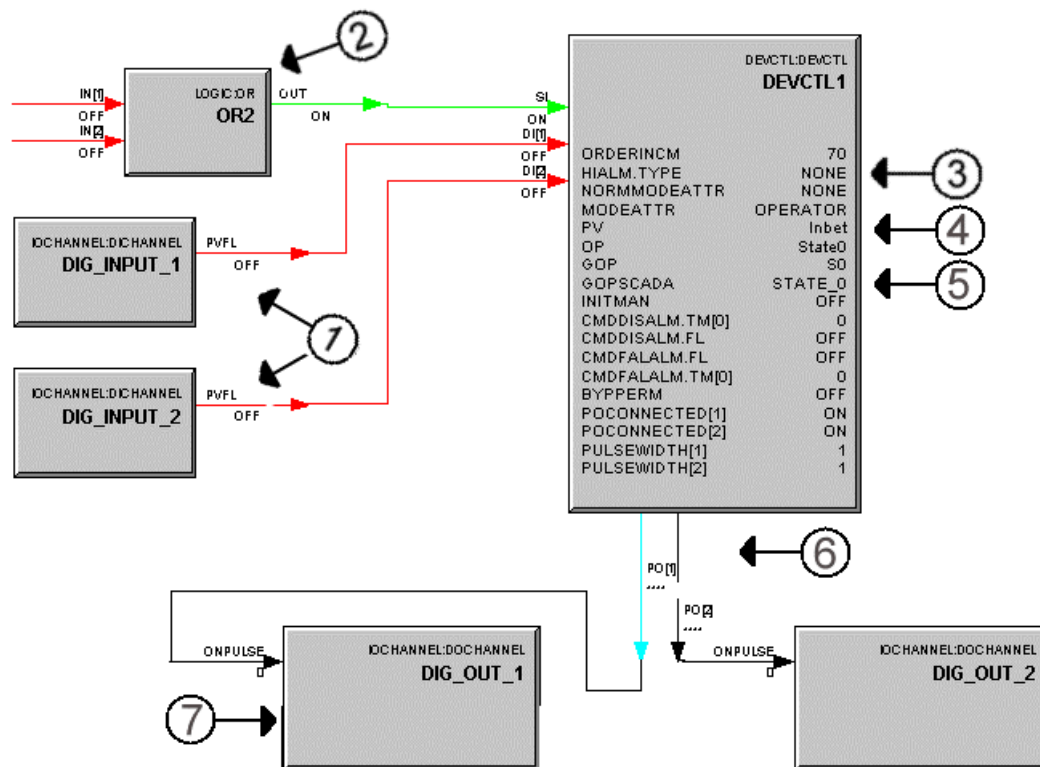


Figure 42 Example of CB configuration using DEVCTL block to provide two on pulse outputs.

Device Control

DEVCTL (Device Control) Block

The following table includes descriptions of the callouts in Figure 42.

Callout	Description
1	Use the PVVAL parameter connection to carry data and status from the DICHANNEL block to the DEVCTL block. In device control, the inputs provide the feedback that the commanded action has or has not taken place.
2	You can use an appropriate interlock logic to activate the safety interlock function.
3	The DEVCTL block is in a CMDDISAGREE alarm state because its input states do not agree with the input conditions consistent with the commanded state.
4	You can command the device through the output (OP), which shows the state names you configured for the block through Control Builder.
5	You can have the device commanded by another block or Sequential Control Module through the generic output (GOP), which shows the state as S0 to S2. The GOPSCADA parameter provides a link to Station detail displays and custom schematics to show the state as STATE_0 to STATE_2.
6	For the DEVCTL block to provide pulse outputs, you must: • Enable pulse outputs through the parameter Pulse Output 1, 2, 3 (check box checked), under Output tab on the parameter configuration form, and configure the desired pulse width for the enabled pulse outputs. • Once the Pulse Output is configured, the PO[x] pin will be automatically exposed on the DEVCTL block symbol. Wire PO[X] pin of DEVCTL block to ONPULSE pin on the corresponding DOCHANNEL block.
7	For the DOCHANNEL block to handle pulse outputs, you must: • configure the block to have visible ONPULSE input pin through the block configuration form. This pin also displays the remaining pulse time upon a state change. • double-click DOWTYPE parameter on block and change selection to ONPULSE.

Inputs

May have from 0 to 4 inputs (DI [1 .. 4]). Each input is a Boolean value, which may represent the state of any other block output or a field DICHANNEL (Digital Input Channel) block.

- The NUMDINPTS parameter determines how many DI inputs are active. When this parameter is 0 (zero), the other inputs and PV parameters have no meaning.
- Depending upon what is providing the input, the DI[1..4] connection may be identified as a DIX[1..4] connection. The DIX is an internal parameter that is not visible to users. It is equivalent to a DI parameter with status (BadPV). The Control Builder determines whether an input is DI or DIX when it is created. The internal DIXCONNECTED[1..4] parameter is set to ON, if the corresponding DI[1..4] input is connected as a DIX type.



ATTENTION

You must assign inputs and outputs in consecutive order without gaps. For example, if the block is to have two inputs and two outputs, you must assign the inputs to DI[1] and DI[2] and the outputs to DO[1] and DO[2]. Assigning inputs and outputs in any other combination, results in an invalid block configuration.

Outputs

May have from 0 to 3 outputs (DO [1 .. 3]). Each output may be Boolean or pulsed (On Pulse or Off Pulse). Each output is a Boolean value, which may be connected to any other block parameter or to a field DOCHANNEL (Digital Output Channel) block.

- An output to any connection except to a DOCHANNEL block is a Boolean output (DO [1 .. 3]) only.
- The DOCHANNEL (DOC) block may connect three different inputs to a DEVCTL block (output). However, only one of these inputs can be connected for any single DOC.
 - DOC.SO may be connected to DO [1 .. 3].
 - DOC.ONPULSE may be connected to pulsed outputs PO [1 .. 3].
 - DOC.OFFPULSE may be connected to pulsed outputs PO [1 .. 3].
- The NUMDOUTS parameter determines how many DO/PO outputs are active. When this parameter is 0 (zero), the other outputs and OP parameters have no meaning.

Device Control

DEVCTL (Device Control) Block

- The internal POCONNECTED[1..3] parameter is set to ON when the respective PO[1..3] is configured as a block pin and connected to a DOC.ONPULSE or DOC.OFFPULSE input. This lets the DEVCTL block know what output is used.
- You can configure an individual PULSEWIDTH for each PO[1..3]. The setting range is between 0.000 and 60 seconds with a resolution of 1 millisecond.
 - The DOCHANNEL block determines the actual pulsewidth resolution and accuracy based on its execution rate. It always rounds the configured pulsewidth value up consistent with its own execution rate. For example, if the execution rate of the DOCHANNEL block is 125 milliseconds and the configured PULSEWIDTH value is 450 milliseconds (.45 seconds), the actual pulse time output would be 500 milliseconds, which is the next highest multiple of 125 milliseconds.
 - A PULSEWIDTH value of 0 is a special case. If a 0 pulse is sent to ONPULSE or OFFPULSE, the DOCHANNEL block immediately turns OFF any existing pulse.



ATTENTION

- For pulsed outputs (ONPULSE and OFFPULSE), only one of these inputs may be connected for any one DOCHANNEL block.
 - You may only connect a DO[1..3] or a PO[1..3] for any one output, but not both.
-



CAUTION

In a peer-to-peer strategy, always locate the DOCHANNEL block associated with a DEVCTL block output in the same CEE. If you use a parameter connector to connect the DEVCTL block output to a DOCHANNEL block included in a CM in another CEE, be aware that this configuration may cause “bumps” in the output.

States

A “state” represents the present condition of a device. For example, Run and Stop could represent the “states” of a two-state motor, with Stop being the safe or failsafe state. A three-state motor could have the states of Run, Stop, and Reverse. Open and Close could represent the states of a valve. You can configure your given device states through the State Assignments tab of the DEVCTL block configuration form. This lets you associate states with Boolean combinations of process feedback inputs from the field. Each input combination is assign to a specific state. The PV parameter represents the present state of a device in the DEVCTL block.

You can also configure the number of output states as two or three through the State Assignments tab. These output states are mapped to specific combinations of digital outputs. These outputs command the field device to the associated state, such as Run or Stop. The OP parameter represents the commanded state or the device state commanded by an operator. The DEVCTL block transmits the OP, monitors the PV, and produces alarms based on the State Assignment configurations, which represent whether or not the process feedback has achieved the state commanded in OP.

State parameters and descriptors

The DEVCTL block includes these two sets of parameters for state associations.

- State Parameters
 - PV
 - PVAUTO
 - OP
 - OPFINAL
- Generic State Parameters
 - GPV (generic version of PV)
 - GPVAUTO (generic version of PVAUTO)
 - GOP (generic version of OP)
 - GOPFINAL (generic version of OPFINAL)

The State Parameters are an enumeration with an assigned text name, which tracks the names assigned to STATETEXT[0..6] parameter. An operator can use these parameters.

The Generic State Parameters are consistent data types, which can be compared with each other through the enumeration GENSTAT_ENM. The generic state enumerations are:

- Null – Stands for Bad Value.
- Inbet – Represents an in between state and could be designated MOVPPV for moving PV.
- Active – Refers to momentary state settings for a two-state device. It is defined as not SAFEOP of State 0 and State 1 and illegal for 3 state configuration. For example, if SAFEOP is designated as State 0 (S0), State 1 (S1) is considered the active state. If S1 is the SAFEOP, S0 is considered the active state. An external FB could issue the

Device Control

DEVCTL (Device Control) Block

Active command to GOP and the state would be set to the not SAFEOP of S0 or S1, accordingly.

- Safe – Stands for SAFEOP. If an external FB issues a Safe command to GOP, the internal value is set to the designated SAFEOP.
- S0 –Represents settable output State 0.
- S1 – Represents settable output State 1.
- S2 – Represents settable output State 2.

The STATETEXT[0..6] parameter is an array of 12-character string parameters corresponding to the members of the generic state enumerations listed above. This allows the various State Parameters to have labels unique to each state. You can assign your own name for a given STATETEXT[0..6] descriptor through the State Name field on the State Assignments tab in the block configuration form. The following table lists the default name for a given STATETEXT[0..6] and shows the corresponding generic states enumeration.

If STATETEXT is . . .	Then, default name is. . .	And, GENSTAT_ENM is. . .
STATETEXT[0]	Bad *	Null
STATETEXT[1]	Inbet *	Inbet
STATETEXT[2]	Active	Active
STATETEXT[3]	Safe	Safe
STATETEXT[4]	State0 *	S0
STATETEXT[5]	State1 *	S1
STATETEXT[6]	State2 *	S2

* These names are configurable through the State Assignments tab.

Two-State motor input example

You can represent a simple two-state motor with one input (DI[1]). In this case, when the input is ON, the motor is in the Run mode. When the input is OFF, the motor is stopped. The following table summarizes the input state combinations as well as the configured state names and the related GENSTAT_ENM and DIPVMAP[0..15]for reference.

DI[1] Input State	Configured State Name	Related DIPVMAP and GENSTAT
0	Stop	S0
1	Run	S1
(bad) *	Fault	Null

* The (bad) state refers to the status that is part of a DIX type input. This represents a failure in the associated DICHANNEL block. If the DIXCONNECTED[1..4] parameter for this input is OFF, this state cannot occur.

Valve input example

You can represent a valve as a device with two digital inputs. One input could represent the contact at the Open end of the valve travel, and the other could represent the contact at the Closed end of the valve travel. The following table summarizes the input state combinations as well as the configured state names and the related GENSTAT_ENM and DIPVMAP[0..15] for reference.

Input States		Configured State Name	Related DIPVMAP and GENSTAT
DI[1]	DI[2]		
0	0	Moving	Inbet
1	0	Open	SO
0	1	Closed	S1
1	1	Fault	Null
(bad) *	X	Fault	Null
X	(bad) *	Fault	Null

* The (bad) state refers to the status that is part of a DIX type input. This represents a failure in the associated DICHANNEL block. If the DIXCONNECTED[1..4] parameter for this input is OFF, this state cannot occur.

Two-Input motor example

You are free to assign states to input combinations as desired. You can assign the same state to more than one combination of input. In this example, the motor is considered to be in Run whenever the DI[2] input is ON (1). The following table summarizes the input state combinations as well as the configured state names and the related GENSTAT_ENM and DIPVMAP[0..15] for reference.

Input States		Configured State Name	Related DIPVMAP and GENSTAT
DI[1]	DI[2]		
0	0	Fault	Null
1	0	Stop	S0
0	1	Run	S1
1	1	Run	S1
(bad) *	X	Fault	Null
X	(bad) *	Fault	Null

* The (bad) state refers to the status that is part of a DIX type input. This represents a failure in the associated DICHANNEL block. If the DIXCONNECTED[1..4] parameter for this input is OFF, this state cannot occur.

Reversible motor input example

You can use all three states to assign input combinations to represent a reversible motor. The following table summarizes the input state combinations as well as the configured state names and the related GENSTAT_ENM and DIPVMAP[0..15] for reference.

Input States		Configured State Name	Related DIPVMAP and GENSTAT
DI[1]	DI[2]		
0	0	Stop	S0
1	0	Run	S1
0	1	Reverse	S2
1	1	Fault	Null
(bad) *	X	Fault	Null
X	(bad) *	Fault	Null

* The (bad) state refers to the status that is part of a DIX type input. This represents a failure in the associated DICHANNEL block. If the DIXCONNECTED[1..4] parameter for this input is OFF, this state cannot occur.

Four-Input two-valve example

You can have up to four inputs with 16 possible state assignments. This can represent an application with two valves that each have open and close contacts. In this case, DI[1] and DI[2] represent open and close states for valve #1, and DI[3] and DI[4] represent open and close states for valve #2. The following table summarizes the input state combinations as well as the configured state names and the related GENSTAT_ENM and DIPVMAP[0..15] for reference.

Input States				Configured State Name	Related DIPVMAP and GENSTAT
DI[1]	DI[2]	DI[3]	DI[4]		
0	0	0	0	Fault	Null
1	0	0	0	Fault	Null
0	1	0	0	Valve Moving	Inbet

Device Control
DEVCTL (Device Control) Block

Input States				Configured State	Related DIPVMAP and
1	1	0	0	Fault	Null
0	0	1	0	Valve Moving	Inbet
1	0	1	0	Valve1 Open	S1
0	1	1	0	Val1&2 Close	S0
1	1	1	0	Fault	Null
0	0	0	1	Fault	Null
1	0	0	1	Fault	Null
1	0	1	0	Valve2 Open	S2
1	1	0	1	Fault	Null
0	0	1	1	Fault	Null
1	0	1	1	Fault	Null
0	1	1	1	Fault	Null
1	1	1	1	Fault	Null
X	X	X	(bad) *	Fault	Null
X	X	(bad) *	X	Fault	Null
X	(bad) *	X	X	Fault	Null
(bad) *	X	X	X	Fault	Null

* The (bad) state refers to the status that is part of a DIX type input. This represents a failure in the associated DICHANNEL block. If the DIXCONNECTED[1..4] parameter for this input is OFF, this state cannot occur.

DI to PV state map

The DIPVMAP[0..15] is the parameter array used to make the actual state assignments for PVAUTO as summarized in the tables for the previous examples. Each element of DIPVMAP[0..15] represents one combination of the input values. DIPVMAP[0..15] is same type of STATTEXT[0..6]. It cannot be assigned to the Active and Safe GENSTAT enumerations and the default state is Bad.

Two-State motor with latched output example

You can command a latched two-state motor through a single output. In this example, if the commanded state is S0 or Stop, the single output DO[1] is set to 0 (OFF). If the commanded state is S1 or Run, the single output DO[1] is set to 1 (ON). There is no “bad” state for outputs. The following table summarizes the output state combinations as well as the configured state names and the related GENSTAT_ENM for reference.

Configured State Name	Related GENSTAT	Output DO[1] State
Stop	S0	0
Run	S1	1

Valve Output Example

You can use two outputs to open and close a valve. Since there are more combinations of outputs than there are states available, you must make unique output state assignments. For this example, when Close is commanded, DO[1] only is set. When Open is commanded, DO[2] only is set. There is no way to command the other possible combinations. The following table summarizes the output state combinations as well as the configured state names and the related GENSTAT_ENM for reference.

Configured State Name	Related GENSTAT	Output States	
		DO[1]	DO[2]
Close	S0	1	0
Open	S1	0	1

Three-State Motor output examples

You can use two outputs to provide different outputs for three states. Of course, the NUMSTATES parameter is set to three. The following table summarizes the output state combinations as well as the configured state names and the related GENSTAT_ENM for reference.

Configured State Name	Related GENSTAT	Output States	
		DO[1]	DO[2]
Stop	S0	0	0
Run	S1	1	0
Reverse	S2	0	1

Since you can assign outputs to any state. It is possible to have more than one output on for a given state. The following table summarizes the output state combinations as well as the configured state names and the related GENSTAT_ENM for reference.

Configured State Name	Related GENSTAT	Output States	
		DO[1]	DO[2]
Stop	S0	0	0
Run	S1	1	0
Reverse	S2	1	1

If you have three outputs instead of two, there are eight possible combinations that can be assigned to three states. The following table summarizes the output state combinations as well as the configured state names and the related GENSTAT_ENM for reference.

Configured State Name	Related GENSTAT	Output States		
		DO[1]	DO[2]	DO[3]
Stop	S0	1	0	0
Run	S1	0	1	0
Reverse	S2	0	0	1



ATTENTION

Output combinations are not necessarily the same as the input feedback combinations for the same state.

Mode and mode attribute

- Mode (MODE) is fixed at MANual. The Normal Mode (NORMMODE) parameter is also fixed at MANual.
- Mode Attribute (MODEATTR) – determines where state commands to the DEVCTL block may originate – that is, who may set the commanded output state (OP), as follows:
 - OPERATOR = only the operator may command the output state.
 - PROGRAM = only other function blocks (such as Logic blocks, SCM programs) may command the output state by setting OPREQ.
 - NORMAL = the setting specified by the Normal Mode Attribute (NORMMODEATTR) is assumed.

Safe output state

The Safe Output State (SAFEOP) parameter defines the default state for certain actions of the DEVCTL block, such as the momentary output state and OP initialization. SAFEOP can be assigned to any of the settable states of the block (that is, those states to which parameter OP may be assigned). The default for SAFEOP is State 0.

- When NUMSTATES = 2, then State 2 is illegal for SAFEOP.
- SAFEOP may not be assigned to a state, which is already configured as momentary.
- When OP or GOP is commanded to safe, the effective value of OP (GOP) is set equal to SAFEOP.

Momentary state

The Momentary State (MOMSTATE) parameter lets you configure states as being momentary. This is like providing push-button operation. When the operator commands a new output state (OP), the selected momentary state is active for only a fixed time or as long as the operator request the value. Once the operator ceases requesting the value and the internal timeout occurs, the DEVCTL block returns to the Safe Output State (SAFEOP).

Device Control

DEVCTL (Device Control) Block

The following table summarizes the MOMSTATE parameter selections.

If Momentary State selection is . . .	Then,
NONE	No state is momentary.
STATE_0	State 0 is momentary and it must not be configured as the safe output.
STATE_1	State 1 is momentary and it must not be configured as the safe output.
STATE_0AND1	Both State 0 and State 1 are momentary and neither one must be configured as the safe output. This can only be selected if the number of states (NUMSTATES) is three.
STATE_2	State 2 is momentary and it must not be configured as the safe output. This can only be selected if the number of states (NUMSTATES) is three.
STATE_0AND2	Both State 0 and State 2 are momentary and neither one must be configured as the safe output. This can only be selected if the number of states (NUMSTATES) is three.
STATE_1AND2	Both State 1 and State 2 are momentary and neither one must be configured as the safe output. This can only be selected if the number of states (NUMSTATES) is three.

Local manual

The local manual (LOCALMAN) parameter is an input flag to support an interface to a local HAND/OFF/AUTO (also called HAND/OFF/REMOTE) switch on the field device. You can hard wire the AUTO position of the switch to a digital input. You can then have the state of the digital input stored to the LOCALMAN pin added to the DEVCTL block through a DICHANNEL connection. Since the control system may not have control over the field device when the HAND/OFF/AUTO switch is not in the AUTO position, the LOCALMAN parameter provides feedback of the switch position.

When the LOCALMAN parameter is ON, the OP state tracks the PV state, if it is a settable state. If PV is in a non-settable state, OP will be set to SAFEOP. This assures that the last commanded state agrees with the present value of the feedback state, when the LOCALMAN is turned OFF. You cannot directly command the OP (GOP) while the LOCALMAN is ON.

You can not access LOCALMAN, if the DEVCTL block has no inputs or no outputs connected. Since PV is illegal for no inputs and OP is illegal for no outputs, LOCALMAN has no meaning for these conditions.

Permissive interlocks

PI[0..2] are Permissive Interlocks which are inputs that may be connected to an external function block to determine whether the operator and/or user program are allowed to change the commanded output (OP) of the DEVCTL block to a specific state. Permissive Interlocks themselves never cause OP to change.

- For OP to be changed to the desired state, the corresponding Permissive Interlock parameter must be set to ON.
- The Permissive Interlocks are all defaulted to ON, thereby allowing permission to all the states – they must be individually set to OFF to prevent access to the corresponding OP state.

Safety Override Interlock

The Safety Override Interlock (SI) forces the commanded output state (OP) to the Safe Output State (SAFEOP) when active. No one may command OP to a different state while SI is active.

- SI may be connected to other blocks or may be directly set by an operator if the MODEATTR parameter is set to Operator and the block is inactive.
- SI is defaulted to OFF, it must be set to ON to force OP to go to SAFEOP.
- When SI turns OFF, OP = SAFEOP is maintained until changed by:
 - the operator
 - a user program
 - another Safety Override Interlock

Override Interlocks

OI[0..2] are Override Interlocks which, when active, force the commanded output (OP) to a respective state regardless of the condition of the Permissive Interlocks. OP cannot be commanded to a different state when an Override Interlock is active.

- Override Interlocks may be connected to other block outputs or may be directly set by an operator if MODEATTR = OPERATOR and the block is inactive.
- Override Interlock parameters are all defaulted to OFF, thereby disabling all the Override Interlocks. They must be set to ON to force OP to go to any specific state. If the Override Interlock forces OP to go to a momentary state, it stays in that state as long as the interlock remains ON and then switches back to the original state when the Override Interlock is reset to OFF.
- SI has a higher priority than any of the Override Interlocks; the priorities of the Override Interlocks themselves are determined by the state assigned to SAFEOP as follows:
 - If SAFEOP = State 0, then priority is SI, OI[0], OI[1], OI[2]
 - If SAFEOP = State 1, then priority is SI, OI[1], OI[0], OI[2]
 - If SAFEOP = State 2, then priority is SI, OI[2], OI[0], OI[1]

Configurable Override/Permissive Interlock Bypass

To grant an operator the ability to bypass the Permissive and Override Interlocks for a DEVCTL block, the parameter BYPPERM must be set to ON. The operator can then set or reset the parameter BYPASS.

- When BYPASS is ON, OP can be changed regardless of the state of the Override Interlocks.
- When BYPASS is reset to OFF, existing Override Interlocks (if any) take effect immediately.
- BYPASS does not affect the Safety Override Interlock (SI).
- When BYPPERM is OFF, BYPASS defaults to OFF and is read-only.

Alarms

An available set of PV state alarms may be configured to represent disagreements between the Commanded Output State (OP) and the Current Active State (PV). A Safety Override Interlock Alarm is also available. Each of these alarms possesses all the standard attributes of system alarms.

- Command Fail Alarm – generated when the Current Active State (PV) fails to change from an original value to any other value within a configurable time interval after the OP parameter is commanded.
 - You can configure the feedback time (CMDFALALM.TM[0..2]) for each state through the Alarms tab on DEVCTL block configuration form. The value of OP just commanded determines which CMDFALALM.TM[0..2] is active. The CMDFALALM.TM[0..2] setting range is 0 to 1000 seconds. Setting a given CMDFALALM.TM[0..2] parameter to 0 disables the alarm for the associated state[0..2]. The alarm function is also automatically disabled, if there are no inputs or no outputs. CMDFALALM.TM[0..2] changes from or to 0, require CM InActive or CEE Idle.



ATTENTION

The CMDFALALM.TM[0..2] setting must be less than the CMDDISALM.TM[0..2] setting for the same state[0..2].

- Bad PV Alarm – generated whenever the Current Active State (PV) is detected to be a NULL (or bad) state.
- Command Disagree Alarm – generated when the Commanded Output State (OP) is changed and the actual input state (PV) does not change accordingly within a specified feedback time.
 - You can configure the feedback time (CMDDISALM.TM[0..2]) for each state through the Alarms tab on DEVCTL block configuration form. The value of OP just commanded determines which CMDDISALM.TM[0..2] is active. The CMDDISALM.TM[0..2] setting range is 0 to 1000 seconds. Setting a given CMDDISALM.TM[0..2] parameter to 0 disables the alarm for the associated state[0..2]. The alarm function is also automatically disabled, if there are no inputs or no outputs. CMDDISALM.TM[0..2] changes from or to 0, require CM InActive or CEE Idle.
 - This alarm condition returns to normal when the input PV state becomes equal to the OP state. The alarm is not generated for momentary commanded states.

- Uncommanded Change Alarm – generated if the actual input state (PV) changes but has not been commanded to change (unless it is a bad PV). This alarm is configured whenever the Command Disagree Alarm is configured.
 - This alarm condition returns to normal when the input PV state becomes equal to the commanded OP state. The alarm is not generated for momentary commanded states.

Off Normal Alarm – This alarm is generated whenever PV does not match OPREQ, if OPREQ is not Null.

- Override Interlock Alarms – When the alarm is enabled and the active interlock causes an OP state change, the alarm will be generated.
- Safety Override Interlock Alarm – When the alarm is enabled and the active interlock causes an OP state change, the alarm will be generated.

If a real-time conflict exists between a Safety Override Interlock Alarm configured to alarm and a PV alarm condition, such as Uncommanded Change Alarm, interlock action (setting of the output state and related alarm notification) always occurs regardless of effects of the other alarm.

Seal-In option

The Seal-In option is used to clear output commands when the process feedback state (PV) cannot follow the commanded output state (OP) as detected by the Command Disagree or Uncommanded Change alarms. If enabled, when the condition is detected, field output destinations are set to the Safe Output State (SAFEOP), but OP is not altered. You can observe OPFINAL to determine what state was actually commanded to the output destinations. The OPFINAL is displayed in reverse video while monitoring Control Builder if it differs from OP. OPFINAL is set equal to OP on the next store to OP, which clears the “seal” condition.

- Seal-In option and Momentary state are mutually exclusive. The Momentary state has to be **None** to configure the Seal-In option.
- You can configure the seal-in option through the SEALOPT (Enable/Disable) parameter.
- When you enable the SEALOPT, any Momentary State selection is negated .

Initialization Manual condition

Initialization Manual is a condition resulting from failure in the field devices connected to the output of the Discrete Control FB. When this condition is active, the parameter INITMAN is set ON. Outputs may not be commanded when INITMAN is TRUE.

- INITDOWN[1..3] - This is an input which may be connected to the DOC INITREQ output. When possible, this connection will be made automatically by the system, without action required of the user.
- This is a structure containing the INITREQ status and the DOC.SO present value.
- INITCONNECTD[1..3] - an internal parameter not normally visible to an Operator, which is set by the FB Builder when the corresponding INITDOWN[] is connected.
- INITMAN - This is a BOOLEAN value which is set TRUE whenever any of the INITDOWN[i].STATUS are TRUE.

OP Initialization Option

The parameter INITOPOPT is used to configure OP Initialization option. It is an enumeration of NORMALOPT, SAFEOPOPT or HOLDOPOPT. The default value is NORMALOPT.

- INITOPOPT = NORMALOPT, perform normal initialization as described below in *Initialization Manual Condition with Safety Override Interlock, Override Interlocks, LocalMan, and OP Initialization*.
- INITOPOPT = SAFEOPOPT, OP is set to SAFEOP
- INITOPOPT = HOLDOPOPT, initialization will not be performed. OP remains the last value.

Initialization Manual Condition with Safety Override Interlock, Override Interlocks, LocalMan, and OP Initialization

The Safety Override Interlock and the Override Interlocks have an impact on how OP initialization works, as described in the following.

When the INITMAN parameter transitions from ON to OFF, the Device Control FB provides an output value OP as follows:

- If the Safety Interlock is active, the OP is set to SAFEOP;
- Otherwise, if any of the Override Interlocks are active and not bypassed, the OP is set to the highest priority Override Interlock;
- Otherwise, if LocalMan is ON, OP tracks PV, if PV is in a settable state (State0, State1, or State2). If PV is in an unsettable state (Null or InBetween), or PV does not exist, OP is set to SafeOP;
- Otherwise, if OP Initialization is configured as HOLDOPOPT, OP remains on the last value;
- Otherwise, if OP Initialization is configured as SAFEOPOPT, OP is set to SafeOP;
- Otherwise (OP Initialization is configured as NORMALOPOPT), in cases where feedback is configured, the stored OP value tracks the PV state if the PV state is settable (State0, State1, or State2);
- Otherwise, OP value is back-initialized from the output connections if
 - there are no output types of ONPULSE/OFFPULSE, and
 - if a valid OP value can be constructed from the values of the output connections.

Otherwise, OP is set to SAFEOP.

Initialization with Pulse Output

If Pulse Outputs are configured, the following rules should be followed in generating pulses when recovering from initialization :

- When PV is good, OP and OpFinal are initialized to PV, no pulses should be generated.
- When PV is Bad, OP and OpFinal are initialized to SafeOp, pulses should be generated.

Initialization Request Flags

The Device Control function block parameter INITREQ[0..2] provides an indication whether a command to a certain state (corresponding to the parameter index 0..2) will be accepted and acted upon at the present time. This parameter can be read prior to sending a command to the block to check if the device can respond as desired. INITREQ[i] (i = 0, 1, or 2) = OFF indicates that the block can be commanded to state_i, and INITREQ [i] (i = 0, 1, or 2) = ON indicates that the block can not be commanded to state_i. Things like override interlocks, permissive interlocks, etc. can cause a certain state to not be settable at a given point in time.



TIP

Note that the INITREQ is used differently in DevCtl block than in other blocks, such as DOC, AOC, or RegCtl.

OP and DO Initialization After Load

This function gives you an opportunity to configure the initialization values of digital outputs (DOs) to their desired values. This feature is typically used for the strategy where the outputs of a DevCtl FB are connected to non-initializable blocks, such as logic blocks. The configuration is done through a new parameter, INITOPAFTLD. The user will have to configure the initialization state for OP, and the value of OP will be mapped to DOs, according to the configured map of OP-DO (OPDOMAP), after load. The options for INITOPAFTLD can be any configured states (State0, State1, or State2 if 3-state is configured), or default. The default option will initialize OP to State0, and all the DOs to 0 (OFF). The OP/DO initialization value configured here

Maintenance Statistics

The DEVCTL block collects a set of Maintenance Statistics which are enabled by configuring MAINTOPT = ON.

The following parameters can be configured to provide suggested maximums. No operations are rejected due to the values of these parameters. These MAXxxx parameters are useful as references for comparison with the actual measured statistics.

- MAXTRANS [0 .. 2] – maximum number of transitions of PV to each state. Useful to compare these values to NUMTRANS [0 .. 2].
- MAXTIME [0 .. 2] – maximum number of hours of PV accumulated in each state. Useful to compare these values to STATETIME [0 .. 2].

The statistics collected include:

- NUMTRANS [0 .. 2] – accumulated number of transitions of PV to each state (since the last statistics reset).
- STATETIME [0..2] – accumulated time of PV in each state (since the last statistics reset).
- NUMSIOVRD – accumulated number of safety interlock trips, which result in OP changing state (since the last statistics reset).

The statistics are accumulated since the most recent reset. The operator only can reset the statistics while the DEVCTL block is red-tagged, but a program (or other function block) can reset the statistics by storing ON to RESET.FL anytime.

Output requests

Whenever an external FB attempts to change the commanded state OP, the DEVCTL block uses the OP request mechanism. The OP request (OPREQ/GOPREQ) differs from direct access an operator uses to the commanded state OP. The OPREQ is a string in the same manner as OP, and GOPREQ is the enumeration GENSTAT_ENM, which is the same as GOP.

There is no direct access to OPREQ when MODEATTR is PROGRAM. It may be changed as part of a control request from an SCM. When MODEATTR is OPERATOR, an operator can change OPREQ, but this does not block a control request. This means a program store to OPREQ cannot be rejected, and no error is returned. The FB retains the stored value until it is overwritten, except in certain non-stored cases when the level 1 drivers are active. OPREQ acts like a repeated attempt to store to OP. The OPREQ is always active unless it is Null. This means the OPREQ will continue to attempt stores even if attributes, such as interlocks, become active and block changes to OP. Thus, once the attributes blocking change to OP have reset OPREQ stores the commanded state to OP.

Output command

The block provides a Boolean command capability through an array of Boolean inputs (OPCMD[0..2]). When the mode attribute (MODEATTR) is Program and the SCM option (SCMOPT) is None, you can use an output from a Logic type block to set the requested output state (OPREQ) through the given Boolean input command (OPCMD[0..2]). When the given OPCMD[0..2] is set to ON, the block sets the OPREQ to the corresponding state. In this case, the OPCMD[0] corresponds to state0, OPCMD[1] corresponds to state1, and OPCMD[2] corresponds to state2. When more than one of the Boolean inputs (OPCMD[0..2]) are ON, the OPREQ is set according to the following priority.

- If SAFEOP is SO, the priority is OPCMD[0], OPCMD[1], OPCMD[2].
- If SAFEOP is S1, the priority is OPCMD[1], OPCMD[0], OPCMD[2].
- If SAFEOP is S2, the priority is OPCMD[2], OPCMD[0], OPCMD[1].

If an SCM commands the device by sending a Null type of request to GOP and there are active OPCMDs (this is possible when SCMOPT = NONE, MODEATTR = Program, and SCM OPTYPE = NULL), the OPCMD has higher priority. An SCM store to GOP will be rejected, if any of the OPCMD[0..2] elements are active (one or more OPCMD[0..2] members are ON). An SCM can only get control, when all OPCMD[0..2] elements are OFF.

Logic override OPREQ

You can use the clear OPREQ flag parameter (CLROPREQFL) through a passive connection to a Logic type block to clear the OPREQ, if the MODEATTR is Program. When the CLROPREQFL changes from OFF to ON, OPREQ is set to NULL and the OP remains unchanged.

DEVCTL parameters



REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the DEVCTL block.

Device Control
DEVCTL (Device Control) Block

Logic Functions

Logic Function Blocks

Functional Overview

The Logic Functions Blocks can be combined with the Device Control Function Block to provide the basis for integrated Logic control. The blocks provided fall into one of these basic functional categories.

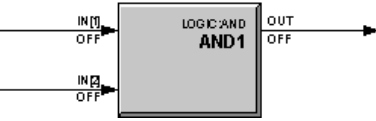
- Bitwise Boolean functions
- Comparison functions
- Arithmetic functions
- Selection functions
- Bistable (flip-flop) functions
- Edge triggered functions
- Timed functions
- Voted functions

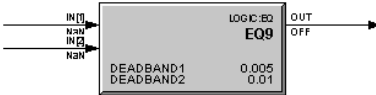
The following table provides a description and a brief explanation of the functional capabilities of the named function block. In most cases, the name of the block is intuitive of its function.

Block Name/ Graphic	Description	Function
2003	2-out-of-3 Voting block outputs (DISCREP and MAJ) are determined as follows: • DISCREP = OFF when all inputs are equal. = ON when all inputs are not equal for time $\geq$ DELAY. • MAJ = value held by the majority of the inputs.	Sets the output (DISCREP) to ON if NOT all inputs agree for a specified time duration (DELAY); otherwise, it is set to OFF.

Logic Functions

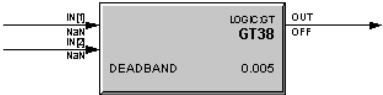
Logic Function Blocks

Block Name/ Graphic	Description	Function
AND 	Provides an up to 8-input AND algorithm, meaning that it performs the Boolean operation of conjunction. Each input (IN[1], IN[2], ..., IN[8]) has the capability of being optionally inverted, if required.	Turns the digital output (OUT) ON only when all inputs (IN[1], IN[2], ..., IN[8]) are ON. Therefore: • If all inputs are ON, then: OUT = ON. • If any input is OFF, then: OUT = OFF.
CHECKBAD 	Provides bad input handling for desired input.	Checks if input (IN) value equals NaN. • If IN = NaN • Then, OUT = ON • Else, OUT = OFF
DELAY 	Provides the ability to delay the output (OUT) response to the given input (IN) by a sample cycle time.	The OUT always follows the input (IN) action by a sample cycle time.

Block Name/ Graphic	Description	Function
EQ 	Provides a 2-input Compare Equal (with deadband range) function, meaning that it compares two inputs for equality within a specified deadband range.	Turns the digital output (OUT) ON only when the two inputs (IN[1] and IN[2]) are considered equal within a specified deadband range or, for single inputs, a designated trip point parameter (TP) as follows: • If $((IN[1] - IN[2]) \leq DEADBAND1)$, then: OUT = ON. • If $((IN[1] - IN[2]) > DEADBAND2)$, then: OUT = OFF. • Else OUT is not changed. DEADBAND1 and DEADBAND2 must satisfy the following constraint:: $0 \leq DEADBAND1 \leq DEADBAND2$ If either input (IN[1] or IN[2]) is NaN, the output (OUT) is set to INBADOPT.
FTRIG 	Falling edged Trigger Block sets the output (OUT) to ON following the ON-to-OFF transition of the input and stays ON until the next execution cycle, at which time it returns to OFF.	Provides falling edged change detection, thereby turning the output ON if an ON-to-OFF transition is detected.

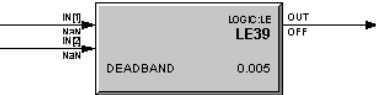
Logic Functions
Logic Function Blocks

Block Name/ Graphic	Description	Function
GE  The graphic shows a rectangular block labeled 'LOGIC GE GE14' and 'DEADBAND 0.005'. It has two input lines on the left labeled 'IN[1]' and 'IN[2]', with 'NaN' written below each. An output line on the right is labeled 'OUT' and 'OFF'.	Provides a 1- or 2-input Compare Greater Than or Equal (with deadband) function, meaning it checks to see if one designated input (IN[1]) is greater than or equal to either a second input (IN[2]) or, for single inputs, a designated trip point parameter (TP).	Turns the digital output (OUT) ON only when one designated input (IN[1]) is greater than or equal to a second input (IN[2]) or, for single inputs, a designated trip point parameter (TP) as follows: • If $IN[1] \geq IN[2]$, then: OUT = ON. • If $IN[1] < (IN[2] - DEADBAND)$, then: OUT = OFF. • If $(IN[2] - DEADBAND) < IN[1] < IN[2]$, then OUT is not changed. If either input (IN[1] or IN[2]) is NaN, the output (OUT) is set to INBADOPT.

Block Name/ Graphic	Description	Function
GT 	Provides a 1- or 2-input Compare Greater Than (with deadband) function, meaning that it checks to see if one designated input (IN[1]) is greater than either a second input (IN[2]) or, for a single input, a designated trip point parameter (TP).	Turns the digital output (OUT) ON only when one designated input (IN[1]) is greater than a second input (IN[2]) or, for single input, a designated trip point parameter (TP) as follows: • If $IN[1] > IN[2]$, then: OUT = ON. • If $IN[1] \leq (IN[2] - DEADBAND)$, then: OUT = OFF. • If $(IN[2] - DEADBAND) < IN[1] \leq IN[2]$, then: OUT is not changed. If either input (IN[1] or IN[2]) is NaN, the output (OUT) is set to INBADOPT.

Logic Functions

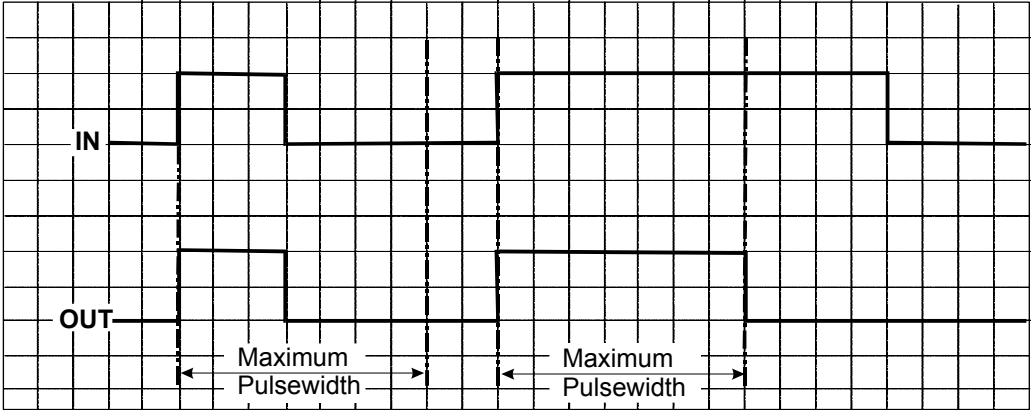
Logic Function Blocks

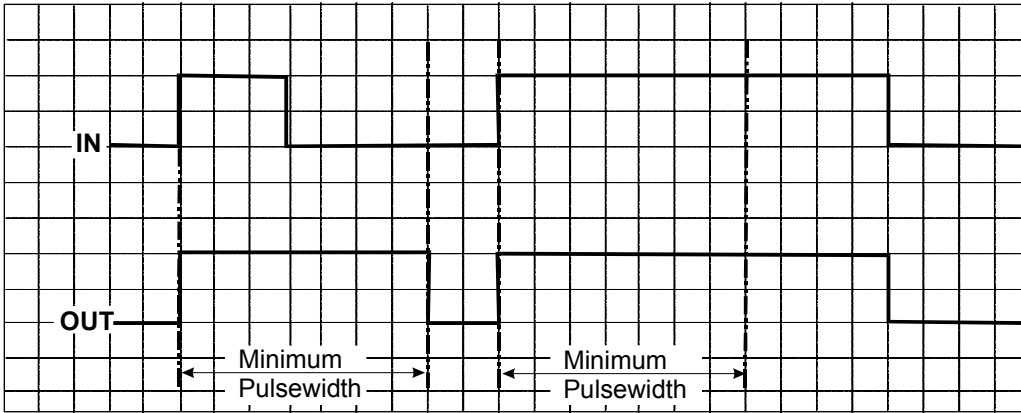
Block Name/ Graphic	Description	Function
LE  The graphic shows a block labeled 'LOGIC:LE' and 'LE39'. It has two inputs on the left: 'IN[1]' and 'IN[2]', both with 'NaN' values. It has one output on the right labeled 'OUT' with an 'OFF' value. A parameter 'DEADBAND' is set to '0.005'.	Provides a 1- or 2-input Compare Less Than or Equal (with deadband) function, meaning it checks to see if one designated input (IN[1]) is less than or equal to either a second input (IN[2]) or, for a single input, a designated trip point parameter (TP).	Turns the digital output (OUT) ON only when one designated input (IN[1]) is less than or equal to a second input (IN[2]) or, for a single input, a designated trip point parameter (TP) as follows: • If $IN[1] \leq IN[2]$, then: OUT = ON. • If $IN[1] > (IN[2] + DEADBAND)$, then: OUT = OFF. • If $IN[2] < IN[1] \leq (IN[2] + DEADBAND)$, then: OUT is not changed. If either input (IN[1] or IN[2]) is NaN, the output (OUT) is set to INBADOPT.
LIMIT  The graphic shows a block labeled 'LOGIC:LIMIT' and 'LIMIT41'. It has one input on the left labeled 'IN' with a 'NaN' value. It has one output on the right labeled 'OUT' with a 'NaN' value. Two parameters are shown: 'MAX' and 'MIN', both set to '0'.	Provides a 3-input limit function, meaning that it provides an output that is maintained within a specified range as defined by user-specified minimum and maximum values.	Provides an output that is maintained within a specified range as follows: • $MN \leq OUT \leq MX$ • If IN is not NaN, OUT = MIN (MAX (IN, MIN), MAX) • If IN = NaN, OUT = NaN

Block Name/ Graphic	Description	Function
LT  The graphic for the LT42 block shows a rectangular box with two input lines on the left labeled IN[1] and IN[2], both with NaN values. Inside the box, the text reads 'LOGIC:LT LT42' and 'DEADBAND 0.005'. An output line on the right is labeled 'OUT' with the value 'OFF'.	Provides a 1- or 2-input Compare Less Than (with deadband) function, meaning that it checks to see if one designated input (IN[1]) is less than either a second input (IN[2]) or, for a single input, a designated trip point parameter (TP).	Turns the digital output (OUT) ON only when one designated input (IN[1]) is less than a second input (IN[2]) or, for a single input, a designated trip point parameter (TP) as follows: • If $IN[1] < IN[2]$, then: OUT = ON. • If $IN[1] \geq (IN[2] + DEADBAND)$, then: OUT = OFF. • If $IN[2] \leq IN[1] < (IN[2] + DEADBAND)$, then: OUT is not changed. If either input (IN[1] or IN[2]) is NaN, the output (OUT) is set to INBADOPT.
MAX  The graphic for the MAX43 block shows a rectangular box with two input lines on the left labeled IN[1] and IN[2], both with NaN values. Inside the box, the text reads 'LOGIC:MAX MAX43'. An output line on the right is labeled 'OUT' with the value 'NaN'.	Provides an N-input MAX function, meaning that it provides an output that is the maximum value of N-inputs.	Used to isolate the highest value of multiple input values and use it as a designated output value.

Logic Functions

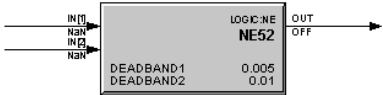
Logic Function Blocks

Block Name/ Graphic	Description	Function
MAXPULSE 	Provides a maximum time limit pulse output (OUT) each time the input (IN) transitions from OFF to ON. You specify the maximum output pulse width (PULSEWIDTH) in seconds through configuration.	Used to limit the output (OUT) pulse to a maximum width. - If the input (IN) pulse time is less than or equal to the specified PULSEWIDTH time, IN is assumed to equal one output (OUT) pulse. - If the IN pulse time is greater than the specified PULSEWIDTH time, OUT pulse terminates at end of specified PULSEWIDTH time.
MAXPULSE timing diagram: 		
MIN 	Provides an N-input MIN function, meaning that it provides an output that is the minimum value of N-inputs.	Used to isolate the lowest value of multiple input values and use it as a designated output value.

Block Name/ Graphic	Description	Function
MINPULSE 	Provides a minimum time limit pulse output (OUT) each time the input (IN) transitions from OFF to ON. You specify the minimum output pulse width (PULSEWIDTH) in seconds through configuration.	Used to define the minimum output (OUT) pulse width. - If the input (IN) pulse time is less than or equal to the specified PULSEWIDTH time, output (OUT) pulse width equals the specified time. - If the IN pulse time is greater than the specified PULSEWIDTH time, OUT pulse width tracks IN pulse time, so OUT pulse exceeds specified time.
MINPULSE timing diagram: 		
MUX 	Provides an up to 8-input Extensible Multiplexer algorithm, meaning that it selects 1 of "N" inputs depending on a separate input K.	Sets the actual output (OUT) to a particular input (IN[1], IN[2], ..., IN[8]) depending on the value of a separate input K. $OUT = IN_{k+1}$

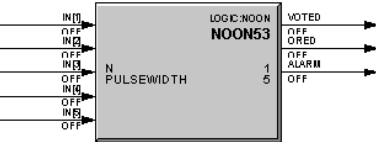
Logic Functions
Logic Function Blocks

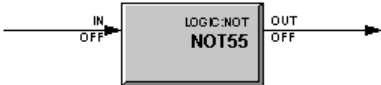
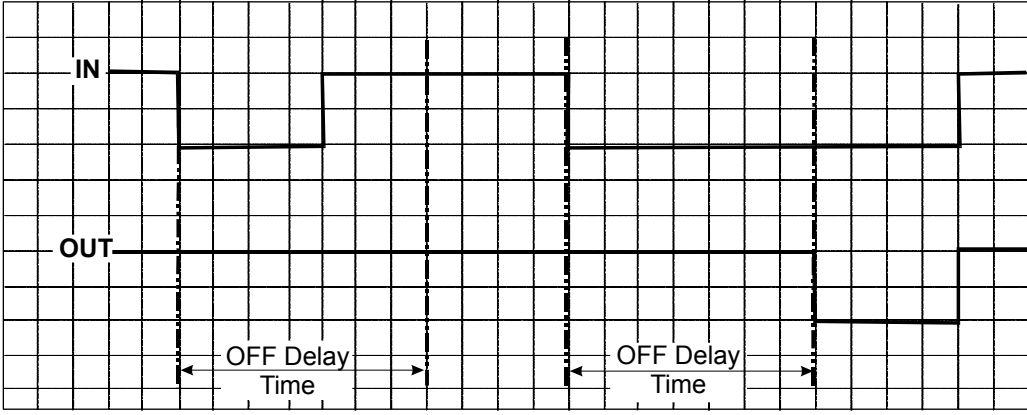
Block Name/ Graphic	Description	Function
MUX-REAL 	Provides an up to 8-input Extensible Multiplexer algorithm, meaning that it selects 1 of “N” inputs depending on a separate input K.	Sets the actual output (OUT) to a particular input (IN[1], IN[2], ..., IN[8]) depending on the value of a separate input K. OUT = IN_k+1
MVOTE 	Provides an output (MAJ) value that equals the value of the majority of the inputs (IN[1..8]) and sets another output (DISCREP) to ON if not all inputs agree for a specified time (DELAY). You specify the time (DELAY) in seconds through configuration. You must also specify the number of inputs (NUMOFINPUTS) through configuration.	Sets the MAJ output equal to the value of the majority of the inputs (IN[1..8]). Sets the DISCREP output to ON, if not all inputs agree during the specified time (DELAY). DELAY is a unit integer with time unit in seconds.
NAND 	Provides an up to 8-input NAND algorithm, meaning that it performs an inverted AND function. Each input (IN[1], IN[2], ..., IN[8]) has the capability of being optionally inverted, if required.	Turns the digital output (OUT) OFF only when all inputs (IN[1], IN[2], ..., IN[8]) are ON; therefore: - If all inputs are ON, then: OUT = OFF. - If any input is OFF, then: OUT = ON.

Block Name/ Graphic	Description	Function
NE 	Provides a 1- or 2-input Compare Not Equal (with deadband range) function, meaning that it checks to see if one designated input (IN[1]) is not equal to either a second input (IN[2]) or, for a single input, a designated trip point parameter (TP).	Turns the digital output (OUT) ON only when the two inputs (IN[1] and IN[2]) are not considered equal within a specified deadband range or, for single inputs, a designated trip point parameter (TP) as follows: • If $((IN[1] - IN[2]) \leq DEADBAND1)$, then: OUT = OFF. • If $((IN[1] - IN[2]) > DEADBAND2)$, then: OUT = ON. • Else OUT is not changed. DEADBAND1 and DEADBAND2 must satisfy the following constraint:: $0 \leq DEADBAND1 \leq DEADBAND2$ If either input (IN[1] or IN[2]) is NaN, the output (OUT) is set to INBADOPT.

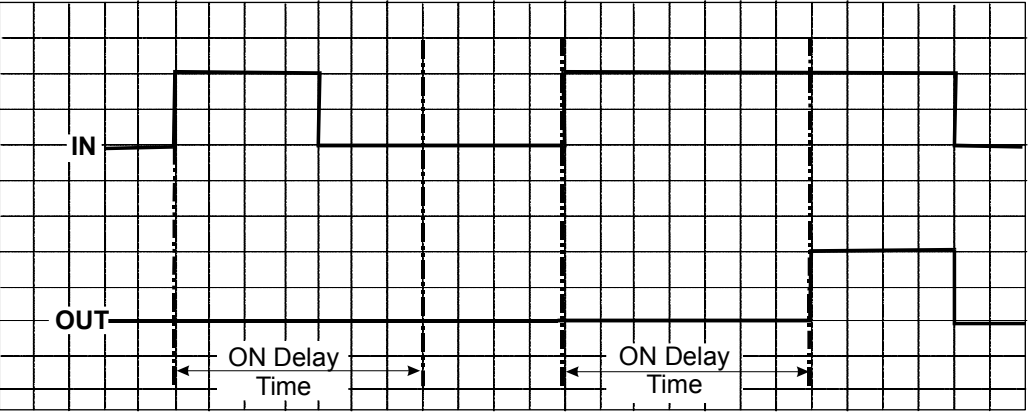
Logic Functions

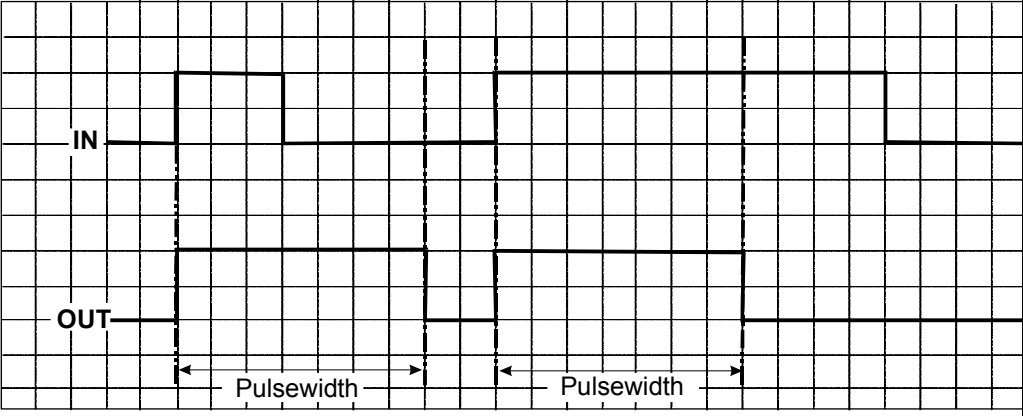
Logic Function Blocks

Block Name/ Graphic	Description	Function
nOON 	n-out-of-N voting block; outputs are computed as follows: - VOTED output is set to ON if at least n inputs are ON, otherwise it is set to OFF. - ORED output is set to ON if any input is ON, otherwise it is set to OFF. - ALARM output is a pulse output – every time an input turns ON, a fixed pulse (of the pulsewidth specified by PULSEWIDTH parameter) is generated, provided the total number of inputs which are ON is less than n.	Provides VOTED, ORED and ALARM outputs in support of logical functions.
NOR 	Provides an up to 8-input NOR algorithm, meaning that it performs an inverted OR function. Each input (IN[1], IN[2], ..., IN[8]) has the capability of being optionally inverted, if required.	Turns the digital output (OUT) OFF if any one input (IN[1], IN[2], ..., IN[8]) is ON; therefore: - If all inputs are OFF, then: OUT = ON. - If any one input is ON, then: OUT = OFF.

Block Name/ Graphic	Description	Function
NOT 	Provides a NOT algorithm, meaning it performs an inversion function.	Reverses the state of a digital input (IN) such that the output (OUT) is the complement of the single input; therefore: - OUT = opposite of IN - If IN = ON, then: OUT = OFF. - If IN = OFF, then OUT = ON.
OFFDELAY 	Delays the input signal supplied at the input (IN) when the input signal transitions from ON to OFF.	Used to delay the input by a specified delay time after an ON/OFF device transitions from the ON state to the OFF state. Delay time is specified by the DELAYTIME parameter.
OFFDELAY timing diagram:		
		

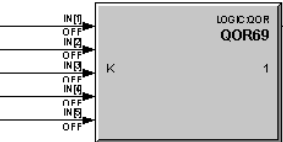
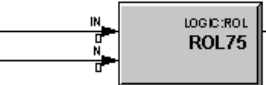
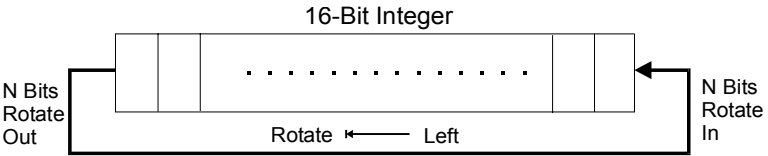
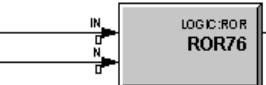
Logic Functions
Logic Function Blocks

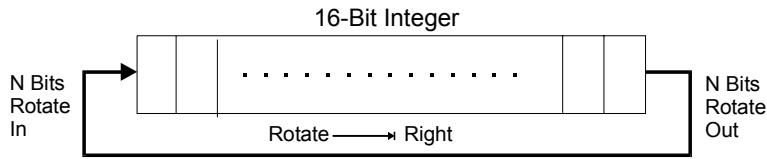
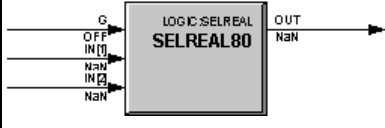
Block Name/ Graphic	Description	Function
ONDELAY 	Delays the input signal supplied at the input (IN) when the input signal transitions from OFF to ON.	Used to delay the input by a specified delay time after an ON/OFF device transitions from the OFF state to the ON state. Delay time is specified by the DELAYTIME parameter.
ONDELAY timing diagram: 		
OR 	Provides an up to 8-input OR algorithm, meaning that it performs the inclusive OR Boolean function. Each input (IN[1], IN[2], ..., IN[8]) has the capability of being optionally inverted, if required.	Turns the digital output (OUT) ON if any one input (IN[1], IN[2], ..., IN[8]) is ON; therefore: - If all inputs are OFF, then: OUT = OFF. - If any one input is ON, then: OUT = ON.

Block Name/ Graphic	Description	Function
PULSE 	Provides a fixed pulse output (OUT) each time the input (IN) transitions from OFF to ON. You specify the fixed output pulse width (PULSEWIDTH) in seconds through configuration.	Used to define the fixed output (OUT) pulse width. - If the input (IN) pulse time is less than or equal to the fixed PULSEWIDTH time, output (OUT) pulse width equals the fixed time. - If the IN pulse time is greater than the fixed PULSEWIDTH time, OUT pulse width is restricted to the fixed time. Another output pulse cannot be generated until the preceding pulse has completed.
PULSE timing diagram: 		

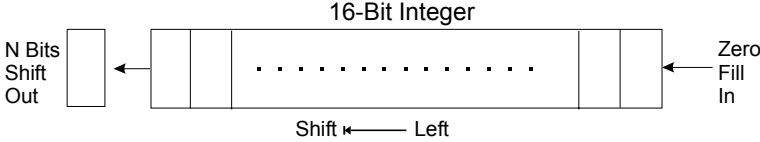
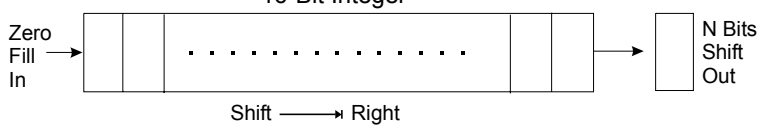
Logic Functions

Logic Function Blocks

Block Name/ Graphic	Description	Function
QOR 	Qualified OR – provides an (N + 1)-input generic-qualified OR function, meaning that the output (OUT) is turned ON if a certain number (k) of total inputs (IN[n]) is ON. Each input (IN[1], IN[2], ..., IN[8]) has the capability of being optionally inverted, if required.	Turns the digital output (OUT) ON if a specified number (k) of total digital inputs is ON.
ROL 	Provides a 16-bit integer output (OUT) that is rotated to the left by the number of bits (N) specified from the 16-bit integer input (IN). You specify the number of bits through configuration.	Used to shift out bits in the output (OUT) by rotating the bits in the input (IN) left by the number of bits (N) specified. - OUT = IN left rotated by N bits, circular. - If IN is NaN, then, OUT = NaN.
ROL execution diagram: 		
ROR 	Provides a 16-bit integer output (OUT) that is rotated to the right by the number of bits (N) specified from the 16-bit integer input (IN). You specify the number of bits through configuration.	Used to shift out bits in the output (OUT) by rotating the bits in the input (IN) right by the number of bits (N) specified. - OUT = IN right rotated by N bits, circular. - If IN is NaN, then, OUT = NaN.

Block Name/ Graphic	Description	Function
ROR execution diagram: 		
RS 	Reset – provides a bistable Reset Dominant flip-flop as defined in the IEC DIS 1131-3 standard.	Specifies the output (Q) of the flip-flop as a function of the input S (Set), the input R (Reset), and the last state of Q.
RTRIG 	Rising-Trigger – sets the output (OUT) to ON following the OFF-to-ON transition of the input (IN) and stays at ON until the next execution cycle, at which time it returns to OFF.	Provides rising edge change detection, thereby turning the output ON if an OFF-to-ON transition is detected.
SEL 	Provides a 3-input selector function, meaning it selects 1 of 2 inputs (IN[1] and IN[2]) depending on the separate input G.	Sets the actual output (OUT) equal to the value of 1 of 2 inputs (IN[1] or IN[2]), depending on the value of a separate input (G). • If G = OFF, OUT = IN1 • If G = ON, OUT = IN2
SEL-REAL 	Provides a 3-input selector function, meaning it selects 1 of 2 inputs (IN[1] or IN[2]) depending on the separate input (G).	Sets the actual output (OUT) equal to the value of 1 of 2 inputs (IN[1] or IN[2]), depending on the value of a separate input (G). • If G = OFF, OUT = IN1 • If G = ON, OUT = IN2

Logic Functions
Logic Function Blocks

Block Name/ Graphic	Description	Function
SHL 	Provides a 16-bit integer output (OUT) that is shifted to the left by the number of bits (N) specified from the 16-bit integer input (IN). You specify the number of bits (N) through configuration.	Used to shift out bits in the output (OUT) by shifting the bits in the input (IN) left by the number of bits (N) specified. - OUT = IN left shifted by N bits, zero filled on right. - If IN is NaN, then, OUT = NaN.
SHL execution diagram: 		
SHR 	Provides a 16-bit integer output (OUT) that is shifted to the right by the number of bits (N) specified from the 16-bit integer input (IN). You specify the number of bits through configuration.	Used to shift out bits in the output (OUT) by shifting the bits in the input (IN) right by the number of bits (N) specified. - OUT = IN right shifted by N bits, zero filled on left. - If IN is NaN, then, OUT = NaN.
SHR execution diagram: 		

Block Name/ Graphic	Description	Function
SR 	Provides a bistable Set Dominant flip-flop as defined in the IEC DIS 1131-3 standard.	Specifies the output (Q) of the flip-flop as a function of the input S (set), the input R (Reset), and the last state of Q.
TRIG 	Sets the output (OUT) to ON following the OFF-to-ON or ON-to-OFF transition of the input (IN) and stays at ON until the next execution cycle, at which time it returns to OFF.	Provides change detection, thereby turning the output ON if an OFF-to-ON or ON-to-OFF transition is detected.
WATCHDOG 	Monitors other system functions or remote devices and sets the output (OUT) to ON if the monitored function or device fails.	Used to monitor other system functions or remote devices. - Monitored function or device must set IN parameter to ON within a specified time interval (DELAYTIME), otherwise it is assumed to have failed and output (OUT) is set to ON. - If output (OUT) is ON, it is reset to OFF as soon as IN is set to ON.

Logic Functions

Logic Function Blocks

Block Name/ Graphic	Description	Function
XOR 	Provides an up to 8-input XOR algorithm, meaning it performs the exclusive OR function. $OUT = IN[1] \text{ XOR } IN[2] \text{ XOR } IN[3] \dots \text{ XOR } IN[n]$ Each input (IN[1], IN[2], ..., IN[8]) has the capability of being optionally inverted, if required.	Turns digital output (OUT): • OFF when even number of inputs (2, 4, 6, or 8) are ON. • ON when odd number of inputs (1, 3, 5, or 7) are ON. This action adheres to the IEC DIS 1131-3 standard. But, this may not be the expected behavior for an XOR with more than two inputs.

Parameters



REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the named logic function block.

Utility Functions

Utility Function Blocks

Functional overview

Utility function blocks provide a variety of configurable functions for storing and alarming selected control data.

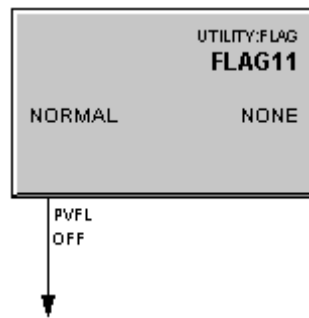
The following table presents the various functions that can be performed through the configuration of the associated Utility function block. Functional descriptions for each block are given in the following subsections.

Function	Block	Description
Store a single two-state value	FLAG Block	Used to define two separate states (for example, Running/Stopped, Off/On) to indicate the status of a particular input.
Store multiple two-state values	FLAGARRAY Block	Used to define two separate states (Off/On) to indicate status of a particular input.
Provide client triggered messages	MESSAGE Block	Used to define up to 16 information only or confirmation type messages that can be triggered by a client of the block.
Store a floating point value	NUMERIC Block	Used to store up to 8 bytes of a floating point value within defined upper and lower limits for use in a control strategy.
Store multiple floating point values	NUMERICARRAY Block	Used to store up to 200 floating point values for use in a control strategy.
Push the value of various data types.	PUSH Block	Used to push the value of different data types to the output destination.
Store multiple text strings	TEXTARRAY Block	Used to store up to 120 text strings for use in a control strategy.
Time process events or create known delays.	TIMER Block	Used to keep track of elapsed time during a process and provides indication when elapsed time reaches predefined limit.
Provide data type conversions	TYPECONVERT Block	Used to convert one data type to another for connecting parameters of different data types.

FLAG Block

Description

The FLAG function block provides storage for a single 2-state value. The value can be accessed as a simple Boolean (Off or On) using the PVFL parameter, or as one of two user-configured State values (for example, Running and Stopped) through the PV parameter. It looks like this graphically.



Function

Used to define two separate states (for example, Running/Stopped, Off/On) to indicate status of a particular input.

- There are 2 user-configurable state descriptors, STATETEXT[0] and STATETEXT[1] which are used to describe STATE0 and STATE1 respectively.
- Current state of flag can be changed/read using PVFL (Boolean) or using PV (either STATETEXT[0] or STATETEXT[1]).
- Block also supports:
 - configurable access lock which determines who can write a value to the block (such as operator, engineer, or other function block).
 - an Off-Normal Alarm whereby one of the flag's states is configured as the normal state; whenever the flag changes state, the Off-Normal Alarm is generated.

Input/Output

The block has one output flag (PVFL). But, all block pin parameters are available to be exposed and connected to using Control Builder graphical connections.

FLAG parameters



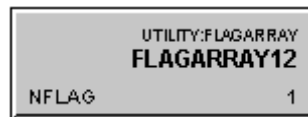
REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the FLAG function block.

FLAGARRAY Block

Description

The FLAGARRAY function block provides storage for up to 1000 2-state values. The value can be accessed as a simple Boolean (Off or On) using the PVFL[n] parameter. Where “n” is the number of the flag. It looks like this graphically:



Function

Used to define two separate states (Off/On) to indicate status of a particular input.

- Number of flag values (NFLAG) is user configurable.
- Current state of flags can be changed/read using flag value (PVFL[n]) (Boolean).
- Block also supports configurable access lock which determines who can write a value to the block (such as an operator, engineer, or other function block).

Input/Output

The block has up to 1000 output flags(PVFL[n]). But, all block pin parameters are available to be exposed and connected to using Control Builder graphical connections.

FLAGARRAY parameters



REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the FLAGARRAY function block.

MESSAGE Block

Description

The MESSAGE block provides up to 16 user configurable messages (MESSAGE[n]) that can be triggered by a client of the block. Where “n” is the number of the message. A client can be the output from a Step block in a Sequential Control Chart module (SCM).

You can also configure each message type (MSGTYPE[n]) to be either:

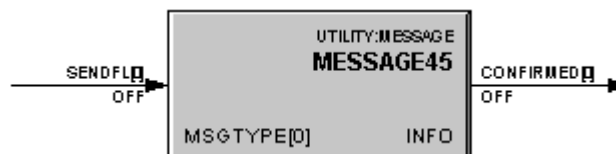
- Information,
- Confirmable,
- Single Signature, or
- Double Signature.



ATTENTION

You must have the Electronic Signature system license to use the Single Signature and Double Signature message types.

It looks like this graphically.



Function

When a client triggers a given send flag (SENDFL[n]) input, the corresponding message (MESSAGE[n]) is sent to the Message and the Event Summary displays in the Station application.

For information only type (INFO) messages, the client trigger sets the corresponding SENDFL[n] to True. Since the SENDFL[n] is a pulse trigger, it is automatically set to False during the next execution cycle. This means the MESSAGE block is ready to send the same message again in the next cycle.

For confirmation type (CONFIRM) messages, the client trigger pulses the corresponding SENDFL[n] to send the MESSAGE[n] to the Server. The client of the MESSAGE block checks for the confirmed parameter (CONFIRMED[n]) to be set to True. The CONFIRMED[n] parameter indicates whether the MESSAGE block has received a confirmation.

For single signature type (SINGLESIGNATURE) messages, the client trigger pulses the corresponding SENDFL[n] to send the MESSAGE[n] to the Server. Once a user acknowledges the message twice to confirm it through the Message Summary display in Station, a Single Signature user interface appears for the user to record an electronic signature. The MEANINGPRI[n] parameter provides an indication for the meaning of the primary signature. Once the message is acknowledged and signature is obtained, the Message Summary Display sends a confirmation to the MESSAGE block that turns on the CONFIRMED[n] parameter to show that the message has been confirmed.

For double signature type (DOUBLESIGNATURE) messages, the client trigger pulses the corresponding SENDFL[n] to send the MESSAGE[n] to the Server. Once a user acknowledges the message twice to confirm it through the Message Summary Display in Station, a Single Signature and Double Signature user interface appear for the user to record the required electronic signatures. The MEANINGPRI[n] and MEANINGSEC[n] parameters provide indications for the meaning of the primary and secondary signatures, respectively. Once the message is acknowledged and signatures are obtained, the Message Summary Display sends a confirmation to the MESSAGE block that turns on the CONFIRMED[n] parameter to show that the message has been confirmed. In addition, the MINLVLSECSIG[n] parameter lets users define the minimum security level required for a secondary signature.

A message can be confirmed by acknowledging it twice through the Message Summary display in Station or through the block's CONFIRM[n] parameter in the Monitoring mode in Control Builder. Both actions set the CONFIRMED[n] parameter true (ON), which initiates a corresponding entry in the Event Summary display to record the action. If the CONFIRM[n] parameter is set through the Monitoring mode, an operator must still acknowledge the message through the Message Summary display to remove it from the display.

The CONFIRM[n] parameter can be configured as a block input pin and/or a monitoring parameter that appears on the face of the block in the Monitoring mode. This means that a client block or an operator, depending upon application requirements, can trigger it.

The MESSAGE[n] and MSGTYPE[n] parameters can also be configured as block input pins and/or monitoring parameters. However, the MESSAGE[n], MEANINGPRI[n], and MEANINGSEC[n] parameters **cannot** be changed online in the monitoring mode. It is possible to change the MSGTYPE[n] and MINLVLSECSIG[n] parameters online in the

Monitoring mode should the application requirements change with an access level of Engineer or greater.

Configuration and Operation Considerations

Some general considerations for configuring and operating MESSAGE blocks are listed here for reference.

- Each message has a maximum length of 132 characters.
- A new message **cannot** be sent when the message is awaiting/blocked on a confirmation (CONFIRMED[n] parameter).
- You **cannot** configure the message type (MSGTYPE[n]) or minimum level secondary signature (MINLVLSECSIG[n]) when the message is awaiting/blocked on a confirmation (CONFIRMED[n] parameter).
- You **cannot** configure a message (MESSAGE[n], meaning primary signature (MEANINGPRI[n]) or meaning secondary signature (MEANINGSEC[n]) through the Monitoring tab. You must configure messages through the Project tab and then load them to the Controller.
- When you acknowledge an Information message, it is removed from the Message Summary display. Confirmation type messages are confirmed by a second acknowledgement and then removed from the display.

Input/Output

The block has up to 16 inputs (SENDFL[0..15]) and 16 outputs (CONFIRMED[0..15]), depending on the message types configured.

MESSAGE parameters



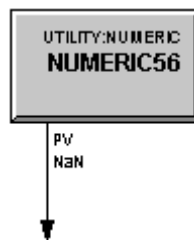
REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the MESSAGE function block.

NUMERIC Block

Description

The NUMERIC block provides storage for a floating-point value which is accessible through the PV configuration parameter. It looks like this graphically.



Function

Used to store up to 8 bytes of a floating point value within defined upper and lower limits for use in a control strategy.

- Configurable high and low limits are also provided.
- Also supports a configurable access lock which determines who can write a value to the block (such as operator, engineer, other function block).

Input/Output

The block has one output (PV). But, all block pin parameters are available to be exposed and connected to using Control Builder graphical connections.

NUMERIC parameters



REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the NUMERIC function block.

NUMERICARRAY Block

Description

The NUMERICARRAY block provides storage for up to 200 floating point values which are accessible through the corresponding PV configuration parameter (PV[n]). Where “n” is the number of the numeric. It looks like this graphically:



Function

The NUMERICARRAY block outputs (PV[n]) can be used as source parameters to provide predefined analog constants to other function blocks. A bad numeric output parameter typically has the value NaN (Not-a-Number).

The block supports these user configurable attributes.

- A configurable Access Lock (ACCLOCK) which determines who can write a value to the block (such as operator, engineer, or other function block).
- A configurable PV Format (PVFORMAT) which lets you select the decimal format to be used to display the PV[n] values. The selections are D0 for no decimal place (-XXXXXX.), D1 for one decimal place (-XXXXX.X), D2 for two decimal places (-XXXX.XX), and D3 for three decimal places (-XXX.XXX). The default selection is D1 for one decimal place.
- A configurable Number of Numeric Values (NNUMERIC) which lets you specify the desired number of numeric values to be supported.

Input/Output

The block has up to 200 outputs (PV[n]), depending on the number of numeric values (NNUMERIC) configured. But, all block pin parameters are available to be exposed and connected to using Control Builder graphical connections.

NUMERICARRAY parameters



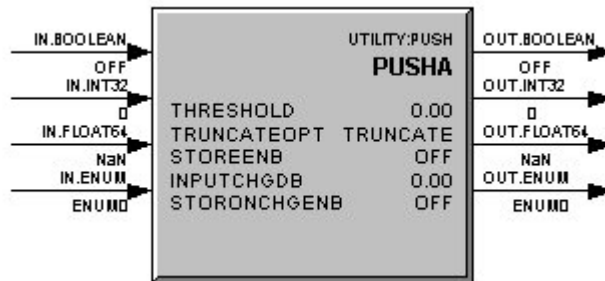
REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the NUMERICARRAY function block.

PUSH Block

Description

The PUSH function block is used to push the value of different data types to the output destination.



Function

The function block fetches the input when it is scheduled to run and stores the output in the same execution cycle after the type conversion. If data type conversion is not necessary, then none will be done.

Execution Status

The status of input fetching is reflected in the following parameter:

- Overall Execution Status (EXECSTS)

The EXECSTS provide information on how successful the block is in fetching the input. EXECSTS can have the following values:

- OK - Successful i.e. when fetching of inputs as well as the conversion was done without any error or clamping.
- CLAMPWARNING - Function completed, but with some limitation (e.g. value clamped after data conversion). This provides information on how successful the block was in type conversion. After fetching good data, if the block had to clamp the input during type conversion, EXECSTS will be CLAMPWARNING.

Utility Functions

PUSH Block

- **BADINPUT** - This happens when the connection to input block is lost or it is simply bad data.
- **INBLKMISSING** – This happens when the block detects that there is no input connection made to any of the inputs of the PUSH block

Store Status

The status of output store is reflected in the following parameter:

- Store status (STORESTS)

The STORESTS provide information on how successful the block is in storing the input. STORESTS can have the following values:

- **STOREOK** - Successful i.e. the store to destination was successful
- **STOREPENDING** – This is an intermediate status when the store is made to a destination, which is in a peer controller. Until the block actually gets store request, the status is STOREPENDING
- **STOREFAIL** – If the output destination block rejects the store, the push block displays the STOREFAIL status. The reason for failure may be very block specific. When the store fails, the PUSH block retries the store immediately in the next execution cycle. If this store also fails, then the store is not tried for two cycles. This continues until the time goes to 6 secs. After that the store is not made until 6 seconds are over. Thus there is exponential increase in time between any two failed stores. This is required to save precious peer-to-peer communication resources
- **DATATYPERR** – This is used if the output store could not be made because of some error in CL/CB where connection of parameters between different data types is allowed. This is also the store status if there is no output connection configured on the PUSH block.

PUSH parameters



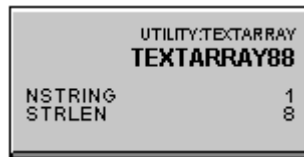
REFERENCE - INTERNAL

Refer to the *Control Builder Components Reference* for a complete list of the parameters used with the PUSH function block.

TEXTARRAY Block

Description

The TEXTARRAY block provides storage for up to 120 text strings which are accessible through the corresponding string configuration parameter (STR[n]). Where “n” is the number of the text string. The length of the text strings is user configurable. It looks like this graphically:



Function

The TEXTARRAY block outputs (STR[n]) can be used to provide predefined text strings to other function blocks.

The block supports these user configurable attributes.

- A configurable Access Lock (ACCLOCK) which lets you define who can write a value to the block (such as operator, engineer, or other function block).
- A configurable Number of String Values (NSTRING) which lets you specify the desired number of string values (up to 120) to be supported.
- A configurable Character Length of String Values which lets you specify the number of characters (8, 16, 32, or 64) allowed in the strings.

The TEXTARRAY block supports a maximum size of 960 two-byte characters. The following table shows the maximum data combinations that you can configure through NSTRING and STRLEN values. Illegal combinations of NSTRING and STRLEN values, those requiring more than 960 two-byte characters of data, will be rejected.

Utility Functions

TEXTARRAY Block

NSTRING Value	STRLEN Value	STR[n] Range
15	64	[1. .15]
30	32	[1. .30]
60	16	[1. .60]
120	8	[1. .120]

Input/Output

The block has up to 120 output strings (STR[n]), depending on the number of string values (NSTRING) configured. But, all block pin parameters are available to be exposed and connected to using Control Builder graphical connections.

TEXTARRAY parameters



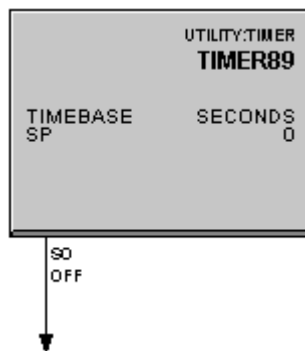
REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the TEXTARRAY function block.

TIMER Block

Description

The TIMER block provides the capability to time process events or create known delays. It looks like this graphically.



Function

Used to keep track of elapsed time during a process and provides indication when elapsed time reaches predefined limit. The TIMEBASE can be configured to represent seconds, minutes, or cycles (number of execution cycles).

Input/Output

The block has one status output (SO). But, all parameters are available to be exposed and connected to using Control Builder graphical connections.

Commands

Commands are sent to the timer in one of two ways:

- By the operator, using the COMMAND parameter
- Through connections to the parameters STARTFL, STOPFL, RESETFL, and RESTARTFL

You can give a Reset command any time, even if the TIMER is not running, and it will always be executed. However, the Stop command is only valid while the TIMER is running. For example, giving a Stop command directly after a Reset command is not allowed.

The Start and Restart commands are not interchangeable. A Start command is only executed after a prior Reset, when the timer is starting from the beginning (PV = 0). Similarly, a Restart command is only executed after a prior Stop command, which froze the timer when it was running (PV usually = non-zero).

When more than one of the Boolean command parameters are set at the same time, the following priority is used:

- RESETFL - *highest priority*
- STOPFL
- RESTARTFL
- STARTFL - *lowest priority*

For example, when both RESETFL and STARTFL are ON, the TIMER executes the Reset command and nothing else will happen until RESETFL goes Off. This leaves the STARTFL as the only Boolean command ON, at which time the TIMER is started.

If you use both methods for issuing commands to the TIMER at the same time, the same priority described above for the flags also applies for the commands. For example, if STARTFL is ON and a Stop command is given (through COMMAND), the Stop command is executed and all lower priority command flags are automatically turned OFF

TIMER parameters



REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the TIMER function block.

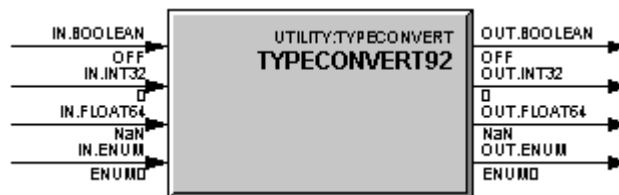
TYPECONVERT Block

Description

The TYPECONVERT block provides the ability to convert one data type to another for connecting parameters of different data types. It supports data type conversions for all combinations among the following major data types.

- Boolean
- Integer (signed 8-bit integer and unsigned/signed 16/32-bit integers)
- Real (32-bit and 64-bit IEEE floating point numbers)
- Enumeration

It looks like this graphically:



Function

The TYPECONVERT block is used to connect one input parameter to one or many output parameters with different data types. For example, a Boolean input (IN.BOOLEAN) can be converted to a 32-bit integer (OUT.INT32), a 64-bit floating point number (OUT.FLOAT64), and an enumeration (OUT.ENUM) outputs. The general Control Builder configuration rule about only connecting parameters of the same data types for block inputs and outputs still applies. The TYPECONVERT block reads the input value and only provides the converted output when the block connected to its output runs.

The block supports these user configurable attributes.

- A configurable Threshold Value (THRESHOLD) which lets you define how the Boolean value is to be interpreted for a 32- or 64-bit floating point to Boolean conversion. If the floating point input (IN.FLOAT32/IN.FLOAT64) value is greater-than or equal-to the configured THRESHOLD, the Boolean output (OUT.BOOLEAN) is turned ON, otherwise, it is OFF.
- A configurable Truncate Option (TRUNCATEOPT) which lets you specify whether the converted integer value is to be truncated or rounded for a 32- or 64-bit floating point to 32-bit integer conversion. For example, if the 64-bit floating point input (IN.FLOAT64) is 3.57, a rounded 32-bit integer output OUT.INT32) value would be 4 and a truncated OUT.INT32 value would be 3. If the IN.FLOAT64 value were 3.49, the rounded OUT.INT32 value would also be 3.
- A configurable Value OFF mapped to Enumeration (BOOLVALUEOFF) which lets you select a given enumeration to be mapped to a Boolean (ENUMBOOLMAP[n]) value of OFF.
- A configurable Value ON mapped to Enumeration (BOOLVALUEON) which lets you select a given enumeration to be mapped to a Boolean (ENUMBOOLMAP[n]) value of ON
- An Enumeration to Boolean Map scroll box lets you configure a given enumeration (ENUMBOOLMAP[n]) to OFF or ON.
- An Enumeration Text scroll box lets you configure up to 12 characters for a given Self Defining Enumeration output (OUT.SDENUM[n]).

Execution status

The block's execution status (EXECSTS) parameter monitors the status of the data type conversion and provides the following status values.

- OK = Successful. This means inputs were brought and the conversion was done without any error or clamping.
- Warning = Function was completed, but with some limitation. For example, the value was clamped after data type conversion.
- Error = Data is Bad. This could simply mean the input block does not exist or the connection to the input block is lost.

Input/Output

The block has up to nine inputs and nine outputs. The pins for the four most common inputs (IN.BOOLEAN, IN.INT32, IN.FLOAT64, IN.ENUM) and outputs (OUT.BOOLEAN, OUT.INT32, OUT.FLOAT64, OUT.ENUM) are exposed by default. But, all block pin parameters are available to be exposed and connected to using Control Builder graphical connections.

TYPECONVERT parameters



REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the TYPECONVERT function block.

Utility Functions
TYPECONVERT Block

Sequential Control

SCM (Sequential Control Module) Block

Description

The SCM (Sequential Control Module) block is a container module for sequences of [STEP Block](#) and [TRANSITION Block](#) grouped by specific [HANDLER Block](#). It gives instructions to regulatory and discrete control devices through Control Modules, which provide the direct interface to process equipment. It looks like this graphically.

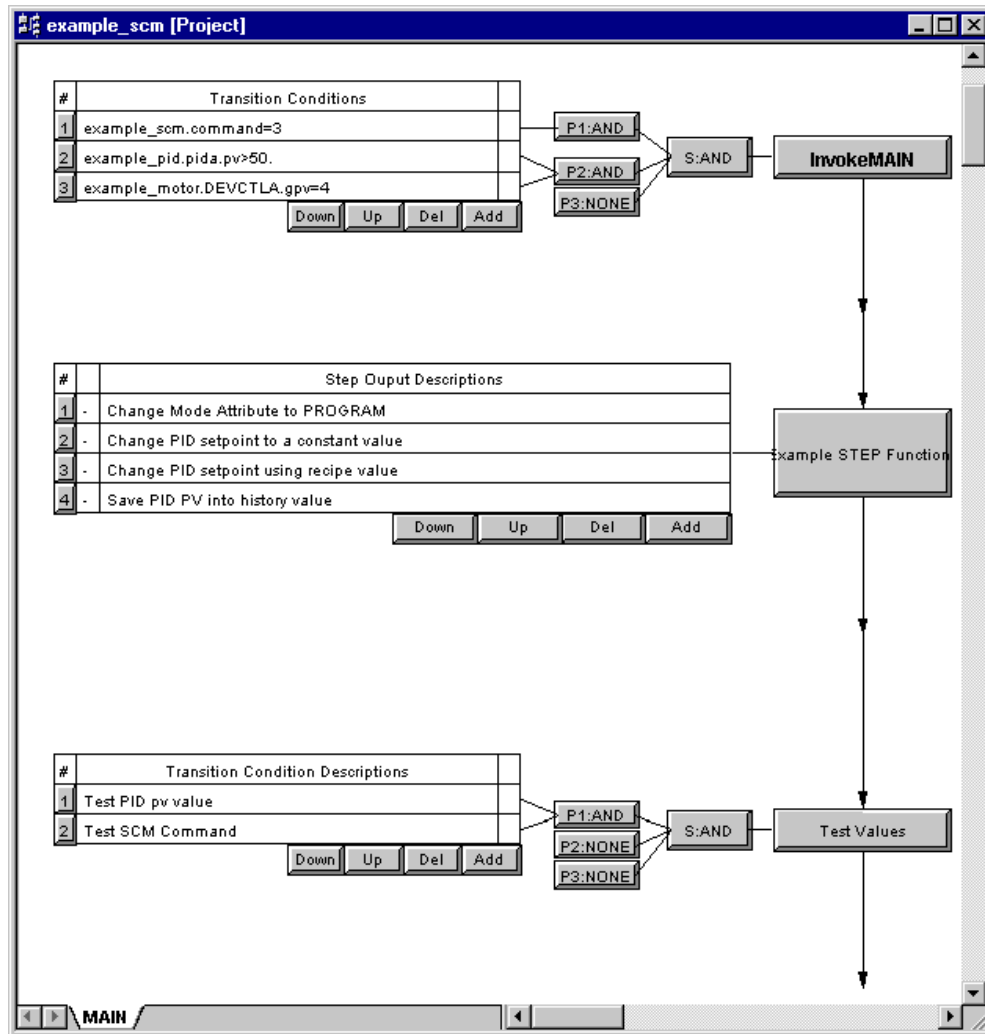


ATTENTION

You can implement a “Common” SCM functionality. This is just an extension to the basic SCM block to enhance operation for batch applications. Please refer to the section [Common SCMs](#) for more information about this functionality. The basic SCM information in this section and the others that follow still applies.

Sequential Control

SCM (Sequential Control Module) Block



ATTENTION

The SCM block may only contain its own components (that is, HANDLER, STEP, and TRANSITION blocks); it cannot contain other function blocks such as PID or logic blocks. The Control Module block is the container for the other function blocks.

Each SCM 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.

Configuration Tab	Description
Main	- Name – Unique SCM name of up to 16 characters. - Description (DESC) – SCM descriptor of up to 24 characters. - Eng. Units (EUDESC) – Text string of up to 7 characters defines how values associated with configuration appear on detail and group displays, and any generated reports. For example, DEGF could be used to represent temperature values in degrees Fahrenheit. - Keyword (KEYWORD) – Lets you enter a keyword descriptor of up to 16 characters to be used on detail and group displays. - Enable Alarming Option (ALARMOPT)– Lets you select if SCM block alarm function is to be enabled or not. - Order in CEE (ORDERINCEE) – Specifies the execution order of the SCM in the CEE timing cycle relative to other blocks scheduled to run in the same cycle. Enter as a number between 0 to 65535 in multiples of 10. The default value is 10. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. - Execution Period (PERIOD) – Specifies the scan period for the SCM block in milliseconds. Settings are 50, 100, 200, 500, 1000, and 2000 milliseconds with 1000ms being the default. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. - Execution Phase (PHASE) – Specifies the Phase for SCM block execution within the CEE execution cycle. There are 40 timing cycles in every execution cycle, which translates into 0 to 39 execution phases. Refer to the Function Block Execution Schedules section in the beginning of this document for more information. - Unit Text (UNITTEXT) – Lets you enter a unique two-

Configuration Tab	Description
	character text string to identify the “unit” that the Control Module or Sequential Control Module is part of. Version – Unique 7 character string for SCM block. You are free to use this however or in whatever form you choose.
Handlers	Lets you select the desired configured Handler of a given type that is to be enabled. You can configure multiple Handlers of any given type, but you can only enable one Handler of any given type at a time. The configurable Handler types are: - Main – Contains the primary sequential activities of the process. - Checking – Contains the sequential activities that need to be executed prior to the Main Handler. The Check Handler executes as soon as you activate the SCM upon initial entry. The SCM returns to the Check Handler when its conditions are met upon completion of a Main, Stop, or Abort Handler. You can also issue a Reset command to get the SCM to return to the Check Handler. - Interrupt – Contains sequential activities for normal exception processing. It acts like a subroutine of the Main Handler and interrupts the Main Handler activities. - Restart – Contains sequential activities for returning operation to the Main/Interrupt Handler after completion of a Hold Handler. - Hold – Contains the sequential activities to preempt the Main and/or Interrupt Handlers activity. A Stop or Abort Handler can preempt a Hold Handler. A Hold Handler can transition to a Restart, Stop, or Abort Handler. - Stop – Contains the sequential activities to preempt Main/Interrupt, Restart, or Hold Handler. An Abort Handler can preempt a Stop Handler. A Stop Handler can transition to an Abort, Check, or Hold Handler. You can also issue a Hold command to get the SCM to go to the Holding/Held state from the Stopped state.

Configuration Tab	Description
	- Abort – Contains the sequential activities to preempt Main/Interrupt, Restart, Hold or Stop Handler. No other Handler can preempt an Abort Handler. An Abort Handler can only transition to a Check Handler when all of the Check Handler conditions are met or you issue a Reset command. Note that every SCM includes the following internal Handlers, which are not user configurable. - Null – Contains sequential activities the SCM uses in place of a Check, Interrupt, Hold, Restart, Stop or Abort Handler that is commanded but not configured. For example, upon completion the Main Handler, the SCM returns to the Idle state through the Check Handler. If there is no configured Check Handler, the SCM uses the Null Handler to return to the Idle state. - Edit – The SCM executes the Edit Handler when it transitions from the Inactive state to the Validated state. The SCM validation checks for configuration errors in all Handlers, Steps, and Transitions in the SCM. Before starting the SCM, you should check the configuration status of the SCM and its components while it is in its Validated/Idle state. See the Exception handling section for more data about the configurable Handlers.
Alarm	Specifies alarm priority and severity settings for STEP duration timeout, Hold state, Stop state, Abort state and Fail state. Specifies access level for SCM access locks. Refer to SCM access level section later in this section for more information.

Configuration Tab	Description
<i>Recipe</i>	Specify configuration for up to 50 recipe parameters per SCM block. Each parameter includes the following configuration data. • Descriptor – Description of up to 23 characters. (RECDESC[1..50]) • Default Target Range – Specify range in percent to be used to automatically determine the low and high target values based on the entered target value. • Use Default Target Range(TGTRNGDEFOPT) – Select whether you want to use a default range to automatically determine the low and high values for the specified target value or not. If not, you must enter low and high values individually. • Target Value – Specify real number for recipe parameter target value. • Target Hi – Specify real number of the maximum value of the Target Value or use default value, if enabled. • Target Lo – Specify real number for the minimum value of the Target Value or use default value, if enabled. • Matl Code – Specify integer that represents the material ingredient of the recipe parameter. • Scale – Select whether parameter can be scaled (Yes) or not (No). If not, the RECSCALE[1..50] parameter is set to Off.

Configuration Tab	Description
History	Specify configuration for up to 50 history parameters per SCM block. Each parameter includes the following configuration data. • Descriptor – Description of up to 63 characters. (HISTDESC[1..50]) • Type – Specify type of history parameter using up to 11 characters. You can enter this string or it can be determined by a batch application, if applicable. Some examples of history types are Location (where the ingredient is located), Inventory (how much was taken from stock), and Actual Value (some other recorded value for the process). • Parameter Value – SCM saves history parameter value as a real number.
Aliases	Enable Alias Configuration (ALIASOPT) – Lets you select whether or not an Alias Table will be configured in the SCM. When enabled, the SCM functions as a Common SCM. The default selection is unchecked or no Alias Table. Number of Aliases (NUMALIASES) – Defines the number of aliases in the Alias Table. Number of Instances (NUMINSTANCES) – Defines the number of instances in the Alias Table. Instance Selected (INSTSELECT) – Represents the selected instance column. Alias Table Parameters – Please refer to the Alias Table section in the <i>Common SCMs</i> section for more information.
Server	Specify server parameter configuration associated with displaying SCM data on standard point and group detail displays as well as custom schematics.
Status	Lets you view run-time status of SCM during Control Builder monitoring. You can also make run-time changes in the SCM Command and/or Mode.

Functional Overview

- The Sequential Control Module block consists of the Main Handler, two Internal Handlers, and the configured Exception Handlers, which are all, made up of TRANSITIONs and STEPs.
- TRANSITIONs evaluate to a Boolean result that when True, allows the Main or Exception Handler to continue to the next STEP. STEPs usually execute Output Actions and also define other properties.
- TRANSITIONs and STEPs follow these basic principles of the IEC 1131-3 standard for sequences:
 - Two STEPs shall never be directly linked; they shall always be separated by a TRANSITION.
 - Two TRANSITIONs shall never be directly linked; they shall always be separated by a STEP.
 - A TRANSITION in the sequence occurs, when a TRANSITION becomes true and all preceding STEPs directly connected to that TRANSITION are active.
 - If a TRANSITION occurs, it activates the following STEP.
- An SCM will exit on any STEP or TRANSITION that has nothing wired to its output.
- Single thread branching (Selection divergence/convergence - one path of many is chosen) is supported.
- Two Internal Handlers which are **not** user configurable are supported as follows:
 - Null
 - Edit
- Two types of Normal (routine or desirable) Exception Handlers are supported as follows:
 - Check
 - Restart

- Four types of Abnormal (undesirable) Exception Handlers are supported as follows:
 - Interrupt (Lowest Priority)
 - Hold
 - Stop
 - Abort (Highest Priority)
- Check and Restart Handlers allow you to specify conditions for transitioning SCM from Checking to Idle and from the Held state back to Running, respectively. If you do not use a Check Handler, the SCM transitions directly to the Idle state. You can only invoke the Restart Handler by command.
- Exceptions can be invoked either by common conditions, STEP specific conditions, operator command or user program command.
 - STEP specific exception invoke conditions are supported only for the Stop and Interrupt Exceptions.
- You can configure multiple Exception Handlers of each given type, but you can only select/enable one of a given type at a time.
 - For Hold, Stop, and Abort, the Exception Handler is optional.
 - For these, if no Exception Handler is specified or none is currently enabled, then only the SCM-State is changed. It is assumed that the device drivers will drive the devices into appropriate states based on the SCM-State.
- SCM supports a Restart Address function for both the Main and Interrupt Handlers. This function identifies the STEP where SCM execution is to resume upon the return from an Exception Handler.
- Each handler must begin with a TRANSITION, which must be set as the Invoke TRANSITION. The Invoke TRANSITION block in the MAIN HANDLER of the SCM block provides the Start Conditions for the SCM.
- Three types of alarms are supported as follows:
 - Step time-out – You can configure a time-out value for each STEP.
 - Abnormal state – You can enable alarms related to Hold, Stop, and Abort states.
 - SCM EXECSTS of Fail
- SCM commands and states let you direct and monitor execution.

Sequential Control

SCM (Sequential Control Module) Block

- Access locks let you define the authority level needed to perform certain SCM operations.
- SCM supports up to 50 configurable recipe parameters and 50 configurable history parameters per SCM block.
- SCM can provide commands directly to the regulatory and device control FBs in the assigned Control Module.

More information about many of the functions covered in this overview can be found in the following sections.

Recipe and history support

The SCM supports up to 50 parameters each for recipe and history data per block. You can use this data to send information to the SCM prior to execution or upload actual values upon completion of the SCM execution. The major characteristics of this support are summarized below.

- Each recipe parameter and history parameter includes their own complement of configurable SCM parameters.
- Recipe target and history parameter values use real number data.
- The STATE of the SCM does not restrict access to recipe target and history parameter values. For example, the SCM accepts stores of recipe targets while it is Running, not just when it is in Checking or Idle.
- SCM state transitions have no effect on the recipe and history data. For example, a Reset command has no effect on the history parameter values, since they are preserved from one execution to the next.

Configuration example

Figure 43 and its companion callout description table shows a sample SCM configuration for quick reference. The view in Figure 43 depicts a configuration in the Project tab.

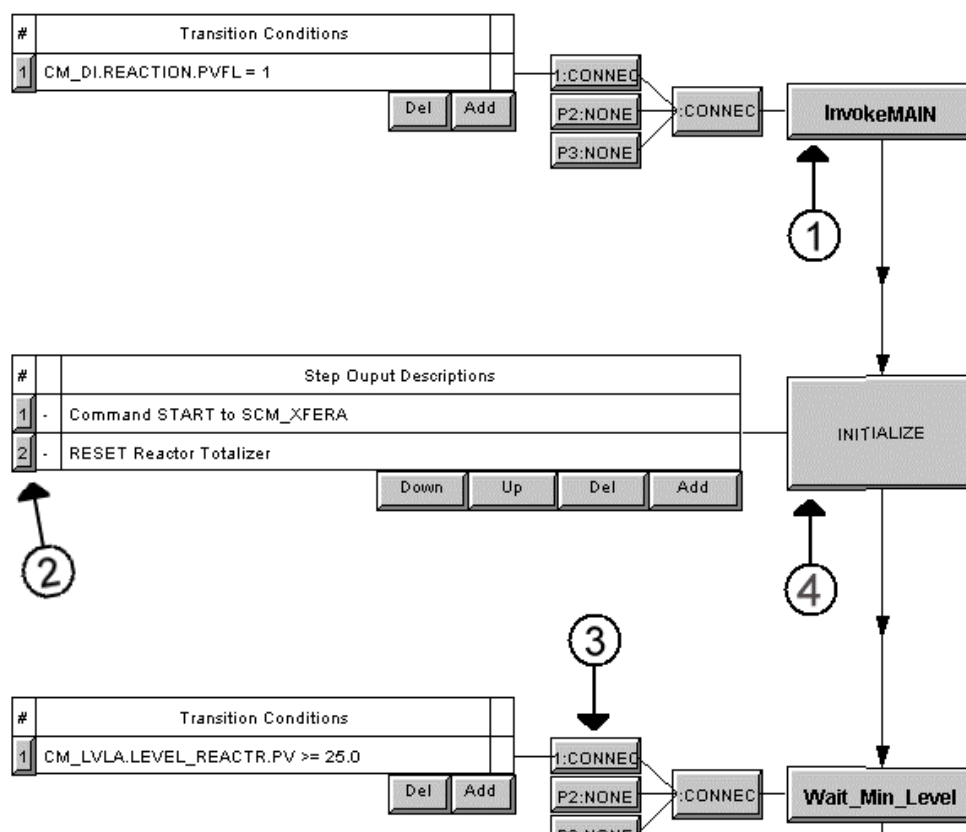


Figure 43 Example of CB configuration for simple SCM function.

Sequential Control

SCM (Sequential Control Module) Block

The following table includes descriptions of the callouts in Figure 43.

Callout	Description
1	You must configure the first transition to be an "Invoke Transition". In Experion PKS software version R110 or greater, a default invoke transition is preconfigured in every Handler selected for configuration.
2	The second Step output is a command to a "Level 1" control device in a CM.
3	You must verify that the gate configuration is legal for the number of conditions attached to it. In this example, the CONNECT gate type is used , since there is only one condition. The AND gate type could also have been used.
4	Since the second Step output is to a Level 1 device, the Control Builder automatically creates an implicit (hidden) connection between the SCM STEP control request and the CM block request connections parameters as required This connection is required to command the control device when the output type is configured as SET or NOT STORED. If the output type is configured as NULL, this connection is not required.

SCM parameters



REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the SCM block.

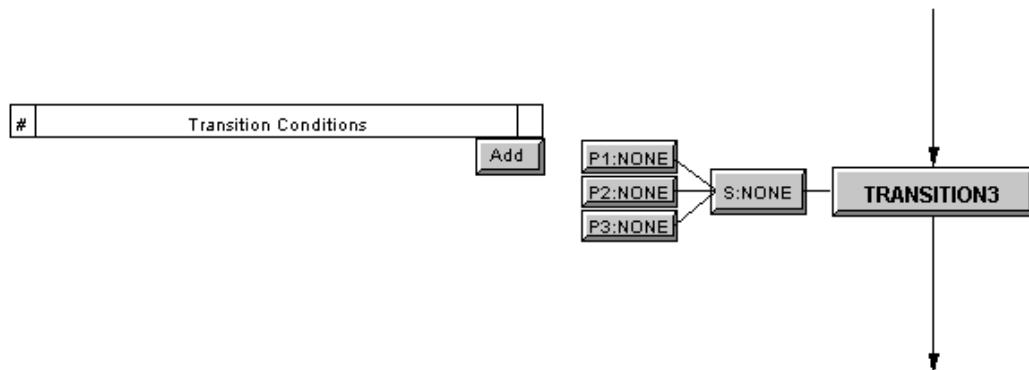
TRANSITION Block

Description

The TRANSITION block defines specific input conditions for a Handler.

- Input conditions of a TRANSITION block define a distinct process state which must be achieved before the output actions specified by the next STEP block can be performed.
- The input conditions grouped into a TRANSITION block are the condition expressions, which direct sequential execution flow. You can configure up to 10 conditions, which are logically connected using the three primary gates and one secondary gate.

It looks like this graphically in the SCM.



Each TRANSITION 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.

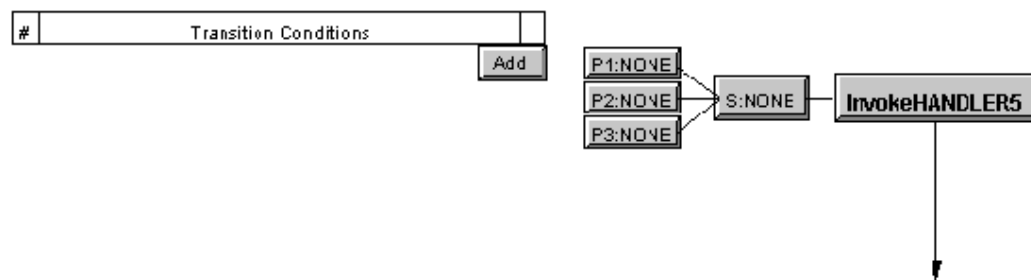
Configuration Tab	Description
<i>Main</i>	• Name – TRANSITION name of up to 16 characters long. Must be unique within the SCM block inclusive of all Handlers. • Description (DESC) – TRANSITION descriptor of up to 39 characters long.
<i>Condition</i>	Lets you specify details for given input conditions as they are added on the chart. This Tab only appears after you click on the “Add” button below the Transition Conditions box on the chart to add an input condition. Refer to the Standard Input Conditions section below for more information about configurable condition attributes. Note that the “Up”, “Down”, and “Del” buttons below the Transition Conditions box only appear as needed based on the number of expressions configured.
<i>Gates</i>	Select type of logic gate from three primary (P1, P2, and P3) and one secondary (S) gate. Refer to Input conditions below for more information about logic gate types.
<i>Block Pins</i>	Lets you select the available parameters that you want to expose as input/output pins on the function block graphic in Control Builder.
<i>Configuration Parameters</i>	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.
<i>Monitoring Parameters</i>	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.
<i>Block Preferences</i>	Lets you change several block-viewing preferences including the color of the block’s faceplate.

Function

Defines the distinct process state, which must be achieved to allow the SCM to advance to the control step (that is, the STEP block) so that it can perform the output actions specified. A transition acts like a big wait statement.

Default Invoke Transition

Every Handler must start with a transition. In systems running Experion PKS software version R110 or greater, a default invoke transition, as shown below, is preconfigured in every Handler selected for configuration. This invoke transition is configured with no conditions and all gates set to none which is considered a “null” transition.



If you use the default invoke transition as is, an operator must issue a start command every time the SCM is to execute.

Configuration example

Figure 44 and its companion callout description table shows a portion of an SCM configuration using a dual transition flow for quick reference. The view in Figure 44 depicts a configuration in Project tab.

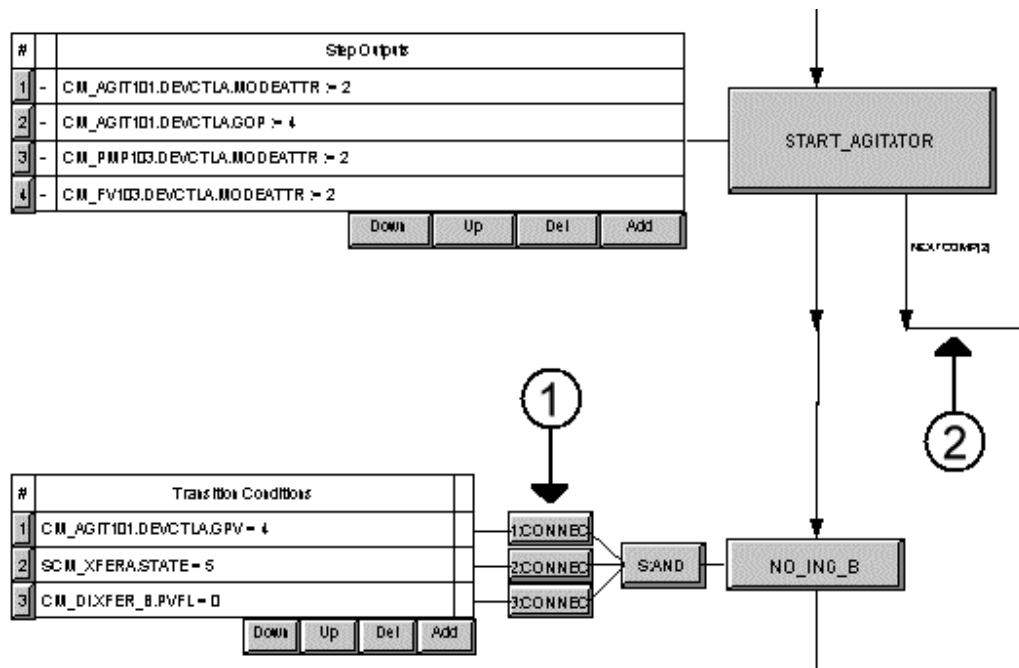


Figure 44 Example of CB configuration using dual transitions in an SCM.

The following table includes descriptions of the callouts in Figure 44.

Callout	Description
1	You must verify that the gate configuration is legal for the number of conditions attached to it. In this example, the CONNECT gate type is used, since there is only one condition. The AND gate type could also have been used.
2	You must configure the STEP block to include another component connection by configuring the block to add another NEXTCOMP[X] as an output pin. This lets you connect the STEP block to another TRANSITION block. The NEXTCOMP[1] pin is always exposed but not labeled for connection to one TRANSITION block.

Input conditions

A maximum of 10 standard input conditions are supported per SCM TRANSITION block.

Logic gates may be AND, OR, NAND, NOR, NOT, XOR, CONNECT, NONE, OFF, or ON.

- XOR must have two inputs. The XOR gate turns output OFF when an even number of inputs (2,4,6,8,10) are ON and turns output ON when an odd number of inputs ON. This action adheres to the IEC DIS 1131-3 standard. **But, this may not be the expected behavior for an XOR with more than two inputs.**
- CONNECT and NOT have only one input.
- NONE, ON, and OFF have no inputs.
- As shown in the transition logic diagram in Figure 45:
- Gates P1, P2, and P3 can have a variable number of inputs as long as the total for the three does not exceed 10. Three inputs maximum are allowed into the secondary gate S.

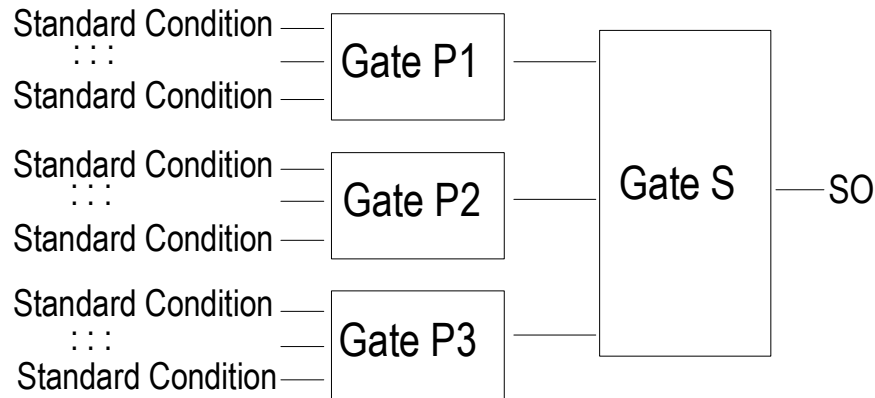


Figure 45 Transition logic diagram.

Standard input Conditions

Each standard input condition consists of the following user configurable attributes. The following table lists the given attributes that appear under a Condition “Tab” in the parameter configuration form along with a description of each attribute.

Attribute	Description
23-Character Descriptor	User defined
Condition expression	You can enter desired expressions into the conditions on the parameter configuration form. Expressions can evaluate to a Boolean value using a combination of arithmetic and logical operators, to an arithmetic value using arithmetic operators, or may simply specify any scalar value (Floating Point, Boolean, Enumeration) for comparison in a logical expression. Parameters of other blocks can be referenced as long as the block is already defined in the system database. Note that : • String data types are not supported. • Enumerations and Booleans are supported, but values must be entered as integers. For example: – cm2.pid1.mode = 2 (Mode is compared to Cascade)

Attribute	Description
Force permit	Allows operator with proper access authority or program to force this input condition, when selected (On). Default condition is Off or force is not permitted. You can assign the C.BYPPERM parameter as a block input to view the force permit setting for an input condition on the chart.
Force request	Specifies the value a condition is to assume when it is forced. Value selections are To On, To Off , and None. The default selection is None. You can assign the C.BYPREQ parameter as block input to view the force request setting for an input condition on the chart. The setting also appears in the far right-column of the conditions box on the chart as T, F, or –, respectively.
Logic gate	Identifies which gate P1, P2, or P3 is associated with the condition expression, as applicable.

Operators and Functions

Since the expression capability is exactly the same as that provided for the REGCALC block, Refer to Table 1 ([Expressions](#)) in this document for a list of the expression operators and functions this block supports for its condition expressions.

Memory optimization for SCM expressions

The system optimally allocates memory for the following SCM expressions rather than an assumed maximum size.

STEP Outputs

STEP Specific Stop Condition

STEP Specific Interrupt Condition

TRANSITION Conditions

The following table summarizes the limits for the various types of expressions supported in the Control Execution Environment for reference.

Sequential Control

TRANSITION Block

If Expression Type Is . . .	Then, Maximum Number of Characters Is . . .	And, Maximum Number of References Is . . .	And, Maximum Number of Bytes for Pcode Is . . .
SCM STEP Output	255	6	200
SCM STEP Specific Stop Condition	255	6	200
SCM STEP Specific Interrupt Condition	255	6	200
SCM Transition Condition	255	6	200
AUXCALC Function Block	255	6	100
REGCALC Function Block	255	6	100

Example condition expressions

The following table lists some example transition condition expressions along with their meaning for reference.

Example Condition Expression	Meaning
CM101.EXECSTATE = 1	CM101 execution state is Active.
SCM301.STATE = 7	SCM301 state is Idle
CM301.PIDA.MODE = 1	PIDA block in CM301 is in Manual Mode.
CM301.PIDA.MODEATTR = 2	PIDA block in CM301 has a Mode Attribute of Program.
CM201.DI04.PVFL	DI04 block in CM201 is ON
CM401.PIDA.PV <= 20	PIDA block in CM401 has a PV value of less than or equal to 20.
CM401.PIDA.PV >20.0 AND CM401.PIDA.PV < 70.0	PIDA block in CM401 has a PV value between 20.0 and 70.0.

Example Condition Expression	Meaning
CM501.PIDA.PV<= CM501.PIDA.SP + 5	PIDA block in CM501 has a PV value that is less than or equal to the value of its SP value plus 5.
CM201.DI02.PVFL AND CM201.DI03.PVFL	The PV flags for blocks DI02 and DI03 in CM201 are TRUE.
NOT(CM201.DI02.PVFL AND CM201.DI03.PVFL)	The negative of the PV flags for blocks DI02 and DI03 in CM201 is FALSE.
(CPMFB.HOUR = 8) AND (CPMFB.MINUTE = 0)	Checks for 8:00 a.m. every day.

Failure handling

The following table summarizes possible failure handling for a TRANSITION input condition.

If. . .	Then,
an error occurs during the evaluation of a TRANSITION Input Condition.	- The condition's C.EXECSTS changes to Fail, which in turn changes the SCM EXECSTS to Fail. The condition's C.EXECCODE specifies the actual error. - The SCM is already waiting for the TRANSITION to become satisfied so it can make the next STEP active.
the evaluation of the Failed TRANSITION input condition continues.	Failure recovery may occur in one of the following ways. - The problem is fixed and the failed input condition can now be successfully evaluated. - The condition status flag (C.FL) is recomputed. - The condition's C.EXECSTS changes to OK, which turn changes the SCM EXECSTS to OK. - The operator forces the input condition, which causes the following. - The condition's C.EXECSTS changes to Bypass, which may in turn change the SCM EXECSTS to Bypass. - The condition status flag (C.FL) is set to On or Off.

Sequential Control

TRANSITION Block

If. . .	Then,
	– Bypasses the evaluation of the input condition to allow the TRANSITION to become true , so the next STEP can become active. – The condition's C.EXECSTS changes to OK, which in turn changes the SCM EXECSTS to OK, depending on the successful evaluation of the TRANSITION following the activated STEP.

TRANSITION parameters



REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the TRANSITION function block.

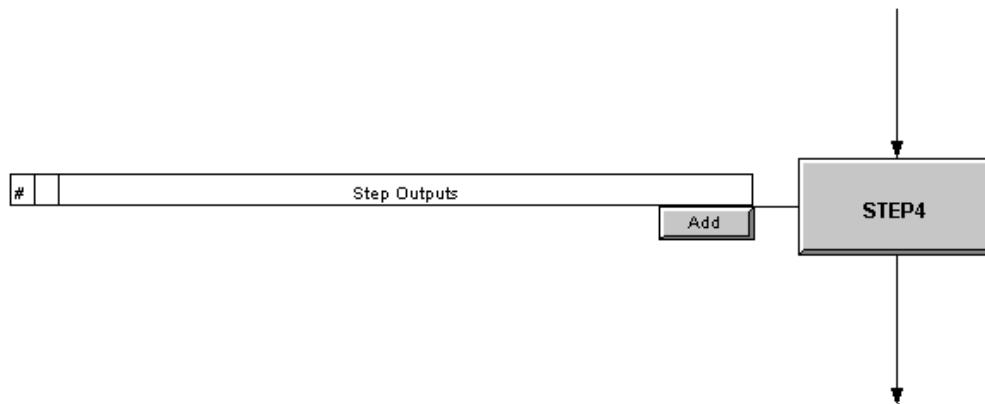
STEP Block

Description

The STEP block defines specific output actions.

- A specified output action usually generates a request to a control device to do something (for example, open a valve, start a pump, set furnace temperature).
- The source value of each output can be an expression (thereby enabling calculations in each output).

It looks like this graphically in the SCM. You can configure the display to show the Step Outputs on the left-hand or right-hand side of the STEP block.



Each STEP 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.

Configuration Tab	Description
Main	- Name – STEP name of up to 16 characters long. Must be unique within the SCM block inclusive of all Handlers. - Description (DESC) – STEP descriptor of up to 39 characters long. - Min Wait Time (MINTIME) – Specifies how long to delay evaluation of succeeding TRANSITION block after STEP outputs are stored. Time is entered in number of SCM execution cycles, which varies based on the configured SCM Execution Period. You can calculate the delay time by multiplying the Execution Period by the Min Wait Time setting. For example, If the Execution Period is 50 milliseconds and the Min Wait Time is 100, the start of the succeeding TRANSITION evaluation will be delayed by 5000 ms (5 seconds), which equals 50 ms times 100. You can assign the parameter MINTIME as a block input to view the Min Wait Time setting on the chart. - Max Active Time (MAXTIME) – Specifies how long the STEP can remain active before tripping the STEP time-out alarm. Time is entered in number of SCM execution cycles, which varies based on the configured SCM Execution Period. You can calculate the STEP time-out alarm trip point time by multiplying the Execution Period by the Max Active Time setting. For example, if the Execution Period is 100 milliseconds and the Max Active Time is 200, the time-out alarm will be tripped when the STEP remains active for more than 20,000 ms (20 seconds), which equals 100 ms times 200. You can assign the parameter MAXTIME as a block input to view the Max Active Time setting on the chart. - Update Restart Address (UPDRESOPT) – Lets you specify if the Restart Address option is to be ON or OFF for this STEP. When the Update Restart Address is selected (ON) for a STEP in a Main or Interrupt Handler, SCM execution will resume at the beginning of this STEP upon its return from a preemption caused by an Exception Handler. Refer to the following Restart address example.

Configuration Tab	Description
	You can assign the parameter UPDRESOPT as a block input to view the state of this option (On/Off) on the chart.
Out # n (1 to 16)	Lets you specify details for given outputs as they are added on the chart. This Tab only appears after you click the “Add” block below the Step Outputs box on the chart to add an output condition. Refer to the Output Contents section below for more information about configurable attributes. Note that the “Up”, “Down”, and “Del” buttons below the Step Outputs box only appear as needed based on the number of expressions configured.
Stop Condition	Lets you specify an optional Stop condition (SC.OPT) for the STEP when activated. When active, you can configure Condition Details that include a Condition Expression, Force Permit, and Force Request selection, which are very similar to the details configured for a TRANSITION input condition. You can assign the parameter SC.OPT as a block input to view the state of this option (On/Off) on the chart.
Interrupt Condition	Lets you specify an optional Interrupt condition (IC.OPT) for the STEP, when activated. When active, you can configure Condition Details that include a Condition Expression, Force Permit, and Force Request selection, which are very similar to the details configured for a TRANSITION input condition. You can assign the parameter IC.OPT as a block input to view the state of this option (On/Off) on the chart.
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.

Sequential Control

STEP Block

Configuration Tab	Description
Block Preferences	Lets you change several block-viewing preferences including the color of the block's faceplate.
	ATTENTION Always select the Update Restart Address for the first STEP in a Main Handler. This sets the UPDRESOPT parameter ON and ensures that the SCM always has somewhere to return to from Exception Handlers.

Restart address example

Figure 46 shows how the Update Restart Address is used to resume SCM execution at STEP1 in the Main Handler after preemption due to a Stop invoke condition during STEP2. In this case, the SCM executes the Stop Handler, transitions to stopping , then Stopped state, issues Hold command, executes Hold handler, transitions to holding, then Held state, issues Restart command and returns to STEP1 since its Update Restart Address flag is set to True and there is no Restart Handler.

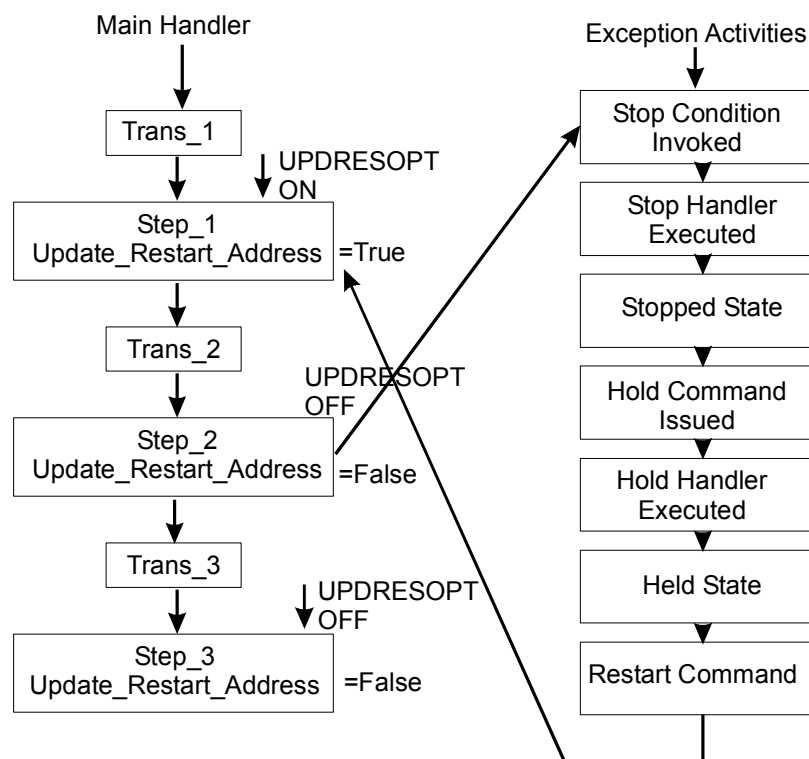


Figure 46 Example of using update restart option.

Restart address configuration considerations

The SCM parameters Restart Address (RESADDR[1..10]) and Restart Address Future (RESADDRFUTUR[1..10]) are updated by the Restart Option Address to tell the SCM which step in the Main or Interrupt Handler it is to return to upon completion of an Interrupt or Hold Handler. You can also use the partner integer parameters Restart Address (RESADDRN[1..10]) and Restart Address Future (RESADDRFUTRN[1..10]) to specify the return step through an output expression.



ATTENTION

- You can use the SCM Component Number (NUM) parameter to modify RESADDRN[X] and RESADDRFUTRN[X] parameters in expressions using only statements referring directly or indirectly to the Step, such as STEP_X.NUM.
 - Do **not** use absolute number constants in the expressions, since the NUM parameter could change after new downloads.
-

For example, you can use the RESADDRN[1..10] to set the Restart Address to the next return level as follows.

- An Interrupt Handler includes an expression to set the RESADDRN[1] parameter to a specified Step in the Main Handler. After the SCM transitions from Running to Interrupting, the Interrupt Handler updates the Restart Address to the specified Step to be executed in the Main Handler upon completion of the Interrupt Handler.
- A Hold Handler includes an expression to set the RESADDRN[1] parameter to a specified Step in the Main Handler. After the SCM transitions from Running to Holding, the Hold Handler updates the Restart Address to the specified Step to be executed in the Main Handler when the Restart command is issued upon completion of the Hold Handler.
- A Hold Handler includes an expression to set the RESADDRN[1] parameter to a specified Step in the Interrupt Handler. After the SCM transitions from Running to Interrupting to Holding, the Hold Handler updates the Restart Address to the specified Step to be executed in the Interrupt Handler when the Restart command is issued upon completion of the Hold Handler. In this case, the Hold Handler can only update the Restart Address in the Interrupt Handler, since it represents the next level up from the Hold Handler.

Note that you could use Flag and Numeric blocks in a CM to coordinate Restart Address functions between the Hold and Interrupt Handlers. Include expressions in the

Hold Handler to set the Flag block to let the Interrupt Handler know that it must update the Restart Address and specify the Step in the Main Handler in the Numeric block to update the Restart Address in the Interrupt Handler. Include a condition in the Interrupt Handler that checks the status of the Flag block to determine if it needs to update the Restart Address to the Numeric block value or not.

Use the RESADDRFUTRN[1] parameter to specify the Restart Address to be used for the current level in the future. For example, you want the Main Handler to restart at a Step based on process conditions instead of returning to the last Step that was executed per the normal Restart Address function. Include an expression in the Main Handler to set the RESADDRFUTRN[1] parameter to a specified Step in the Main Handler, so the Restart Address is set equal to the Restart Future Address after an SCM transition from Running to either Interrupting or Holding occurs.

Function

Organizes the output expressions of an SCM HANDLER at a specific stage of the HANDLER's execution.

Configuration example

Figure 47 and its companion callout description table show a portion of an SCM configuration identifying color-coded Step output indications for quick reference. The view in Figure 47 depicts a loaded configuration in Monitoring mode.

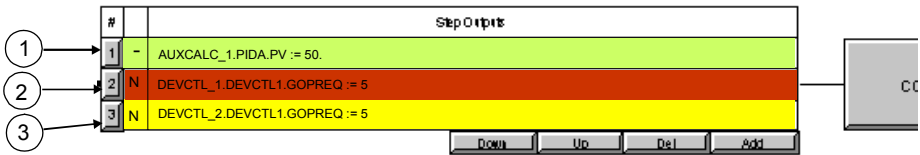


Figure 47 Example of CB configuration showing use of color coding in monitoring an SCM.

Sequential Control

STEP Block

The following table includes descriptions of the callouts in Figure 47.

Callout	Description
1	Green means that the Step output has been successfully completed.
2	Red means that the Step output has not been completed. Some possible reasons for a failed Step are: • The commanded device is not owned by the SCM. Check that SCM option has been correctly configured for the identified control device in the given CM. • The CM containing the control device is inactive. • Be sure the device's mode attribute (MODEATTR) is not set to OPERATOR instead of PROGRAM. The device's MODEATTR must be PROGRAM for the SCM to command it. • Be sure the mode of the device set properly to allow the command to succeed.
3	Yellow means that the Step output has been completed. But, this could mean that an attempted value store to a parameter exceeded the parameter's legal range. In this case, the parameter goes to its highest or lowest legal value.

Outputs

Up to 16 outputs may be defined per SCM STEP block.

The STEP block whose outputs are active is called the Active-STEP.

Output contents

Each single STEP block output supports the following configurable attributes. The following table lists the given attributes that appear under an Out # n “Tab” in the parameter configuration form along with a description of each attribute.

Attribute	Description
23-Character Descriptor	User defined
Output Expression	Supports one and only one assignment. Both sides of the assignment must be a calculation that results in a single value.
Output Type Option	Type defines how the destination device handles the SCM request. Following Output Types are supported: • Not-Stored (N) – A "control request" is posted to the destination FB and it remains active only as long as the current STEP remains active. This is the safest form of output since the device itself resets the control request from the SCM whenever: A new (other than the one that issued the request) STEP becomes active, <i>or</i> SCM transitions into or out of an abnormal state, <i>or</i> communication between the SCM and the device is interrupted. – Output destination point must be a DEVCTL (Device Control) function block configured for SCM-Option of other than None. The destination variable must be Generic Output (GOPREQ) for a DEVCTL block. • Set (S) – A "control request" is posted to the destination point and it remains active until explicitly overwritten. Normal operational situations such as operator intervention, Interlocks etc. do not overwrite the posted control request. – Output destination point must be a DEVCTL (Device Control) function block or a Regulatory Control type (AUTOMAN, PID, etc.) function block configured for SCM-Option of other than None. The destination

Attribute	Description
	variable must be one of the following depending on the given function block: Output (OPREQ) Generic Output (GOPREQ) (DEVCTL block only) Set-Point (SPREQ) SP-Target-Value (SPTVREQ) Mode (MODEREQ) • Null (–) – All variables other than the ones identified above fall into this category. The value is stored (subject to all the normal store restrictions applicable) and it remains there until overwritten. It can be easily overwritten during normal operational situations. – If the destination FB Variable is a control request (XXREQ) one that supports Set (Stored) or Not-Stored output type then Null is not allowed. However, an SCM can store a Null type output to the SP of a PID FB in a CM with an SCMOPT of None. Although, this configuration does not make use of any of the Level 1 device driver functions. You can assign the parameter OP.TYPE as a block input to view the Output Type setting on the chart. The Output Type setting is also displayed in the second column on the left side of the condition output box as N (Not Stored), S (Set), or – (Null).

Operators and Functions

Since the expression capability is exactly the same as that provided for the REGCALC block, Refer to Table 1 ([Expressions](#)) in this document for a list of the expression operators and functions this block supports for its output expressions.

Memory optimization for SCM expressions

The system optimally allocates memory for the following SCM expressions rather than an assumed maximum size.

STEP Outputs

STEP Specific Stop Condition

STEP Specific Interrupt Condition

TRANSITION Conditions

The following table summarizes the limits for the various types of expressions supported in the Control Execution Environment for reference.

If Expression Type Is . . .	Then, Maximum Number of Characters Is . . .	And, Maximum Number of References Is . . .	And, Maximum Number of Bytes for Pcode Is . . .
SCM STEP Output	255	6	200
SCM STEP Specific Stop Condition	255	6	200
SCM STEP Specific Interrupt Condition	255	6	200
SCM Transition Condition	255	6	200
AUXCALC Function Block	255	6	100
REGCALC Function Block	255	6	100

Example output expressions

The following table lists some example output expressions along with their meaning for reference.

Example Output Expression	Meaning
CM151.PIDA.OPX := 35.0, OR CM151.PIDA.OPX := 35.	Sets the output of the PIDA block in CM151 to 35 percent.
CM590.EXECSTATE := 1	Sets CM590 to Active state.
SCM301.COMMAND := 3	Commands SCM301 to Reset.
CM152.PIDA.MODE := 1	Sets the PIDA block in CM152 to MANual Mode.
CM152.PIDA.MODEATTR := 2	Sets the Mode Attribute of the PIDA block in CM152 to Program.
CIM191.DEVCTL.GOP := 5	Sets the output of the DEVCTLA block in CM191 to State 1 (ON).
SCM235.HISTVALUE[1] := (CM151.PIDA.PV + CM251.PIDA.PV)2	Stores the average of the PV values for CM151 and CM251 as first historical value.
ENBHANDLER[1] := 2	Enables the Main Handler that is second in the list of configured Main Handlers.
CM456.PID.SP := (CM456.SPREC1FLAG.PVFL) ? SCM457.RECTARGET[1] : SCM457.RECTARGET[2]	If the PV flag for the FLAG block in CM456 is ON, then set the SP for the PID block in CM456 equal to the number 1 recipe target value in SCM457. If the PV flag for the FLAG block in CM456 is OFF, then set the SP for the PID block in CM456 equal to the number 2 recipe target value in SCM457.

Output processing and failure handling

The processing of outputs includes:

- Storing the source value to the destination (for all outputs).
- Collecting store status response for each output.
- Providing separate Output Execution Status (OP.EXECSTS) for each of the 16 outputs. These possible status's are supported:
 - Ok => output was stored successfully.
 - Warning => output was stored but may be with some constraint.
 - Fail => output store failed for a reason such as invalid mode or state, or configuration mismatch. The Output Execution Code (OP.EXECCODE) defines the exact cause of this failure.



ATTENTION

On a Reset command to the SCM, the Output Execution Status (OP.EXECSTS) and the Output Execution Code (OP.EXECCODE) are initialized to Ok.

STEP parameters



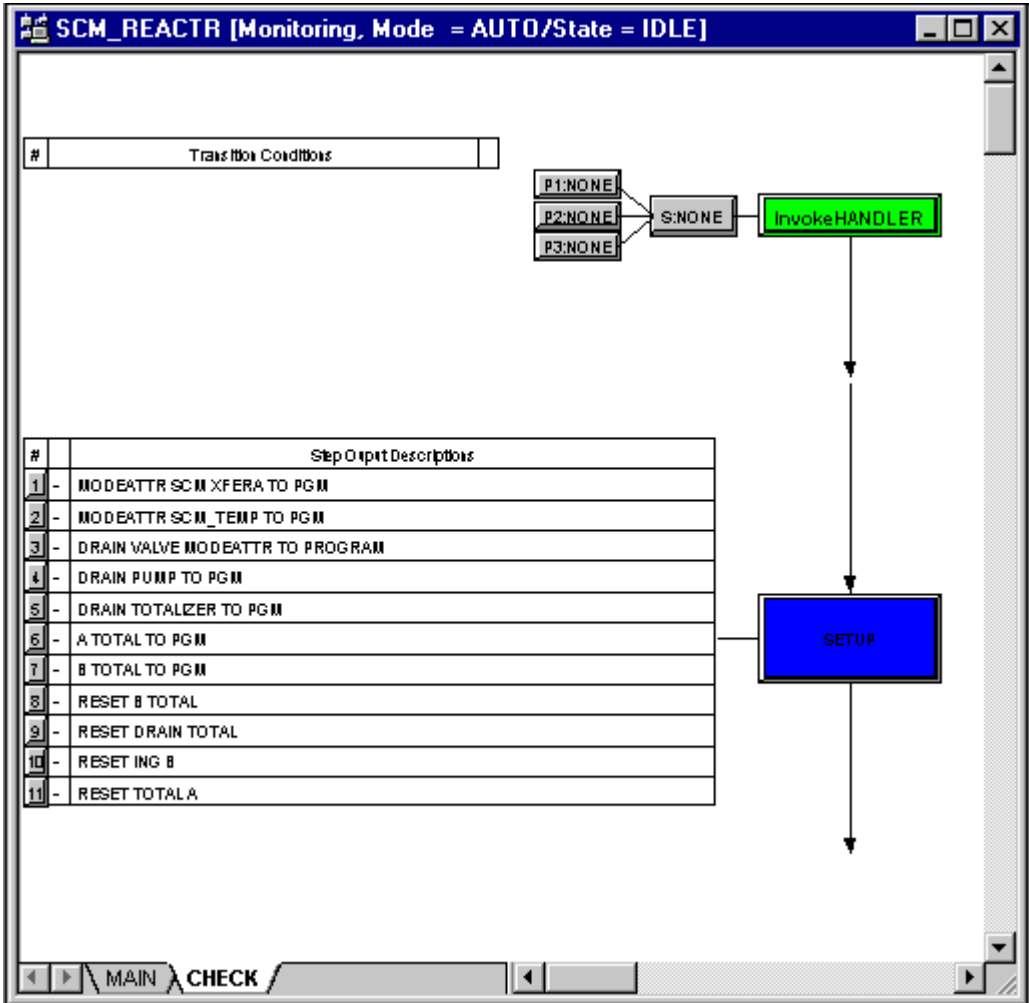
REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the STEP function block.

HANDLER Block

Description

The HANDLER blocks are represented as “Tabs” on the SCM chart. SCMs can contain several HANDLER blocks. Each HANDLER is an execution module that contains STEP and TRANSITION blocks. They look the same as the SCM block as shown in the sample graphic below.



Each Handler block supports the following user configurable attributes through the Main Tab on its parameters configuration form.

Attribute	Description
Name	Handler name of up to 16 characters long. Must be unique within the SCM block inclusive of all Handlers.
Description	Handler descriptor of up to 39 characters long.

Attribute	Description
<i>Handler Type</i>	Specify Handler as one of the following types. • Null (Not user configurable) • Edit (Not user configurable) • Main • Check • Interrupt • Restart • Hold • Stop • Abort

Functional Overview

Multiple HANDLER blocks may be contained within an SCM block, each modeled as a set of STEP and TRANSITION blocks, based on the following categories:

- Main Handler – for primary sequence SCM execution (lowest priority).
- Check Handler – for execution prior to Main sequence when desired.
- Interrupt Handler – for normal exception processing.
- Restart Handler – for execution on return to SCM State of Running from Held.
- Hold Handler – for processing of Hold exception.
- Stop Handler – for processing of Stop exception.
- Abort Handler – for processing of Abort exception (highest priority).

HANDLERs are selected through the Handlers tab in the SCM configuration form.

A HANDLER is invoked when

- its invoke conditions, modeled in its Invoke TRANSITION block, are met
- the SCM block is commanded to invoke the Handler (for example, the STOP command causes the STOP Handler to execute).

- STEP specific optional Stop Condition or Interrupt Condition is met which invokes the Stop Handler or the Interrupt Handler.

Configuration example

Figure 48 and its companion callout description table show how to add a HANDLER block to an SCM configuration for quick reference. The view in Figure 48 depicts a configuration in Project mode.

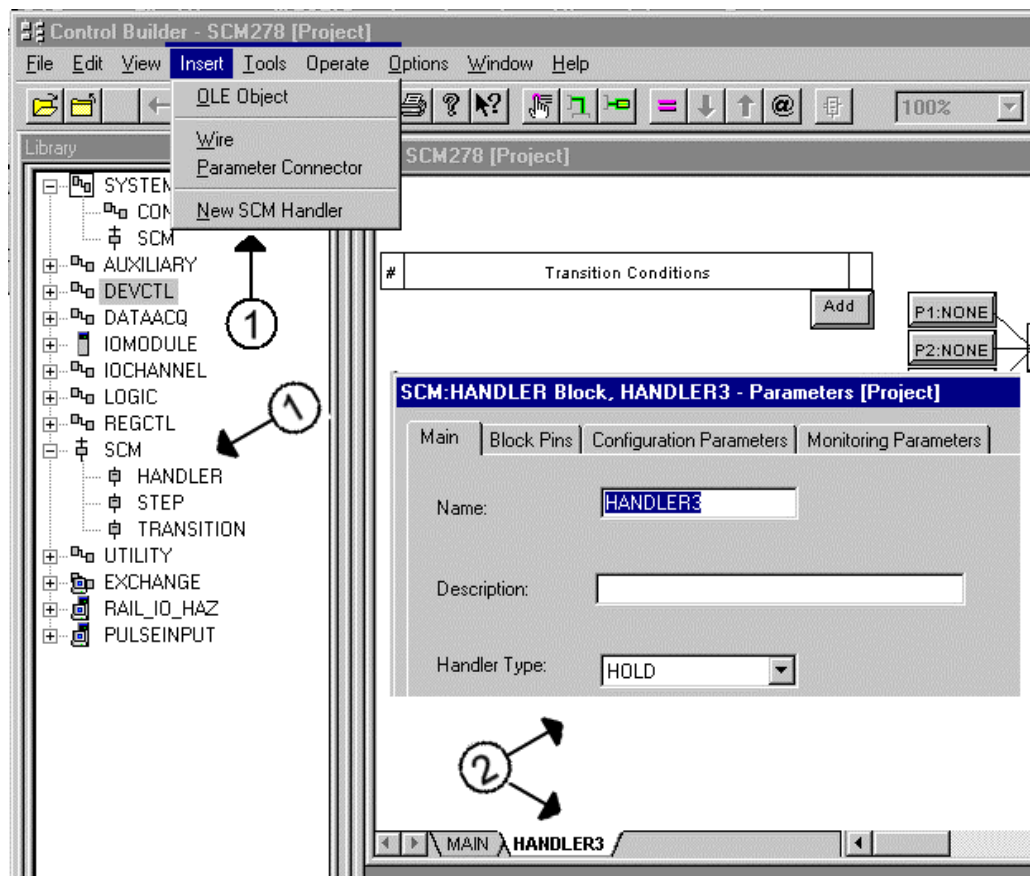


Figure 48 Example of adding a HANDLER to an SCM in a CB configuration.

Sequential Control

HANDLER Block

The following table includes descriptions of the callouts in Figure 48.

Callout	Description
1	You can add a HANDLER block to an open SCM chart by: - Clicking Insert->New SCM Handler and using menu to insert a new HANDLER. - Dragging HANDLER object from the SCM Library onto the SCM chart.
2	Double-click "HANDLER" tab at bottom of SCM chart to configure HANDLER block parameters.

Branching

The SCM lets you build HANDLERs for non-branching or single-thread branching execution.

As shown in Figure 49, the non-branching execution is the simplest form. It is a sequential ordering of TRANSITION and STEP blocks. In Figure 49, once input conditions in INVOKE1 become true, the trySPchg STEP is executed followed by an evaluation of TRANSITION9 input conditions. Execution continues sequentially until the last STEP is executed. If this is the Main Handler, the SCM state changes to Complete.

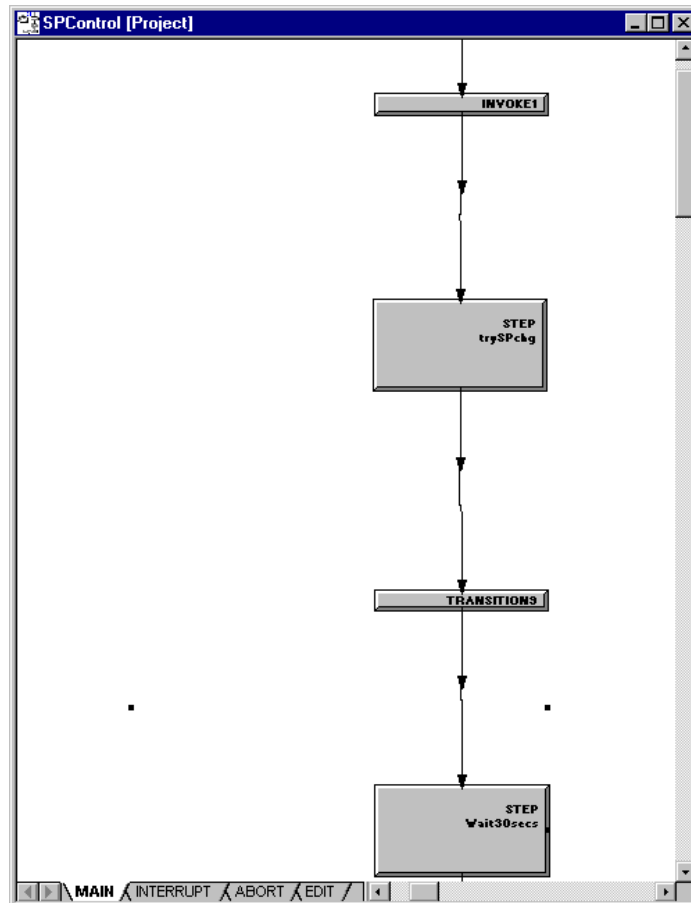


Figure 49 Example of non-branching configuration.

As shown in Figure 50, single-thread branching (Selection divergence/convergence) allows alternate action paths based on the process conditions or recipe parameters. In Figure 50, once one of the paths gated TRANSITIONs 13 and 14 is chosen, that path is executed to completion (assuming it is not preempted by a higher precedence HANDLER). TRANSITION17 is evaluated and once true, the SCM proceeds to the next STEP.

Sequential Control HANDLER Block

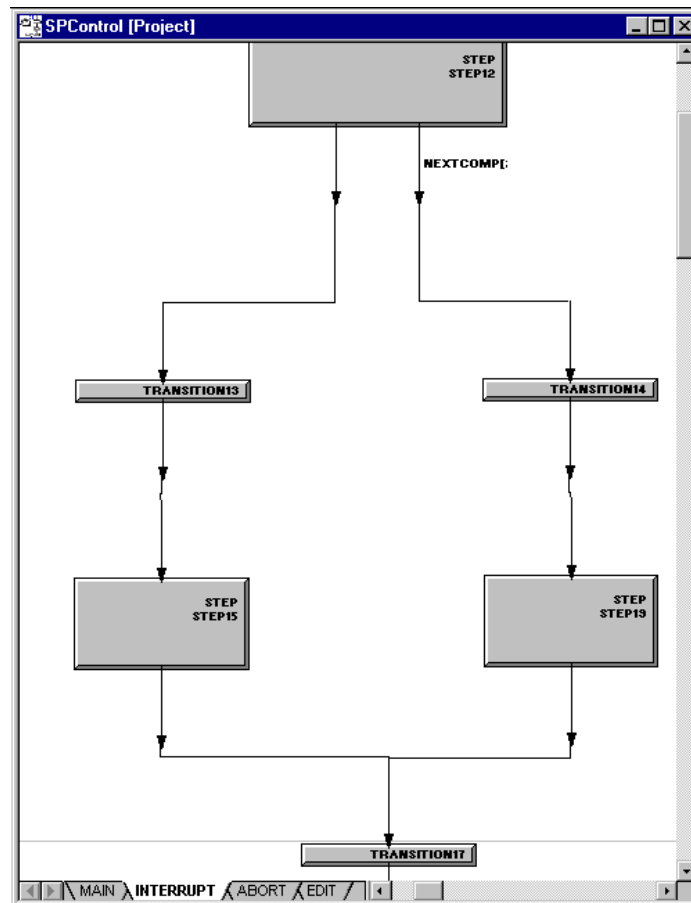


Figure 50 Example of single-thread branching configuration.

Looping

You can also create a looping construct as shown in Figure 51. If TRANSITION30 is chosen after the execution of STEP27, STEP33 is executed and the SCM loops back to TRANSITION26. This allows TRANSITION30 to test for some illogical condition that STEP 33 attempts to correct, and then the SCM loops back to choose either the TRANSITION28 or TRANSITION29 path for logical execution.



CAUTION

We recommend that you do not use a looping construct back to an Invoke TRANSITION.

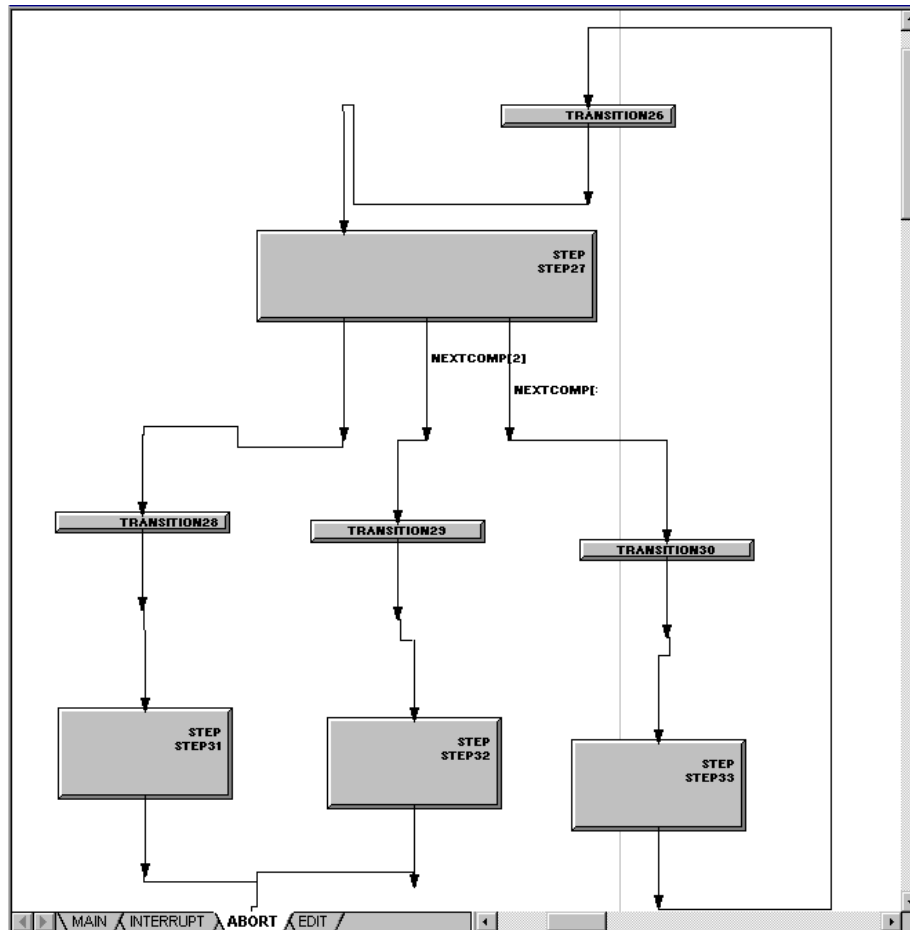
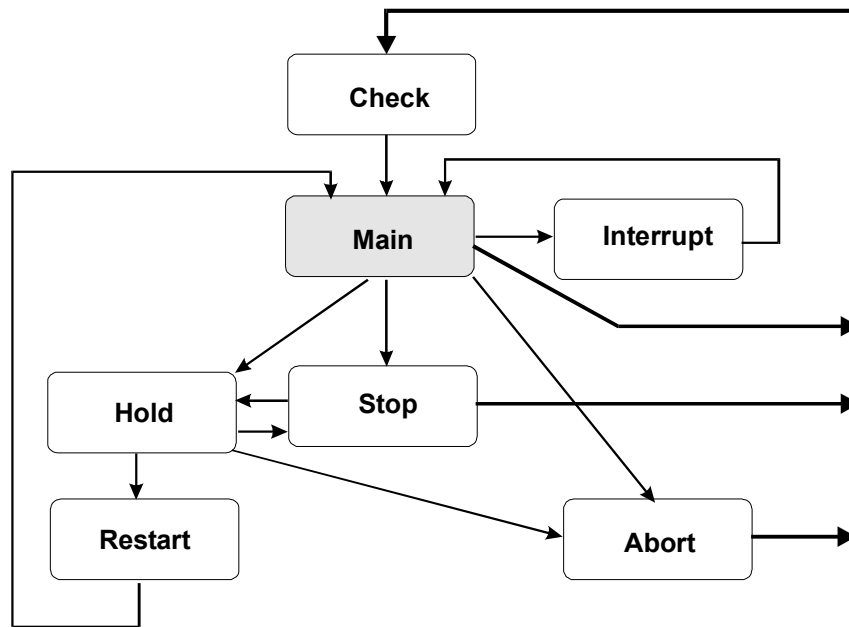


Figure 51 Example of looping construct configuration.

Handlers versus SCM states

The following table and block diagram summarize how the Handler being executed or completed determines the state of an SCM.

When SCM is in. . .	Then,
the Checking State,	the Check Handler is executing
the Idle State,	the Check Handler is complete.
Running or Complete,	the Main Handler is executing or complete.
Holding or Held,	the Hold Handler is executing or complete.
Interrupting,	the Interrupt Handler is executing.
Restarting,	the Restart Handler is executing.
the Stopping or Stopped State,	the Stop Handler is executing or complete.
the Aborting or Aborted State,	the Abort Handler is executing or complete.



Check Handler

The SCM executes the Check Handler just before entering the IDLE State. On initial entry to the SCM, the Check Handler is executed as soon as the SCM is activated. The SCM also returns to the Check Handler after a *RESET* command or when the conditions for the Check Handler are met after the STOP Handler, MAIN Handler, or ABORT Handler is completed. A configured Check Handler could be used to initialize process equipment and/or reset values for a new activity.

Main Handler

The Main Handler contains the primary sequential activities of the process. The Main Handler is the core of the SCM.

Interrupt Handler

The Interrupt Handler interrupts the activity of the Main Handler acting like a subroutine of the Main Handler. The Interrupt Handler begins executing when

- the Step's interrupt conditions are met, or
- the Invoke Transition's conditions are met, or
- an operator command is given.

When the Interrupt Handler completes, the program activity returns to the last step in the Main Handler that had the Update Restart Address Option ON. The Restart Address Option updates the SCM Resume address future parameter. The Resume Address and Resume Address Future are the two parameters of the SCM which determine the step to which the SCM will return to in the Main or Interrupt Handler.

The Resume Address Number and Resume Address Future Number parameters can be changed by an Output expression in an SCM. The SCM Component Number parameter can be used to modify the Resume Address Number or Resume Address Future Number using only statements that refer directly or indirectly to the Step, such as STEP_X.NUM.

If you are in the Main or Interrupt Handler and want to change the return location, then set the Resume Address Future. If you are in any other Handler and want to change the return location, then set the Resume Address. When you leave the Main or Interrupt Handler, the Resume Address is set equal to the Resume Address Future. For example, if the Interrupt Handler preempts the Main Handler, the Resume Address is set equal to the Resume Address Future and the Resume Address Future is updated according to each Step's update resume address option in the Interrupt Handler. If the Interrupt Handler is preempted, the Resume Address is set equal to Resume Address Future, and on Restart, the SCM will return to this Step in the Interrupt Handler.

Hold Handler

The Hold Handler preempts the activity of the Main and/or Interrupt Handlers. The Stop and Abort Handlers can preempt the Hold Handler. From the Hold Handler you can go to the Restart, Stop, or Abort Handler.

Restart Handler

The Restart Handler can only be initiated from the Held state. The Restart Handler returns the activity to the Main/Interrupt Handler at the Step designated in the Resume Address

Stop Handler

The Stop Handler preempts the activity of the Main/Interrupt, Restart, or Hold Handler. The Abort Handler can preempt the Stop Handler. From the Stop Handler, you can go automatically to the Abort or Check Handler or by command to the Hold Handler.

Abort Handler

The Abort Handler preempts the activity of the Main/Interrupt, Restart, Hold, or Stop Handlers. The Abort Handler can **not** be preempted. From the Abort Handler, you can only return to the Check Handler by command or by meeting the conditions of the Check Handler.

Null Handler

The Null Handler is **not** configurable. The Null Handler is the handler that the system uses when a Check, Interrupt, Hold, Restart, Stop, or Abort Handler is commanded yet **not** configured. For example, when a Main Handler is complete, it returns to Idle State through the Check Handler. If there is no Check Handler configured, the Null Handler is used.

Edit Handler

The Edit Handler is **not** configurable. The Edit Handler is executed while the SCM is in the Inactive state. The Edit Handler completes initialization activities and SCM validation. SCM validation looks for and identifies configuration errors in all Handlers, Steps, and Transitions of the SCM. When an SCM is in the Validated state, the configuration status of the SCM and its components should be checked by the user before starting / using the SCM.

HANDLER parameters



REFERENCE - INTERNAL

Refer to the *Experion PKS Control Builder Components Reference* for a complete list of the parameters used with the HANDLER function block.

SCM Interface and CM Interaction

Control module interaction

The SCM interacts with the Control Module to establish its interface with process control functions. The Control Module monitors information from the SCM and makes the information available to its contained regulatory and device control function blocks. The SCM also provides commands directly to the regulatory and device control function blocks, which provide a driver interface to final control elements in the process, in the Control Module. The SCM can use measurement type function blocks, such as Logic, Data Acquisition, and I/O blocks, but these blocks do not have a driver interface to the process components.

Figure 52 provides a functional block diagram overview of the SCM and Control Module interaction.

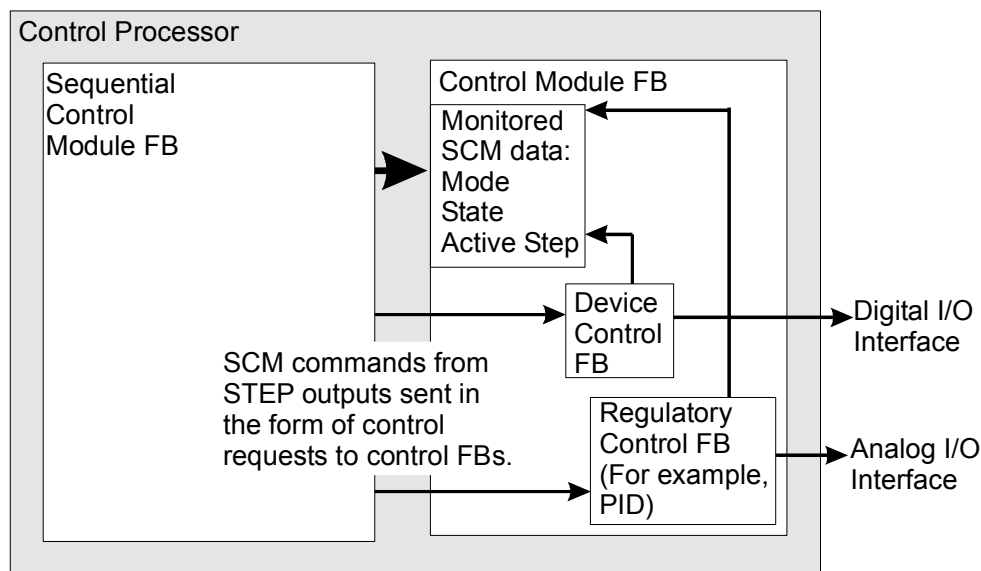


Figure 52 Functional overview of SCM and Control Module interaction.

Control module parameters

The Control Module (CM) parameters that define its relationship with an SCM FB are listed and described below. Some of these parameters are configurable through the Control Module configuration parameters form and the others are associated with monitoring SCM operation.

Parameter Name	Configuration Selections	Description
SCMOPT	None (0) Fixed (1) Dynamic (2)	SCM option is configurable on the CM configuration form. It determines the relationship between the Control Module and the SCM • If the selection is None, there is no relationship established between SCM and CM. The regulatory control and device control FBs contained in the CM will not accept Control Request commands from the SCM, but “normal” parameters are supported. . • If the selection is Fixed, only the SCM identified by the SCMNAME can send commands to the regulatory control and device control FBs contained in the CM. • The Dynamic selection is not supported at this time. If the Device Control FB is configured for no outputs (NUMDOUTS = 0), the FB will not accept SCM commands regardless of the SCMOPT selection.
SCMNAME	Valid SCM name	SCM Name is configurable on CM configuration form. It identifies the related SCM and it cannot be changed on-line. . This entry is not required if SCMOPT selection is None.
UNITTEXT	2-character string	Unit Text is configurable on CM configuration form. It associates the Control Module, and all events generated by its contained FBs, to a given equipment Unit.

Parameter Name	Configuration Selections	Description
SCMSTATE		For monitoring only, SCMSTATE identifies the state of the SCM, which determines the abnormal handling of the CM contained control devices. The CM monitors this data from SCM with given SCMNAME. The CM maintains this data for use by its contained control devices. An operator can also access this monitored data.
SCMMODE		For monitoring only, SCM Mode identifies the mode of the SCM, which determines the MODEATTR of the CM contained control devices. The MODEATTR is based on the contained control device's mode tracking option. The CM maintains this data for use by its contained control devices. An operator can also access this monitored data.
SCMASTEP[1...10]		For monitoring only, SCM Active Step identifies the current active step of the monitored SCM. This determines control activity related to the SCM STEP outputs, which send control requests to the CM contained control devices. For now, only SCMASTEP[1] is valid, since the SCM supports only single-thread branching at this time.

Regulatory control and device control parameters

Like the Control Module, the regulatory control and device control FBs contained in the CM have parameters that define their relationship with an SCM FB. These parameters are listed and described below. Some of these parameters are configurable through the given FB parameters configuration form.

Parameter Name	Configuration Selection	Description
MODETRACK	None(0) OneShot(1) ContRtn(2) SemiCont(3) Cont(4)	Mode Tracking is a configurable regulatory control or device control FB parameter. It defines how the regulatory control or device control FB will set the state of the MODEATTR based upon the MODE of the SCM. • If selection is None (No Tracking) and SCM mode changes from Manual to Auto, SemiAuto or SingleStep, the MODEATTR is not set to Program. There is no action, when the SCMMODE changes back to Manual. • If selection is OneShot (One Shot Tracking) and SCMMODE changes from Manual to Auto, SemiAuto or SingleStep or SCM STATE changes from Idle to Running, the MODEATTR is set to Program, but an operator can change it. MODEATTR returns to Operator , when SCMMODE changes back to Manual. • If selection is ContRtn (Continuous Tracking with Return) and SCMMODE changes from Manual to Auto, SemiAuto, or SingleStep or SCMSTATE changes from Idle to Running, the MODEATTR is set to Program, and an operator cannot change it while SCM remains in Auto. MODEATTR returns to Operator, when SCMMODE changes back to Manual. • If selection is SemiCont (Semi-Continuous Tracking) and SCMMODE changes from Manual to Auto, SemiAuto, or SingleStep or SCMSTATE changes from Idle to Running or SCMASTEP[1]

Parameter Name	Configuration Selection	Description
		changes while SCMMODE is Auto, the MODEATTR is set to Program, but an operator can change it. MODEATTR returns to Operator, when SCMMODE changes back to Manual. If selection is Cont (Continuous Tracking) and SCMMODE changes from Manual to Auto, SemiAuto, or SingleStep, MODEATTR is set to Program and an operator cannot change it while SCMMODE is Auto. MODEATTR does not change, when SCMMODE changes back to Manual; but an operator can overwrite it.
CONTROLREQ		The SCM STEP Outputs send Control Request information to regulatory control and device control FBs contained in the CM. This establishes specific control behavior for the control devices. If the CM SCMOPT is None, the request is rejected with an error. CONTROLREQ contains these elements of information: - The Name of the requesting SCM. If the Name in the Control Request does not match the configured SCM Name for the CM, the request is rejected and a Config Mismatch Error is returned. - The STATE of the requesting SCM. If the STATE in the Control Request is Running, the request is accepted, even if the Control Module SCMSTATE does not match. Otherwise, a Config Mismatch Error is returned. - The identity (ID) of the active STEP in the SCM thread sending the Control Request. A store to one of the XXREQ parameters must immediately follow the CONTROLREQ. Otherwise, the CONTROLREQ is ignored. The XXREQ parameter may only be stored by another FB, if it has been preceded by a store

Sequential Control
SCM Interface and CM Interaction

Parameter Name	Configuration Selection	Description
		to CONTROLREQ. If MODEATTR is Operator, an operator may override the XXREQs.
LASTMODEREQ		Last value of MODEREQ (Mode Request), when SCMMODE is Running.
LASTOPREQ		Last value of OPREQ (Output Request), when SCMMODE is Running.
LASTOPTYPE		Last value of OPTYPE (Output Type), when SCMMODE is Running.
LASTRATEREQ		Last value of SPRATEREQ (Set Point Rate Request), when SCMMODE is Running.
LASTSPREQ		Last value of SPREQ (Set Point Request), when SCMMODE is Running.
LASTSTEP		Last value of SCMASTEP (SCM Active STEP), when SCMMODE is Running.
MODEREQ		The requested MODE. The SCM CONTROLREQ wants to change the regulatory block's MODE.
OPREQ		The requested OP (Output) value. The SCM CONTROLREQ wants to change the regulatory control block's output (OP).
GOPREQ		The requested GOP (Generic Output) value. The SCM CONTROLREQ wants to change the device control block's generic output (GOP).
OPTYPE		Output request Type (not configurable). Null (0) Set (1) Not Stored (2) Default (3) The OPTYPE parameter on the block holds the local copy of SCM step output OPTYPE. This parameter shows how long the current request can remain active. The meaning of OPTYPE and its different values can be stated as follows, • NULL - This means that the block does

Parameter Name	Configuration Selection	Description
		not have any outstanding requests. In this situation, the operator can not make any additional requests. But, SCM can command the device. The OPTYPE = NULL, if the request has never been made. The SCM or operator has canceled the request by storing NaN, None or Bad or the abnormal state has been reached for which the abnormal state option is None. • SET – This means that the request was made by an agent outside the device (operator or SCM) and remains active until canceled by the device or by an SCM. Request is persistent until SCM, Operator, or the device itself changes or cancels the request. • NOTSTORED – This means that the request was made by an SCM step and when the step is exited the device will make a SAFEOP request itself with no action from the SCM. When the SAFEOP request is made, the OPTYPE changes to DEFAULT, since the device itself made the request. Operators in this case can only anticipate the SAFEOP request or can cancel the request altogether. • DEFAULT - This means that the device made the request itself and an operator can only cancel the request. Thus, the OPTYPE = DEFAULT, if an abnormal state is reached for which an option was configured (including those covered by STARTOPT). Note that in a DEVCTL block, the OPTYPE = DEFAULT for all the cases, when GOPREQ = SafeOP after a NOTSTORED request. These cases are as follows, – Operator storing SafeOP – Monitored step change

Sequential Control
SCM Interface and CM Interaction

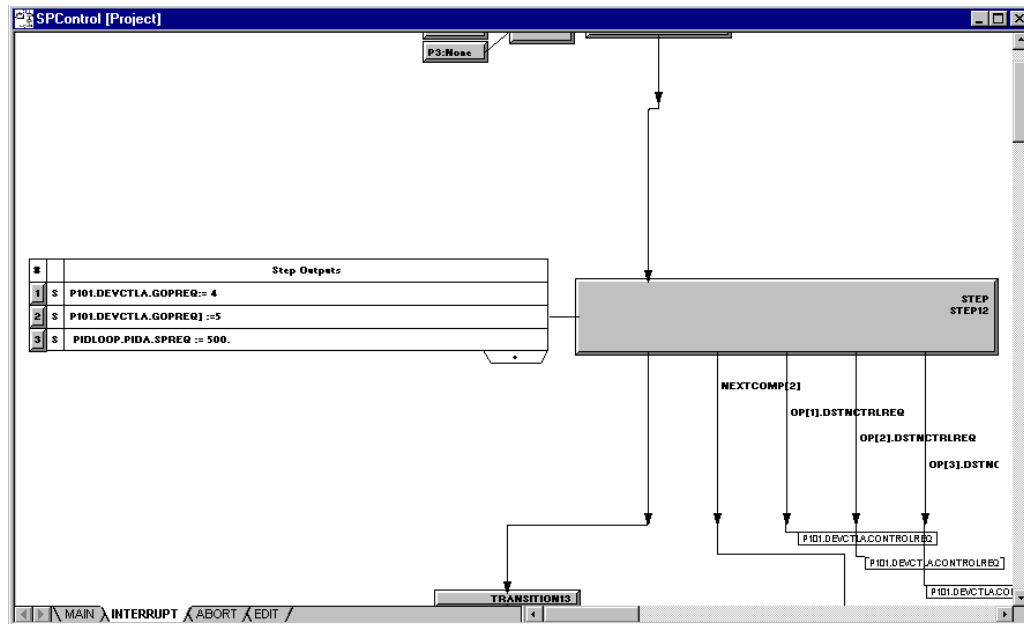
Parameter Name	Configuration Selection	Description
		– Monitored state change
SPREQ		The requested SP (Set Point) value. The SCM CONTROLREQ wants a step-change in the regulatory control block's SP.
SPTVREQ		The requested SP target value. The SCM CONTROLREQ sets a target value in regulatory control block's SP in preparation for an SP ramp request.
SPRATEREQ		The requested SP ramp Rate value. The SCM CONTROLREQ sets a ramp rate in regulatory control block's SPTVRATE for processing ramped SP target value.
STARTOPT	Regulatory Control type blocks: None (0) Man (1) Auto (2) Cas (3) FixedOP (4) HoldPV (5) FixedSP (6) RampedSP (7) DEVCTL blocks: None (0) SafeOp (1)	The Start Option is a configurable regulatory control or device control FB parameter. It specifies the action the control device takes, when SCMSTATE is Checking, Idle, or Complete.
STARTRATE	+ or – values, 0.0, NaN	The value used for setting SPRATEREQ, when STARTOPT is RampedSP.
STARTVAL	+ or – values, 0.0, NaN	The value used for setting OPREQ, when STARTOPT is FixedOP; or for setting SPREQ or SPTVREQ, when STARTOPT is Fixed SP or RampedSP.

Parameter Name	Configuration Selection	Description
HOLDOPT	Regulatory Control type blocks: None (0) Man (1) Auto (2) Cas (3) FixedOP (4) HoldPV (5) FixedSP (6) RampedSP (7) DEVCTL blocks: None (0) SafeOp (1)	The Hold Option is a configurable regulatory control or device control FB parameter. It specifies the action the control device takes, when SCMSTATE is Holding or Held. This option should be disabled, when sequencing actions are required as part of the SCM. In this case, the SCM should execute its HOLD Handler. The SCM can also disable this option.
HOLDRATE	+ or – values, 0.0, NaN	The value used for setting SPRATEREQ, when HOLDOPT is RampedSP.
HOLDVAL	+ or – values, 0.0, NaN	The value used for setting OPREQ, when HOLDOPT is FixedOP; or for setting SPREQ or SPTVREQ, when HOLDOPT is FixedSP or RampedSP.
STOPOPT	Regulatory Control type blocks: None (0) Man (1) Auto (2) Cas (3) FixedOP (4) HoldPV (5) FixedSP (6) RampedSP (7) DEVCTL blocks: None (0) SafeOp (1)	The Stop Option is a configurable regulatory control or device control FB parameter. It specifies the action the control device takes, when SCMSTATE is Stopping or Stopped, or Aborting or Aborted. This option should be disabled, when sequencing actions are required as part of the SCM. In this case, the SCM should execute its Stop Handler. The SCM can also disable this option.
STOPRATE	+ or – values, 0.0, NaN	The value used for setting SPRATERQ, when STOPOPT is RampedSP.
STOPVAL	+ or – values, 0.0, NaN	The value used for setting OPREQ, when STOPOPT is FixedOP; or for setting SPREQ or SPTVREQ, when STOPOPT is FixedSP or RampedSP.

Sequential Control
SCM Interface and CM Interaction

Parameter Name	Configuration Selection	Description
<i>RESTARTOPT</i>	None (0) LastReq (1)	The Restart Option is a configurable regulatory control or device control FB parameter. It determines what happens to the control device when SCMSTATE returns to a Restart or Running state after an abnormal state. • If RESTARTOPT is None, control device takes no action. In this case, a separate Restart Handler in the SCM is needed to command the control device, if SP, OP, or MODE changes are required. • If RESTARTOPT is LastReq, the control device copies the values used for the last Running state as follows: – MODEREQ = LASTMODEREQ – OPREQ = LASTOPREQ – SPREQ = LASTSPREQ – SPRATEREQ = LASTRATEREQ – SPTVREQ = LASTSPTVREQ
<i>LASTREQFL</i>	off on	The Last Request Flag is a regulatory control or device control FB parameter. This flag is true, when SCMSTATE is Stopping, Stopped, Holding, or Held; and RESTARTOPT is LastReq.

The STEP outputs may be configured to interact with parameters for PID and DEVCTL FBs in Control Modules configured for a Fixed SCM with a given name. Control Builder will automatically determine when to build implicit/hidden connections for SCM Level 1 output special connections.



Control devices mode reference

The MODE parameter for regulatory and device control FBs determines who may store to writable inputs and/or outputs for a given block. Since MODE changes may be permitted through external or safety interlock requests to regulatory control blocks, it is determined at runtime based on current conditions. The MODE for device control FBs is fixed at Manual (MAN).

The MODEATTR (Mode Attribute) parameter further defines who may store to a control device variable when the MODE is MAN or AUTO (Automatic). The main MODEATTR choices are Operator or Program. A MODEATTR of Program means that an SCM can store to applicable control device variables. For regulatory control blocks, the MODEATTR is determined at runtime based on current conditions.

The NORMMODE (Normal Mode) and NORMMODEATTR (Normal Mode Attribute) parameters let you define the “normal” MODE and MODEATTR values that can be invoked through a display or keystroke in the Station Application. The NORMMODE for device control FBs is fixed at MAN.

SCM execution overview

The main activities associated with the execution of an SCM are graphically summarized in Figure 53. The SCM evaluates exception invoked transitions, TRANSITIONs in the active Handler, and active STEP in active Handler; and sends Control Requests and information for monitoring to the Control Module. Basically, the SCM parameters of STATE and MODE show what the SCM is doing and how automatically it is being done.

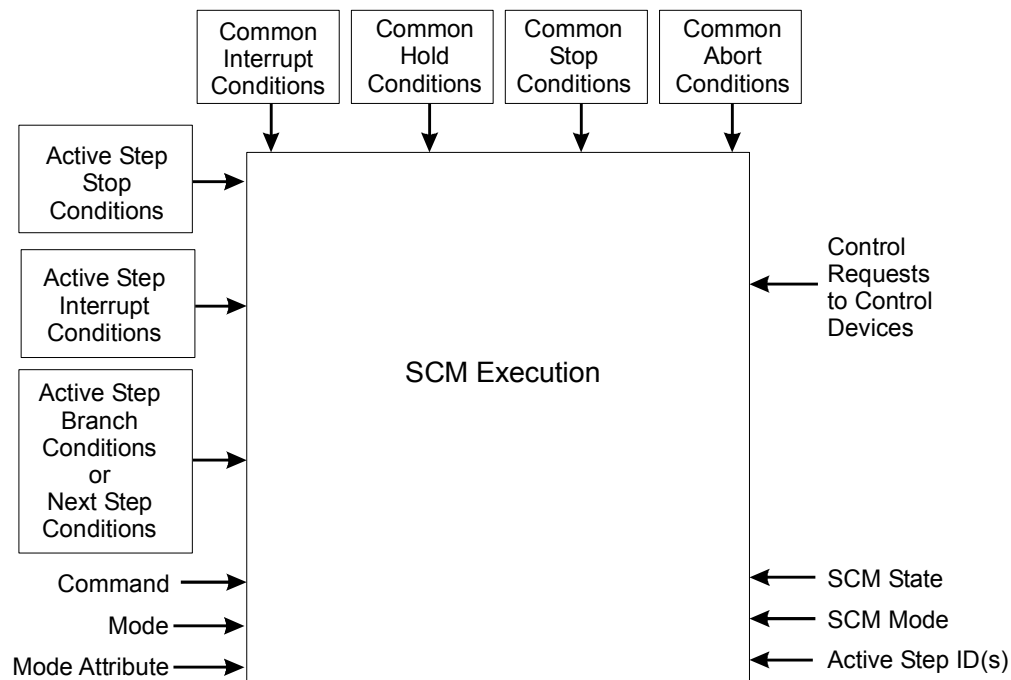


Figure 53 Scope of SCM execution.

Scope and status of SCM execution

The SCM present scope of execution includes the following current attributes.

- Evaluated exception invoke TRANSITIONS. (Those exception HANDLERs that are enabled and are being monitored, but they are not active.)
- Evaluated (those enabled) TRANSITIONS in the active HANDLER.
- Active STEP in the active HANDLER.

The following rules are used to apply the scope of execution to the management of SCM state, status and alarm data.

- The SCM's current status is a composite of all lower level status in the present scope of execution. Of course, this status changes with changes in the present scope of execution. This means the status from the previous scope has no influence on the SCM's current status.
- As the SCM advances, the present scope of execution changes and updates the SCM's current status. This means all previous lower level status no longer influences the current execution status (EXECSTS) of the SCM. For example, a TRANSITION with bypassed conditions in the previous scope of execution caused the SCM EXECSTS to change to Bypass; but, in the present scope, the bypassed conditions no longer influence the SCM EXECSTS and it returns to OK, assuming all operations are OK.



ATTENTION

The EXECSTS of a TRANSITION remains Bypass for as long as its input condition is bypassed (C.EXECSTS = Bypass). This doesn't affect the SCM's EXECSTS if the TRANSITION condition is not in the present scope of execution.

- The current SCM EXECSTS is a function of the present scope of execution. When a Main Handler completes, the SCM goes to the Complete STATE and its EXECSTS changes to OK. The same is true for an SCM in its Idle STATE. When an SCM is started again, its STATE goes from Idle to Running and the present scope of execution determines the current SCM EXECSTS. If the composite of all lower level status is OK, the current SCM EXECSTS will be OK. Otherwise, the current SCM EXECSTS reflects the given not OK status of a TRANSITION or STEP in the present scope of execution. When the SCM EXECSTS becomes Fail, an SCM Fail alarm is generated, if the Fail alarm priority is other than None.

Sequential Control

SCM Interface and CM Interaction

- The TRANSITION and STEP status is made up of its lower level attributes. The following Figure illustrates the present scope of execution concept.

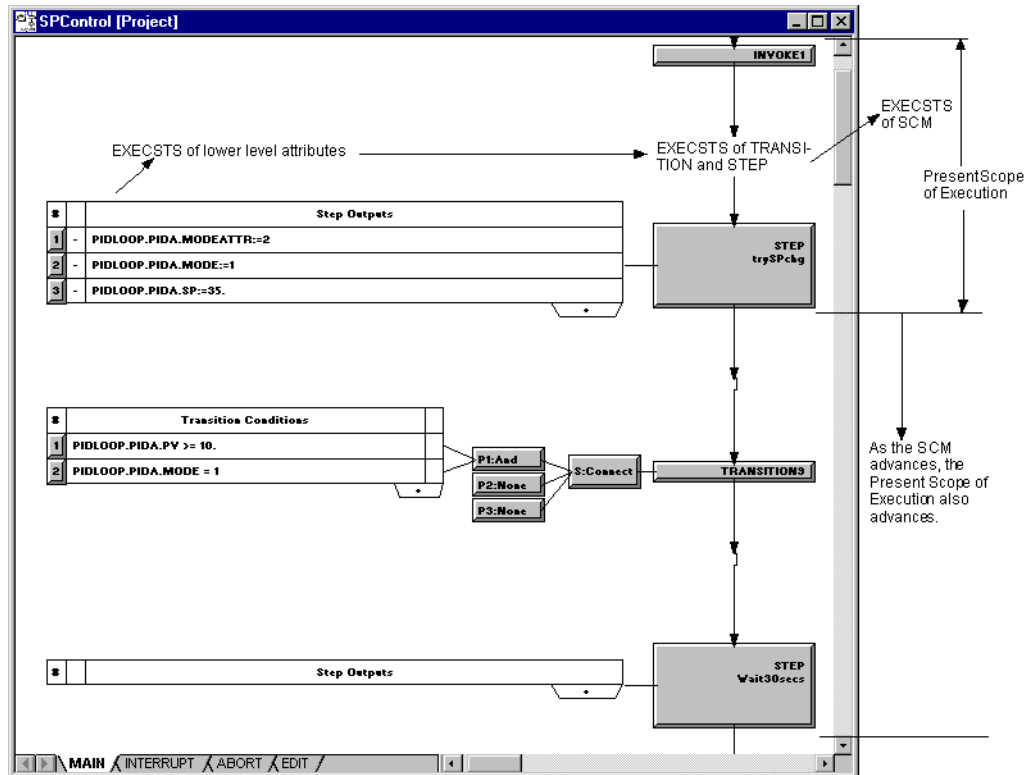


Figure 54 Graphic representation of present scope of execution concept.

Key SCM parameters

The parameter values for State, Execution Status, Command, Mode and Mode Attribute are listed below for quick reference.

Parameter	Values
STATE	Loading(0) Loaded(1) Inactive(2) Validated(3) Running(4) Complete(5) Checking(6) Idle(7) Interrupting(8) Interrupted(9) Restarting(10) Restarted(11) Holding(12) Held(13) Stopping(14) Stopped(15) Aborting(16) Aborted(17) CommErr(18)
EXECSTS (Execution Status)	Ok(0) Bypass(1) Info(2) Warning(3) Fail(4) Error(5) OkPaused(6) BypassPaused(7) InfoPaused(8) WarnPaused(9) FailPaused(10) ErrorPaused(11)

Sequential Control
SCM Interface and CM Interaction

Parameter	Values
COMMAND	None(0) Inactive(1) Start(2) Reset(3) Interrupt(4) Restart(5) Hold(6) Stop(7) Abort(8) Resume(9) Active(10)
MODE	None(0) Auto(1) SemiAuto(2) SingleStep(3) Manual(4) Normal(5)
MODEATTR (Mode Attribute)	None(0) Operator(1) Program(2) Normal(3)

SCM Modes

The Mode parameter defines how automatically the SCM is running.

The following table summarizes the available Modes along with a description.

Mode	Description
<i>Automatic</i>	- This is the normal mode of operation. - SCM runs through the program without operator intervention.
<i>Semi-Automatic</i>	- Used primarily during debugging, for stepping through the SCM under operator control, but without bypassing the TRANSITIONS. - The SCM Execution Status is changed to Pause after the TRANSITION preceding the next STEP is satisfied (but before it becomes the active STEP). SCM State remains unchanged. - All internal timers continue to Run while at Pause during Semi-Auto Mode. - The operator must give a Resume command to cause one more STEP to be executed.
<i>SingleStep</i>	- Used primarily for maintenance and working around failure situations. The operator can step through the SCM without being constrained by the TRANSITIONS and also can jump around and bypass STEPs. - The SCM Execution State is changed to Pause after executing the STEP outputs for the active STEP. SCM- State remains unchanged. - The TRANSITION preceding the next STEP is monitored for operator display purpose only, but is not required to make that STEP the active STEP. - All internal timers continue to Run while at Pause during SingleStep Mode. - The parameter TARGETSTEP can be used to select the next STEP to be executed - The operator must give a Resume command to cause the next STEP to become the active STEP.

Mode	Description
<i>Manual</i>	• Used to stop the SCM (but not the process) to allow the operator to take over control of the devices (after putting them in Operator MODEATTR). The Hold command can be used to stop the process. • The SCM Execution Status and SCM State remain unchanged. SCM program stays at the last preemption point but monitoring of abnormal exception conditions continues. • All internal timers freeze during Manual mode. • All Not-stored requests from the active STEP are reset (this is actually done by the devices themselves). • To restart, the operator must first change the Mode to Automatic (or Semi-Automatic or Single-Step) and then give a Resume command. SCM program starts processing from the last preemption point

A word about SCM mode attribute

The MODEATTR variable defines who can do what to the SCM. This is needed to establish an interface with a third-party batch application.

The characteristics of the MODEATTR and NORMMODEATTR parameters for SCM are almost identical to the same parameters supported by the regulatory control and device control FBs in the CM. Of course, the difference is that the parameters for the SCM define its relationship to a batch application and the parameters for control FBs define their relationship with the SCM.

The planned MODEATTR settings are described in the following table and Figure 55 provides an overview of MODEATTR relationships including a batch application for future reference.

Mode Attribute	Description
None	Mode is neither manual nor automatic – it is more-or-less a “do not care” setting.
Operator	- COMMAND and MODE can be accessed only by the operator (subject to the configured SCM access locks). - The loaded recipe data can be changed by the operator and recipe variables actual values and target values can still be written by the SCM
Program	COMMAND, MODE, and recipe variables can be changed only by the unit level and the operator is locked out..
Normal	The mode attribute assumes the Normal Mode Attribute (NORMMODEATTR) setting, when the Normal command is issued through a display or keystroke in the Station application.

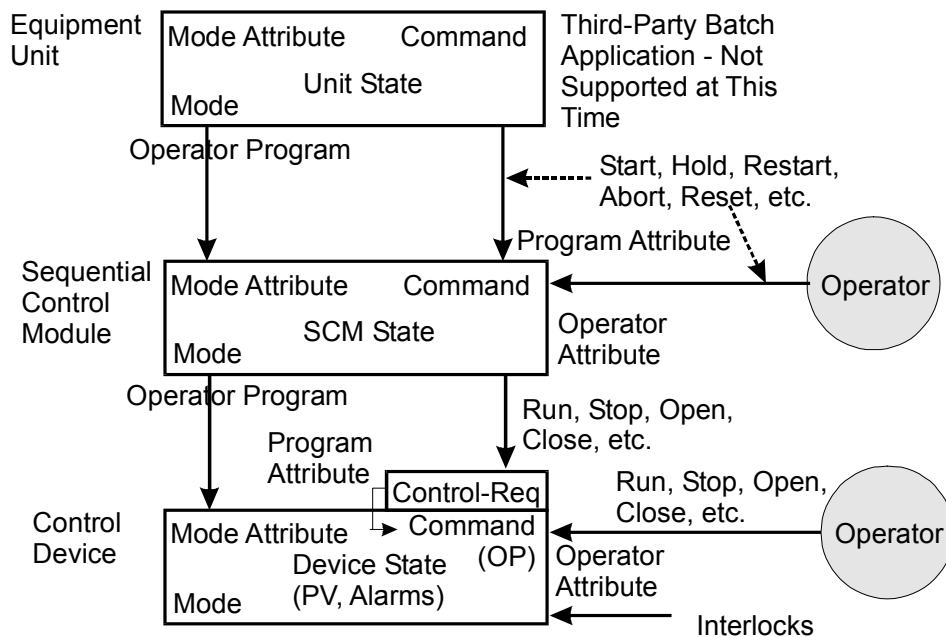


Figure 55 Overview of SCM and control device mode attributes.

SCM STATE and COMMAND interaction

The overall interaction between SCM STATE and COMMAND parameters is illustrated in the Figure 56. The COMMANDS drive the SCM from one STATE to another. You can issue COMMANDS through the SCM run-time monitor and SCM detail displays.

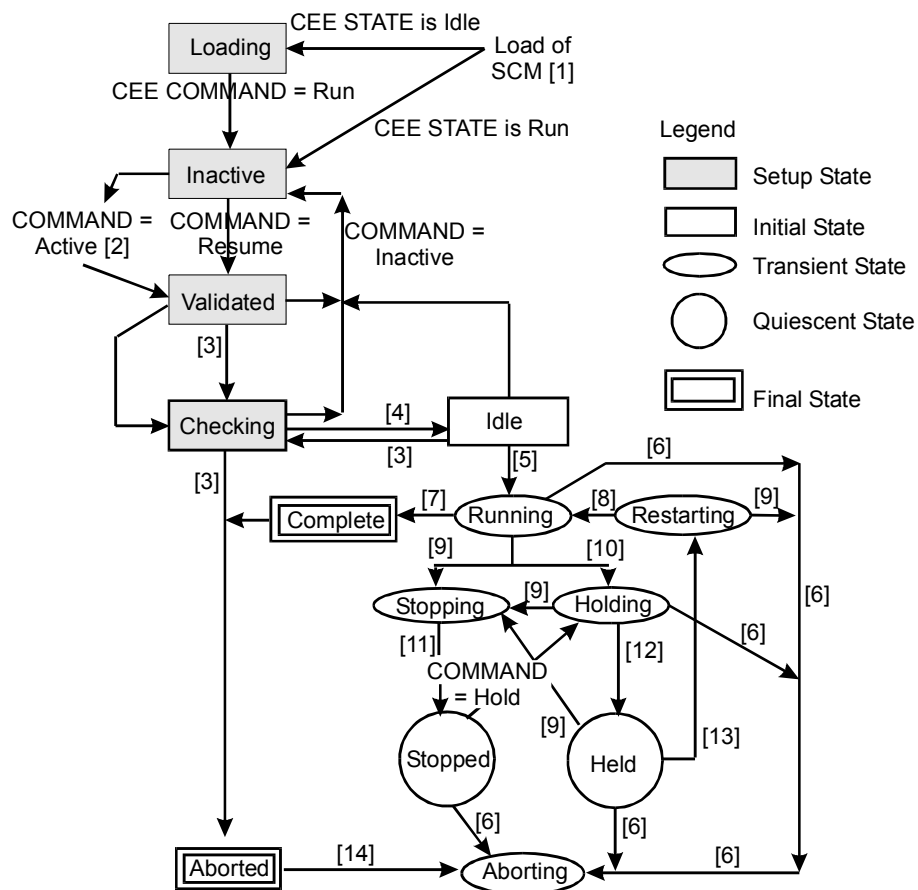


Figure 56 Overview diagram of SCM State and Command interaction.

The explanations for the notes [] in Figure 56 are as follows.

Note	Explanation
[1]	SCM STATE after load depends on the CEE STATE.
[2]	Active command is a macro command that internally issues Resume and Reset commands.
[3]	Transition to Checking is by Command or Invoke Condition.
[4]	Transition to Idle is on completion of Check Handler.
[5]	Transition to Running is by Command or Invoke Condition.
[6]	Transition to Aborting is by Command or Invoke Condition.
[7]	Transition to Complete is on completion of Main Handler.
[8]	Transition to Running is on completion of Restart Handler.
[9]	Transition to Stopping is by Command or Invoke Condition.
[10]	Transition to Holding is by Command or Invoke Condition.
[11]	Transition to Stopped is on completion of Stop Handler.
[12]	Transition to Held is on completion of Hold Handler.
[13]	Transition to Restarting is by command or Invoke Condition
[14]	Transition to Aborted is on completion of Abort Handler.

Notes • Where a "COMMAND =" is shown, the transition must be commanded.

Exception handling

The SCM constantly monitors the process conditions for any "unusual" situation that may require special action. An unusual situation is called an "exception". When the SCM detects an exception, it suspends the normal processing and performs special Exception Handling. Upon completion of the Exception Handling, the SCM may resume normal processing. Exception handling can also be initiated on a command from the operator or the unit level of a batch application.

The Exceptions may be normal (routine or not undesirable) or abnormal (undesirable).
The following Exceptions are supported:

- Abort – abnormal Exception
- Stop – abnormal Exception
- Hold – abnormal Exception
- Restart – normal Exception
- Interrupt – normal Exception
- Check – normal Exception

The Exceptions are handled in one the following ways:

- By executing an Exception Handler, or
- in case of abnormal Exceptions, by changing the SCM STATE and relying on the device drivers to take appropriate actions, or
- by some combination of the previous two ways.



ATTENTION

If any of the device level abnormal handling options (STARTOPT, HOLDOPT, STOPOPT) are enabled and the CM copy of SCMSTATE is in a state which requires an abnormal handler to be active, an SCM Control Request to a control device contained in the CM is rejected as follows,

1. If SCMSTATE = Checking, Idle, or Complete, and STARTOPT is enabled.
2. If SCMSTATE = Stopping or Stopped, Aborting or Aborted, and STOPOPT is enabled.
3. If SCMSTATE = Holding or Held and HOLDOPT is enabled.

This means the device driver has precedence over the SCM Exception Handler, should both commands be present at the same time. This situation is possible, since the abnormal handling option could be enabled on the control device at the same time an SCM Handler is commanding the device. We recommend that you avoid this situation by disabling the device option for the devices the Exception Handler is commanding, when the SCM needs to execute an Exception Handler.

You create Exception Handlers the same as you do a Main Handler using the SCM STEP and TRANSITION blocks. Exception Handlers are entered when initial invoke conditions associated with the SCM exception becomes true. They can be dynamically selected, and one of a set of Handlers can be associated with the same exception and run time. The choice of Handler are managed through a named exception selection list.

Exception Handlers are also invoked when a state change command is received. In this case, the entry conditions to the Handler are skipped and the Handler starts execution at the first STEP.

The following table gives an overview of the operation associated with each type of Exception Handler.

Exception Handler	Operation
<i>Abort</i>	• The process operation needs to be aborted. This is either automatically determined by the SCM (based on the process conditions) or commanded by the operator or unit level. • The SCM transitions to the Aborting state when there is a selected Abort Handler. Some clean up of the process may be required. This may be accomplished in several ways as follows: – The SCM simply goes into its Aborted state. Device drivers force the control devices into appropriate states through the STOPOPT configuration. – If some sequencing operations or more complex processing is required, an Abort Exception Handler (part of the SCM) can be executed. The SCM State will be Aborting, while the Handler is executing. – If the unit level is present, it may take additional actions to complete the clean up. • After aborting, the SCM is in the Aborted state waiting for the next command.

Exception Handler	Operation
Stop	• The process needs to be driven into a fail-safe state. This is automatically determined by the SCM (based on the process conditions) or commanded by the operator or unit level. – The SCM transitions to its Stopped state. Device drivers force the control devices into fail-safe states through the STOPOPT configuration. – If some sequencing operations or more complex processing is required, a STOP Exception Handler (part of the SCM) can be executed. The SCM State will be Stopping, while the Handler is executing. – If the unit level is present, it may take additional actions to complete the fail handling. • When the stop situation is recovered, the SCM goes into its Stopped state. However, you can make the last action in a Stop Handler a Hold Command to establish an automatic transition from Stopped to Holding/Held. If you do not do this, you must issue a Hold command to get the SCM to go to the Holding/Held state. • You can configure a Stop Handler to transition to an Abort or Check Handler.

Exception Handler	Operation
Hold	• The process needs to be put in Hold. This is either automatically determined by the SCM (based on the process conditions) or commanded by the operator or unit level. – The SCM transitions to the hold state. Device drivers force the control devices into their configured Hold states through the HOLDOPT configuration. – If some sequencing operations or more complex processing is required, a Hold Exception Handler (part of the SCM) can be executed. The SCM State will be Holding, while the Handler is executing. – If the unit level is present, it may take additional actions to complete the hold handling. • When the hold situation is recovered, the SCM waits for a restart command to return to the normal (or main interrupt) processing STEP(s) which were suspended. If some special processing or sequencing operations are required to put the process in an appropriate state before continuing, a Restart Handler is executed. • If you use a Restart Handler, we recommend that the last block in the Hold Handler be a TRANSITION one that is configured to trigger a transition only if the Hold invoke conditions are false. Otherwise, it is possible for a Restart Handler to execute its first STEP while the Hold conditions are still True. In this case, the SCM returns to the Hold Handler, but the first STEP of the Restart Handler has already been executed.
Restart	• Special processing or sequencing operations are required to put the process in an appropriate state before restarting. – A Restart Exception Handler is executed on a transition from Held to Running state, if configured and enabled. – There is no monitored invoke condition for a Restart Handler. You must issue a Restart command.

Exception Handler	Operation
<i>Interrupt</i>	• Special processing (different from the SCM's current processing path) is called for based on the process conditions. – An Interrupt Exception Handler (part of the SCM) is executed. – This situation is considered part of normal processing and the unit level above or the device level below do not need to take any special actions. • When the Interrupt Exception Handler is finished, the Main Handler is automatically resumed.
<i>Check</i>	• Verification of base conditions is needed for the SCM to execute properly. This is either automatically determined by the SCM (based on the transition from Checking to Idle state) or commanded by the operator or unit level. – A Check Exception Handler is executed as a normal handler for transitioning from the Checking to Idle state, if configured and enabled. No other Handlers are enabled.

A word about SCM/CM communications error

There are several error conditions that can occur when the Control Module is attempting to read the monitored data from the SCM. These conditions could reflect errors made during the configuration of the Control Module, a contained control device FB, or the SCM.

If a communication error occurs with the SCM, the local CM parameters assume the following default values.

- SCMMODE changes to Manual
- SCMSTATE changes to CommErr
- SCMASTEP is set to 0

SCM access locks

You can configure Single Step, Abort, and Control locks for one of these access levels through the SCM block configuration form.

- Operator
- Engineer
- View-Only
- OtherFB

A brief description of the specific function provided by each lock is given below.

Lock Type	Description
<i>Single Step</i>	SingleStep-Mode-Access-Lock defines who can do the following • change the Mode to SingleStep • store to the Target Step.
<i>Abort</i>	Abort-Command-Access-Lock defines who can give the Abort command.
<i>Control</i>	Control-Access-Lock defines who can do the following: • give a Hold command • change the Mode to Semi-Automatic or Manual

SCM run-time monitoring color code reference

The run-time monitoring display for a loaded SCM uses the following color-coding scheme to identify various states.

- SCM TRANSITION block color-coding reference.

If TRANSITION block's color is . . .	Then, its state is . . .	And, related block parameter setting is . . .
Gray	Not Executed Yet	PROCESSED = Off
Green	In Execution	STATE = Enabled
Blue	Already Executed	PROCESSED = On

- SCM TRANSITION block Condition Expression color coding reference.

If TRANSITION Condition's color is . . .	Then, its state is . . .	And, related block parameter setting is . . .
Green	ON	C[X].FL = On
Red	Fail or Error	C[X].EXECSTS = Fail or Error
Yellow	Warning	C[X].EXECSTS = Warning

- SCM STEP block color-coding reference.

If STEP block's color is . . .	Then, its state is . . .	And, related block parameter setting is . . .
Gray	Not Executed Yet	STATE = Inactive, and PROCESSED = Off
Green	In Execution	STATE = Active, or ProcOutputs, ProcMessages, or ProcMinWait
Blue	Already Executed	STATE = Active or ActiveComplt, and PROCESSED = On

- SCM STEP block Output Expression color coding reference.

If STEP Output's color is. . .	Then, its state is. . .	And, related block parameter setting is . . .
Red	Fail or Error	OP[X].EXECSTS = Fail or Error
Yellow	Warning	OP[X].EXECSTS = Warning
Green	OK	OP[X].EXECSTS =Ok, and STATE = In Execution

Sequential Control
SCM Interface and CM Interaction

Common SCMs

Functional Overview

The SCM Batch Enhancements provide the user with Common SCM functionality. A Common SCM is one that can control several equipment units, but only one at a time. If the Batch Enhancements option is not used, the SCM will behave as a normal SCM.

The Common SCM must be assigned to the equipment before controlling it. This assignment can be done at either configuration or run time. The choices of equipment that a Common SCM can be assigned to are configured at configuration time using the *Alias Table* on the Aliases Tab of the SCM Configuration Form.

A user may define aliases and the associated references in the Alias Table. These references represent parameters of the equipment that the SCM could be assigned to at run time. The references are called "instance parameters". Once defined in the Alias Table, the aliases could be referenced by an SCM expression. At run-time the user can select a column of references, or an instance, from the Alias Table, to tie the aliases to the parameters of specific piece of equipment. Once the alias reference is selected, the SCM is assigned to the equipment referred to by the selected instance. The SCM can then command this equipment.

All of the Read and Write requests generated by the SCM Step output / Transition condition expression, which reference the aliases in the SCM, will be re-directed to the currently selected equipment. Once the SCM is done with the selected equipment, the user can select another instance in the Alias Table, causing the SCM to control another piece of equipment. All of the commands that the SCM sends to the alias will now be sent to the newly selected equipment. This is called Dynamic Indirection.

For more information on Common SCMs, refer to the Control Builder Components Theory sections:

- [Alias Table](#)
- [Dynamic Indirection](#)
- [Relationship between Common SCM and Batch Level 1 Function](#)
- [Performance Requirements](#)
- [Scenarios and Examples](#)

Alias Table

An Alias Table is a matrix that associates alias names with the actual parameters that the aliases may resolve to at run-time. The Alias Table is the key component to the Common SCM function. It creates the foundation for dynamic indirection.

Structure of the Alias Table

An Alias Table consists of a number of rows or entries. Each row, or entry, contains the following information: alias name, block model of the alias, parameter model of the alias, alias reference, and binding status of the current instance. The user builds all the information at configuration time, except binding status, which is obtained at run-time to reflect the current binding status of the selected instance parameters.

Alias name (Alias Name), block model (Model Block), and parameter model (Model Param) are the properties of the aliases. They define the identification and the data type of the aliases. Alias Name, Model Block, and Model Param are arrays, sized by the number of aliases, and indexed by the row number of the Alias Table.

Alias reference (Instance 1, Instance 2, Instance 3, ...) defines the references of the aliases. These references represent the parameters the aliases may resolve to at run-time. Alias reference is a two dimensional array, sized by the number of aliases and number of references for each alias. Each row holds all the references for the alias defined in that row, and each column holds one reference for each of the aliases defined in the Alias Table. Each reference is called an instance parameter, and each column of references is called an instance. The instance numbers are automatically assigned, starting at one. On the Alias Table configuration form, Instance 1 represents column 1, and Instance 2 represents column 2 etc.

As mentioned above, binding status (Status) is not a configurable parameter. It is a run-time parameter, which represents the current binding status of the selected instance. Status is an array, sized by number of aliases, and indexed by the row number of the Alias Table.

Alias Table parameters

The following parameters were added to the SCM to support the Alias Table:

Parameter Name	Description
ALIASOPT	Defines whether or not an Alias Table will be configured in the SCM.
NUMALIASES	Defines number of aliases in the Alias Table.
NUMINSTANCES	Defines number of instances in the Alias Table.
INSTSELECT	Represents selected instance column.
ALIASBLKTYP[]	Defines the block model of the alias.
ALIASPRMTYP[]	Defines the parameter model of the alias.

Alias Table Configuration

Number of Aliases
(NUMALIASES)

This field determines how many aliases, or rows, can be configured in the Alias Table. An edit box is provided in the Alias Table form for configuring this field.

The user can configure up to 500 aliases in an Alias Table. Once a non-zero number is entered into the edit box, the Alias Table grid with the configured number of rows will be exposed on the page.

NUMALIASES can be changed any time under project tab. When NUMALIASES changes after the Alias Table grid is exposed, the number of rows exposed will be updated accordingly. If NUMALIASES is decreased, the appropriate number of rows will be removed from the bottom of the Alias Table. If NUMALIASES is increased, the appropriate number of rows will be added to the bottom of the Alias Table.

NUMALIASES combined with NUMINSTANCES (number of instances) determines the size of the Alias Table.

Common SCMs

Alias Table

Number of Instances (NUMINSTANCES)

This field determines how many instances can be configured in the Alias Table, or how many instance parameters (references) can be configured for each alias. An edit box is provided in the Alias Table form for configuring this field.

The user can configure up to 100 instances in an Alias Table. Once a non-zero number is entered into the edit box, the configured number of columns will be exposed on the Alias Table grid.

NUMINSTANCES can be changed any time under project tab. When NUMINSTANCES changes after the Alias Table grid exposed, the number of instance columns exposed will be updated accordingly. If NUMINSTANCES is decreased, the appropriate number of columns will be removed from the right end of the Alias Table. If NUMINSTANCES is increased, the appropriate number of columns will be added to the right end of the Alias Table.

NUMINSTANCES combined with NUMALIASES determines the size of the Alias Table.

Restriction on the Size of the Alias Table

Due to the memory resource limitation and the performance consideration, the size of an Alias Table is limited to maximum 4,500 instance parameters. The user has the flexibility to configure more aliases and fewer instances, or more instances and fewer aliases, but $\text{NUMALIASES} \times \text{NUMINSTANCES}$ can not exceed 4,500. For example, if the user configure 500 aliases, he/she can configure up to 9 instances. If the user chooses to configure 100 instances, he/she can configure up to 45 aliases.

Alias Name

Alias Name is a user defined name that identifies the entry for each row in the Alias Table. It is a text string that allows the users to choose meaningful names to represent the parameters of their process. The alias name has to be unique within the Alias Table and among parameter names of the container the Alias Table is associated with.

The Alias Name is configured through an edit box. The valid forms for the Alias Name could be xxxx or xxx.yyy, i.e. My_Alias, StepOutput1.Alias. The maximum length of an Alias Name is 32 characters.

Alias Name can be changed only at configuration time, under Project tab.

Model Block (ALIASBLKTYP)	Model Block defines the block model of the alias. This field combined with Model Parameter derives data type of the alias, which is used to validate the data type of the instance parameters of the alias, when the instance parameters are configured, or at run-time when an instance parameter is changed. The options for this field are all system templates. A combo box is provided for configuring this parameter. The Model Block combo box contains all the container and component block templates in the system. Model Block can be modified only at configuration time, under Project tab. The Model Parameter of the alias can not be configured if the Model Block is not configured, and none of the instance parameters for the alias can be configured if any one of the Model Block and Model Parameter of the alias is not configured.
Model Parameter (ALIASPRMTYP)	Model Parameter defines the parameter model of the alias. This field combined with Model Block derives data type of the alias, which is used to validate the data type of the instance parameters of the alias, when the instance parameters are configured, or at run-time when an instance parameter is changed. The options for this field are all the parameters available for the selected model block, except the aliases. A combo box is provided for configuring this parameter. The Model Parameter combo box contains all the parameters available in the selected model block. Model Parameter can be modified only at configuration time, under Project tab. None of the instance parameters for the alias can be configured if any one of the Model Block and the Model Parameter of the alias is not configured.
Instance Parameter Validation	The instance parameters are validated at the configuration time. The alias data type derived from the alias model block and model parameter is used to ensure the consistency between the alias and the instance parameters. The instance parameters do not have to be of identical block type and parameter of their alias. However data type between the instance parameter and the alias should match. Otherwise, the Control Builder will not accept the value.
Null Instance Parameters	The user is allowed to leave an instance cell blank, in the case that a piece of equipment is not used for a particular instance column. A blank instance cell is called a "null" instance parameter. A store to a null instance parameter may cause the SCM step output failure, therefore causing the SCM EXECSTATE to fail. The user must program around these cases to avoid making connections to null instances.

Common SCMs

Alias Table

Instance Selection (INSTSELEC)

This field specifies which column of reference connections the aliases will resolve to at run time. The options for this field are the column numbers of the configured instances, i.e. 1 represents Instance 1 and 2 represents Instance 2. This parameter is configured through a combo box, which provides the valid instance numbers, 1 to NUMINSTANCES.

At run-time, each individual alias in the Alias Table is resolved to the corresponding instance parameter in the selected column.

The Instance Selection can be changed at run-time.

Deleting/Adding/ Inserting Rows

An alias can be deleted from the Alias Table under Project tab, if there is no connection made to that alias. When an alias is deleted, the whole row will be deleted, and the rows below it will be pushed up. Once a row is deleted, parameter NUMALIASES will be updated to reflect the changes on the number of rows.

An alias can not be deleted if it is referenced in a connection.

A row can be added to the end of the Alias Table at configuration time. When a row is added to the Alias Table, parameter NUMALIASES (number of aliases) will be updated to reflect the change on the number of rows.

A row can be deleted or added by changing NUMALIASES.

A row can be inserted anywhere in the Alias Table at configuration time. When a row is inserted in the Alias Table, all the rows below it will be pushed down, and parameter NUMALIASES (number of aliases) will be updated to reflect the change on the number of rows.

Deleting/Adding/
Inserting Columns

A column can be deleted from the Alias Table at configuration time. When a column is deleted, all the columns behind it will shift to the left. For example, if instance n is deleted, instance n+1 will become instance n, instance n+2 will become n+1, so on and so forth. Parameter NUMINSTANCES (number of instances) will be updated to reflect the change on the number of columns. If the selected instance is deleted, the instance next to it will replace it. The user should receive a warning about the impact of deleting a column.

An instance can be added at the very right of the Alias Table at configuration time. When an instance is added to the Alias Table, parameter NUMINSTANCES (number of instances) will be updated to reflect the change on the number of instances.

A column can be deleted or added by changing NUMINSTANCES.

A column can be inserted to the Alias Table at configuration time. When a column is inserted, all the columns behind it will shift to the right. Parameter NUMINSTANCES (number of instances) will be updated to reflect the change on the number of columns.

SCM Expression

To use the dynamic indirection feature, an Alias Table must be configured for the SCM, and the SCM expression has to refer to the aliases defined in its Alias Table. The aliases are referenced as Tagname.alias name. The Tagname here refers to the name of the SCM who owns the Alias Table. The following examples show how an alias is used in an SCM expression.

Assume that SCMA owns the Alias Table. SCMA Step and Transition expression have references to its aliases.

SCMA Step Out expression - open the valve:

SCMA.FV_A_OP := 5

SCMA Transition Condition expression - checks if the valve is open:

SCMA.FV_A_PV = 5

FV_A_OP and FA_A_PV will be resolved to one of their instance parameters at run-time, based on the instance selection.

Note: References to aliases defined in other SCMs' Alias Table is not supported at this time.

Dynamic Indirection

Dynamic indirection allows a client block (i.e. a step output or transition block of an SCM) to communicate to different data owner /destination blocks at run-time, through a single communication channel. This provides the ability for a user to create a single SCM that may control different equipment each time it runs.

There are two levels of connections between the client block and the data owner/destination block in the communication path. The first level connection is the connection between the client blocks and the aliases. The second level connection is the connection between the aliases and the data owner/destination blocks. The Alias Table sits in between the client block and the data owner/destination block. Since one alias has many instance parameters, one connection can provide multiple communication paths between the client block and the data owner/destination block through the second level connection, but only one can be used at a time. The alias acts as a switch to connect the client block to different data owner/destination blocks.

The following sections describe how the dynamic indirection works.

Binding

The Alias Table owner tries to locate the data owner/reference destination blocks for the aliases at load time or at run-time. This process is called binding. Binding process results in the creation of two sets of connections, the connection between the client block (i.e. an SCM Step Output or Transition block) and the aliases, and the connection between the aliases and the data owner/reference destination blocks.

The connection types to the aliases are also determined at load time by the SCM Step Output and Transition Condition Expression. If an alias is referenced by a Step Output expression as the destination for a store, the connection of this alias will be created as a “Store” connection. If an alias is referenced by a Transition Condition expression, the connection of this alias will be created as a “Get” connection. If an alias is referenced by both, the connection will be created as Get and Store connection. For the aliases who have remote references, if they are not referenced by SCM Step Output or Transition Condition expression, the memory resource for the remote Get access will not be allocated, and that alias can not be accessed by clients, such as the Server Display, at run-time.

Binding process also results in the binding status being set.

Binding Between
the Client Blocks
and the Aliases

Binding between client blocks, i.e. SCM step output or transition blocks, and the aliases is performed at load time when loading the connection configuration to the client blocks. This process will always succeed because alias table configuration and connection references are loaded prior to connection configuration for contained components. This process results in creation of the first level connections.

Binding Between
the Aliases and
the RDB

Load of connection configuration to a client block also results in creation of connections between aliases and owner/destination blocks represented by the instance parameters. During this process, alias table requests the binding **for all of its instance parameters**. This process results in creation of the second level connections.

If the instance parameters of an alias represent parameters of devices on remote controllers, and the alias is referenced as a Get reference, memory will be allocated for all of the instance parameters of the alias for the remote get access as the result of binding. If there is not enough memory resource, a severe error will be returned, and load of the SCM will be aborted. In the case of REF before DEF or DEF deleted, memory for remote get access will still be located, and load will continue. However the situation will be reflected by Binding Status. The Fail-Safe value will be used before CDA can successfully collect data from the DEF block.

Because of the "Full Binding At Load " approach, memory resource will be allocated for all of the remote instance parameters of an alias, if the alias is used as a Get reference, even though only one instance parameter (the selected instance parameter) is used at a time. **The application engineer should be aware of this fact, and manage the peer to peer resource properly when configuring the Alias Table.**

Binding at Run-
Time

Run-time binding is needed if the binding fails at load time because of REF before DEF or DEF block is deleted. The run-time binding is initiated by the client block upon the first access to the alias.

Binding Status	Binding status represents the status of binding for the selected instance column. The binding status is represented by parameter STATUS. STATUS is an enumeration with value Null (0), OK (1) and Binding (2). If the data owner/reference destination block of an alias is successfully located, the STATUS of the alias will be set to OK. If the instance parameter is not configured (null instance parameter), the STATUS of the alias will be set to Null. If the data owner/reference destination block of an alias can not be located, the STATUS of the alias will be set to Binding.

Failure to bind is caused by any of the following situations:

5. The data owner/reference destination block is not loaded,
6. The data owner/reference destination block is deleted.

Instance Selection

Instance selection determines which references are the aliases resolved to at run-time. It ties the aliases to the reference parameters that the selected instance represents. The instance may be selected at configuration time, and changed at run-time. Normally, the aliases are bound to all of their instance parameters at load time. So changing instance selection at run-time should not affect the data fetching from the aliases if the instance parameters of the aliases are on remote nodes. The user should not see the delay in fetching data when the instance selection changes.

The instance selection may be changed in the following ways at run-time:

- Change the instance selection from display;
- Store to INSTSELEC from Total Plant Batch; or
- Store to INSTSELEC from an SCM Step Output.

Store to the Aliases A step output block of an SCM can store to an alias by referring to the SCM name.alias name, i.e. SCMA.FV_A_OP, in the expression. When a store request is made to an alias, the Alias Table owner forwards the request to the reference destination block, based on the currently selected instance parameter for that alias. The reference destination block completes the store and returns the status to the Alias Table owner. The Alias Table owner then passes the status to the step output block. If the reference destination block is on a remote controller, the Alias Table owner will issue a peer store request to CDA. The "Pending" status will be returned since the store can not be completed in one controller cycle. In this case, the step output block will have to check the status with the Alias Table owner every cycle, until the request is completely processed.

At this time, SCM step output blocks are the only clients that can store to the aliases.

Monitoring Alias Values Read requests specified using an alias name return the value of the current instance parameter for that alias. A client (i.e. a transition block of an SCM) can access an alias by referencing the SCM name.alias name, i.e. SCMA.FV_A_OP. Two levels of access are involved in this type of access. The read request for the alias comes to the Alias Table owner, and then gets re-directed to the data owner block by the Alias Table owner, based on which instance parameter is currently selected for the alias. The data owner block processes the request, and return the value and status to the Alias Table owner. The Alias Table owner then returns the value and status to the client.

If the data owner block is on a remote controller, the alias value can be monitored by the Server Display only if the SCM Step Output or Transition Condition expression has read access to the alias. This is the only way that the memory resource for remote Get access can be allocated.

Monitoring Alias Binding Status The user can monitor the binding status of an alias through SCM name.alias name.STATUS, i.e. SCMA.FV_A_OP.STATUS. The request comes to the owner of the Alias Table, i.e. SCMA. The owner of the Alias Table gets the binding status for the selected instance from the alias reference connector, and return the value to the client.

Binding status is owned by the owner of the Alias Table, so there is no second level of access involved.

Monitoring
Expressions

When a CB chart is monitoring an expression, the alias names will be displayed. The enhancement may be done in the future release, so that the user will have the option to either display alias names or the current bound instance parameter.

Fail-Safe Values

The fail-safe values are defined in the following Table, according to the data type of the aliases.

Table 2

Data Type	Fail-Safe Value
Real	NaN
Integer	0
Boolean	FALSE
Enumeration	0

Relationship between Common SCM and Batch Level 1 Function

Due to some design issues, Common SCM and Batch Level 1 function can not be used on the same device. This means that the Common SCM making stores with OPTYPE = NOTSTORED/SET to the device (DevCtl and RegCtl blocks) cannot be used, if the CM is referenced by the SCM's Alias Table. The combination of the two functions may cause unpredictable device behavior at run-time, when the SCM enters an abnormal handler. This configuration of the two-function combination is prevented at the configuration time.

Prevent Two Function Combination When Configuring Instance Parameters

When an instance parameter is configured, the SCM Option of the CM that the instance parameter references is evaluated. If the CM has Batch Level 1 relationship with the SCM that the Alias Table belongs to, the value of the instance parameter will be rejected, and an error message is returned to the user.

Prevent Two Function Combination When Configuring SCM Option of CMs

On the other hand, when configuring the CM's Batch Level 1 relationship with an SCM, the Alias Table Option of the SCM that the CM is connected to is checked. If the Alias Table Option is enabled, all the instance parameters of the Alias Table will be checked. If the CM is referenced by any of the instance parameters in the SCM's Alias Table, the CM will not be allowed to have batch Level 1 relationship with the SCM. An error message will be returned to the user. The user can either configure the CM's Batch Level 1 relationship with another SCM, or remove the references to the CM from the SCM's Alias Table, and re-configure the Batch Level 1 relationship with that SCM.

Performance Requirements

The function must meet the following performance requirements:

- Performance of CMs and SCMs should not be affected if the function is not used.
- None of the alias parameters should be loaded to the controller if the Alias Table is not configured.
- The Alias Table must not affect the performance when opening charts.
- Binding the aliases to the data owner blocks must happen as early as possible to avoid unnecessary waiting time when the peer to peer fetch is involved.
- On-line changes in instance selection should not affect the CPU usage of the SCM.

Scenarios and Examples

Scenarios and examples are provided in the following sections to explain Alias Table configuration and run-time behavior of dynamic indirection.

Scenarios

Ram-Retention-Start-Up	Binding process will happen as part of Ram-Retention-Start-Up. It results in the aliases bound to currently selected instance. Once the CEE goes to RUN, the normal operation on the aliases should be resumed. The currently selected instance will take effective.
SCM Step Output Block Stores to Null Instance Parameters	A store to null instance parameter from an SCM step output expression results in the SCM step output failure. The failure is propagated to the SCM, and results in the SCM EXECSTATE to fail. The user may check the binding status of the alias before storing to it, and do proper branching if the binding status = Null, to avoid SCM failure.
SCM Transition Block Fetches from Null Instance parameters	When an SCM transition condition expression fetches data from a null instance, the Alias Table owner will issue a Null instance warning to the expression, and the piece of condition which references to the null instance parameter will not become TRUE.
Monitor Alias Value from the Server Display	The server display monitors an alias that has remote references. Assume that the alias is not referenced by the SCM Step Output or Transition Condition expression as a Get reference. The Server Display can not monitor the alias value. An error will be returned.

Examples

Alias Table Configuration	An Alias Table can be configured for SCMA as follows:
SCM Expression	SCMA Step and Transition expression have reference to its aliases. Transition condition expression in Transition1A - checks the binding status of the alias: SCMA.FV_A_OP.STATUS = 1 Step output expression in Step1A - open the valve: SCMA.FV_A_OP := 5 Transition condition expression in Transition2A - checks if the valve is open: SCMA.FV_A_PV = 5
Run-Time Access to the Alias Value	Assume that Instance 2 is selected as the current instance. At run-time, when the pCode of transition expression SCMA.FV_A_OP.STATUS = 1 is executed, a get request to the binding status of the alias FV_A_OP is generated. The Alias Table's owner, SCMA, returns status = OK to Transition1A. When the pCode of the step output expression SCMA.FV_A_OP := 5 is executed, a store request to alias FV_A_OP is generated. The Alias Table owner, SCMA, re-directs the request to the current instance parameter of FV_A_OP, which is GOP of DevCtlA in CM2. DevCtlA successfully sets its GOP to 5, and completes the store request. DevCtlA returns status = OK to SCMA, and SCMA passes the status = OK to Step1A. Assume that CM1.DevCtlA.PV = 4, CM2.DevCtlA.PV = 5, and CM3.DevCtlA.PV = 6, and current instance is Instance 2. When the pCode of the transition expression SCMA.FV_A_PV = 5 is executed, a get request to the alias FV_A_PV is generated. SCMA forwards the request to the current instance parameter of FV_A_PV, which is CM2.DevCtlA.PV. Then DevCtlA gets the value of PV, and return value =5 and status = OK to SCMA. SCMA returns value = 5 and status = OK to Transition2A.

Common SCMs
Scenarios and Examples

Part II
Control Builder Notifications
Theory

Notification System

Design Basics and Component Identification

Overall Scheme

The Experion PKS system generates notifications when it detects certain changes in the process or the control system. For reference purposes, Experion PKS notifications fall into one of these general notification classes.

Notification Class	Description
Message	Operator message defined as part of user-configured MESSAGE function block within Control Builder.
Operator Change	A journal entry triggered by an operator change in value for a function block parameter. Changes made through Station displays and Monitoring mode in Control Builder are journalized.
Process Alarm	A condition defined as part of a user-configured control strategy within the Control Builder application.
System Diagnostic	A condition determined by system diagnostic software.
System Information	A condition based on the execution of a system service and it may not be an explicit user configuration.
System State	A condition determined by the change of state in the process control equipment or a major component of the equipment.
Third-Party Alarms	Similar to process alarms, but the points for these alarms are not linked to Control Module function blocks. These alarms are defined as part of user-configured Status, Analog, and/or Accumulator points through the Quick Builder application.

Notification System

Design Basics and Component Identification

Figure 1 shows the overall scheme that the Experion PKS system uses for notification generation. While the Experion Server knows of all the “points” in the system, the alarm detection and annunciation determination occurs in the Control Processor. Note that the term event is used interchangeably with notification.

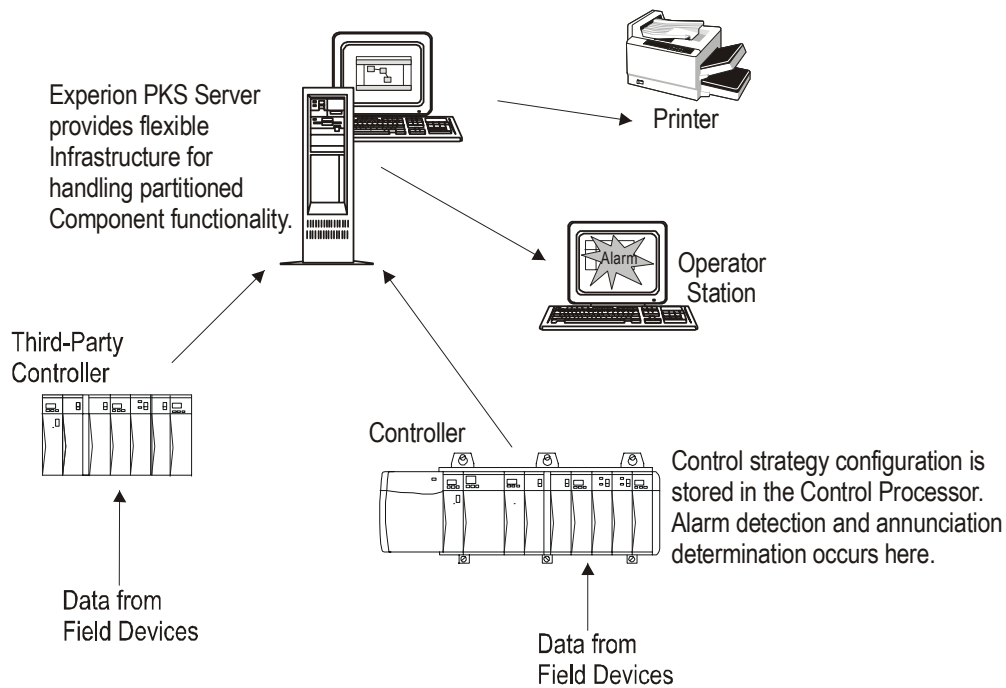


Figure 1 Experion PKS scheme for notification generation.

Communications model

Figure 2 shows how the communications hierarchy for the Experion PKS system compares with the International Standards Organization (ISO) - Open Systems Interconnection (OSI) model. The Control Data Access (CDA) is the communications application layer for the Experion PKS system. CDA provides these two major communication services:

- Named Access, and
- Notifications.

Experion PKS System		ISO-OSI
Control Processor	Server	Reference Model
CEE	Dynamic Cache: Run-time Monitor, Builder Load	
CDA (Pub/Sub,Req/Resp, Notif. Pub)	CDA (Sub,Req, Notif. Sub)	Server DA DDE
Data Types	Edian Conversion	
Null	Null	
CN Transport Class 5 Frag/Reassembly, Mult Msg	CN Class 5	TCP
CN Network	CN Network	IP
ICP, SMAC	KTC-SMAC	802.3
ICP, ControlNet	ControlNet	Ethernet Media, Serial Comm.
		User Layer
		Application Layer)
		Presentation Layer
		Session
		Transport
		Network
		Link
		Physical

Figure 2 Experion PKS communications hierarchy comparison.

Distribution model

Figure 3 is a graphic representation of the flow of notification data in Experion PKS system. The emphasis is on CDA communication services and its relationship with the notification classes previously listed in this document. Particular emphasis is given to the process alarm class, which is defined as part of the Control Module configuration through the Control Builder application. The pertinent control data is identified as Experion PKS points. These points are stored in the server's Real Time Database (RTDB) whenever the configured Control Module is loaded into the Control Processor (in the Hybrid Controller). Server subsystems can handle the processing of notification data for operator displays, printouts, reports, and logs.

Figure 3 includes a third-party controller to show that alarm and events generated by another controller can be integrated with Experion PKS data. You configure third-party controller data as Analog, Status, and/or Accumulator points through the Quick Builder application. These points are also stored in the server's RTDB and processed by server subsystems. This means that third-party controller notification data is seamlessly integrated with Experion PKS notifications.

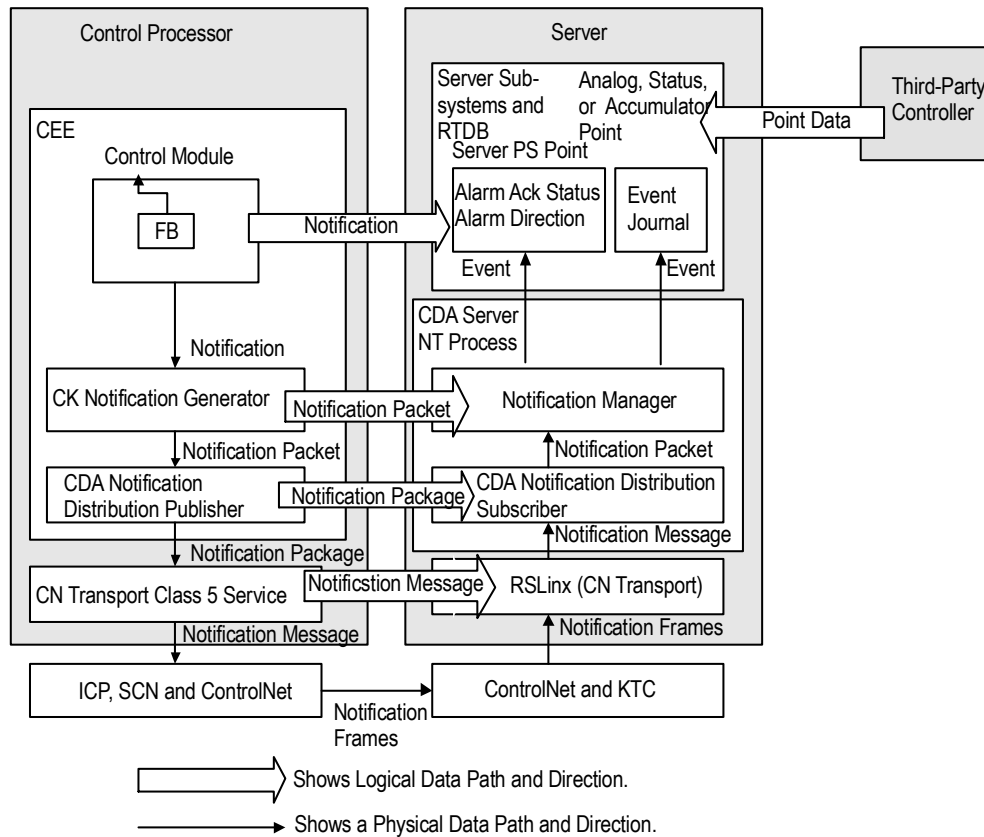


Figure 3 Notifications data flow diagram.

Control functions summary

We refer to the following elements, relative to the flow diagram in Figure 3, as control functions. These functions deal primarily with transforming detected notifications into notification packets, and passing those packets to the notification distribution system.

- Control Execution Environment (CEE)
- Control Module (CM) and Sequential Control Module (SCM) - not shown
- Function Block (FB)
- Control Kernel (CK)

Three additional elements, not included in the list above, that get involved with generating notifications are:

- Notification Manager (NM),
- Network Diagnostic Manager (NDM), and
- Station Display subsystem.

The NM detects and generates system diagnostic events and state changes. The NDM complements the NM by periodically checking the status of all the devices that are physically part of the system, but they are not explicitly configured as part of the user's control strategy. The NDM detects and generates system alarms or events depending on the severity of the conditions reported by the physical devices. The following table lists the devices checked by the NDM for reference. (Note that NDM is active on both the Primary and Secondary Server in systems with redundant Servers; however on the Secondary Server, NDM only monitors the PCIC device.)

Display Abbreviation	Description
CL	Control Logic 5550 Controller
CNETDRVR	RSLinx ControlNet Driver
CNI	ControlNet Module
ENET	Ethernet Module
ENETDRVR	RSLinx Ethernet Driver
GW	ControlNet Gateway for either Series A or Series H Rail I/O.
KTC	CNI ISA PC to ControlNet Interface Card

Display Abbreviation	Description
LD	Fieldbus Linking Device
PCIC	PCI PC to ControlNet Interface Card
PLC	Family of Programmable Logic Controllers
RM	Redundancy Module

The Station Display generates operator change events for those changes made to CM parameters from a Station display. The Station application can be included on the Experion PKS Server as well as a remote Operator Station.

The roles that these control functions play in the notification operation are summarized below.

Control Function	Notification Role
CEE	Serves as notification generator for all notifications that are detected by blocks within the Control Processor, these would include the Control Processor, CEE, IOM, CM SCM, and function blocks (FB). However, if the Control Processor is in a not OK state (which does not support CEE execution), then the Control Processor does not support notification generation.
Control Kernel	Executes blocks in an automatic and synchronous fashion, according to specified period and execution order. Supports the transport of variable values between blocks, by letting the client blocks do a call-get or a call-store of supplier blocks. Manages mapping of external block identifiers to internal database. Serves as a main interface between CDA and function block database. Manages notification distribution from each CM and SCM to the CDA notification publisher. Processes event recovery requests by commanding the same from each assigned CM.
Control Module and Sequential Control Module	Provide tag name for alarms sent by the function blocks. Provide alarm enable/disable function, so users can initiate or suppress notification traffic. Provide point active/inactive function, so users can access given function block parameters.

Notification System

Design Basics and Component Identification

Control Function	Notification Role
Function Block (FB)	Detects notification condition for configured alarms and events. Supports disable and enable alarms, and recreate event methods. It passes disabled alarms to the CM. Invokes the recreate event method, when the alarm enable is set (disabled cleared) to restore current alarm and event information. Note that CM invokes recreate event for notification regeneration, due to recovery.

CDA functions summary

We refer to the following elements, relative to the flow diagram in Figure 3, as CDA functions. These functions deal primarily with the communication services that send notification packets from a publisher (server) to a subscriber (client). This includes both the distribution of notifications on the ControlNet from the Control Processor to the Experion PKS Server as well as distribution from the server to event clients.

- CDA Notification Distribution Publisher
- Notification Manager
- Event Journal
- Alarm Journal
- CDA Notification Distribution Subscriber
- Network Diagnostic Manager
- Notification Client

The roles that these CDA functions play in the notification operation are summarized below.

CDA Function	Notification Role
Alarm Journal	Serves as Server log for all alarms passed to the Station Alarm Summary display and other displays as well as reports.
CDA Notification Distribution Publisher	Takes notifications from one (or more) notification generator and distributes them on ControlNet to a CDA Notification Distribution Subscriber. If the CK-CDA event queue is full, CK stops putting events into the queue, and it puts backpressure on its event producers. There is no "queue full" event sent by CDA to the server. CK is not allowed to overwrite existing events in the queue.

CDA Function	Notification Role
CDA Notification Distribution Subscriber	Initiates and maintains the connection to all ControlNet based Notification Distribution Publishers and passes the notifications it receives on this channel to the Notification Manager.
Event Journal	Serves as Server log for all events passed to the Station Event Summary display and/or reports.
Network Diagnostic Manager	Reports its events and alarms to the Notification Manager (NM) for final reporting to the registered client. It periodically polls network status.
Notification Client	Receives events from the Notification Manager.
Notification Manager	Receives all notifications from all notification generators. It formats all events for the server (Unicode to MCBS, enumeration conversion, CDA handle conversion) and passes the events to the registered client.

About Transport services

The CDA Notification Distribution makes use of the CNET Transport System for:

- large (greater than 500 byte) published messages through fragmentation/reassembly,
- transport layer acknowledgment per frame,
- CNET Network priority "High" (highest priority unscheduled service on ControlNet), and
- queued interface between interchange and CDA Notification Distribution Subscriber.

System summary

The Experion PKS Notification System uses dynamic communication services to provide comprehensive generation, distribution, management and recovery of all notifications from process alarms through operator change classifications.

The Notification System in no way impacts the controller's ability to execute its primary control functions including interaction with I/O devices and peer control with other control devices.

Alarm and Event Processing

Alarm indication

The Station application provides an Alarm Summary display that lets you view all the current and unacknowledged alarms in the system or just those alarms with configured priority level of high and/or urgent, or just those assigned to a given area.

While the Alarm Summary display shown in Figure 4 is the focal point for viewing alarms, individual alarms can appear on any display including Detail, Group, Trend, and user built schematics. Alarms that are configured for a priority of journal are not displayed but can be included in a report or logged in a printed journal.

Alarm indications are displayed in this general format on the Summary display:

- <Date/Time><Area><Point ID><Alarm Type><Priority><Point Description>
<Value><Units>

The following table provides a more detailed description for each of the display fields.

Display Field	Description
Date/Time	Shows the date and time the Experion PKS Server received the alarm.
Area	Shows Server associated with given area assignment.
Point ID	The name of the alarm source. For Experion PKS, this is the CM tag name or the NDM coded identification tag. Up to 16 characters can be displayed.
Alarm Type	The alarm condition/PVpnt type being indicated. Up to 6 characters can be displayed.
Priority	Shows the priority assignment of the alarm: U for urgent, H for high, and L for low. Can also show the severity assignment of the alarm, if applicable. Severity can be a number between 0 to 15.
Point Description	The description associated with the point in alarm. For Experion PKS, this is the configurable Basic Function Block description with up to 24 characters.
Value	Not applicable for Experion PKS points.

Display Field	Description
Units	Not applicable for Experion PKS points.

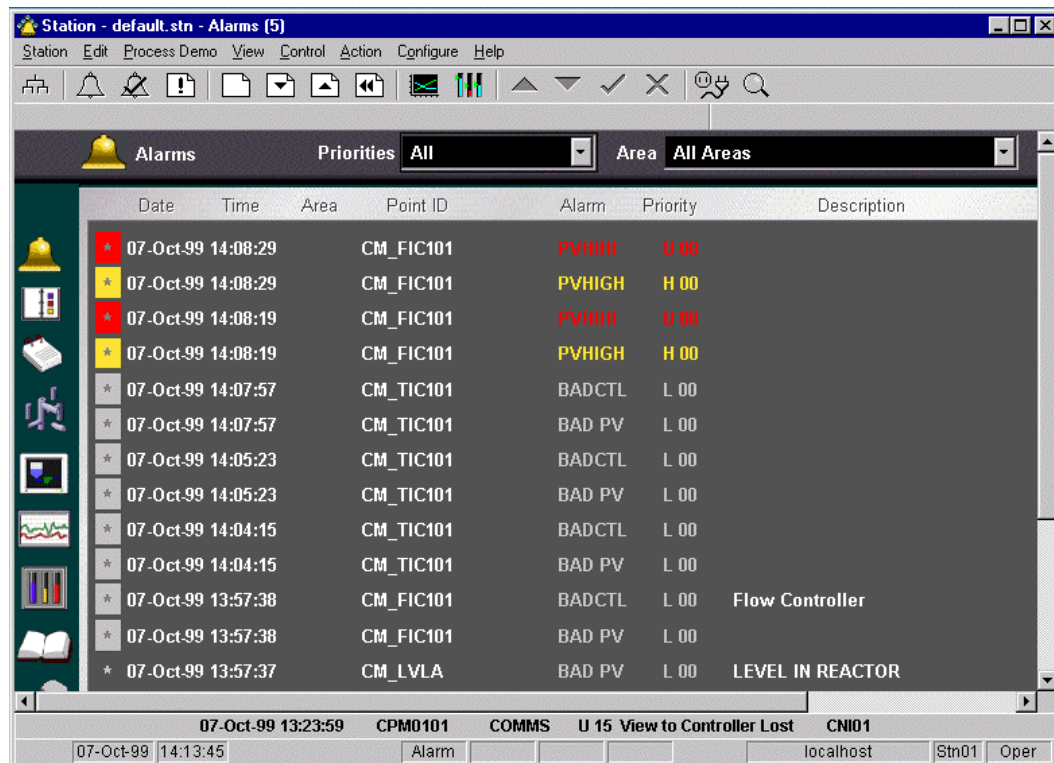


Figure 4 High alarm summary display example.

One of these alarm indicators may prefix the alarm field, as noted below and shown in Figure 5. (Note that color is used to show the relative priority of the alarm such as Red for Urgent , Yellow for High, and Gray for Low.)

If Indicator is . . .		Then Alarm is . . .	
	Inverse flashing asterisk	Unacknowledged, and has returned to normal.	
	Flashing asterisk	Unacknowledged, and still in alarm.	
	Asterisk	Acknowledged, and still in alarm.	
	Inverse flashing dash	Unacknowledged, and alarm is disabled.	

Total unacknowledged alarms: 6
Total acknowledged & still in alarm: 0

 Unacknowledged & returned to normal
 Unacknowledged & in alarm
 Acknowledged & still in alarm

Figure 5 Example of alarm indicator field on alarm summary display.

Figure 6 is a graphic summary of basic alarm indication for quick reference.

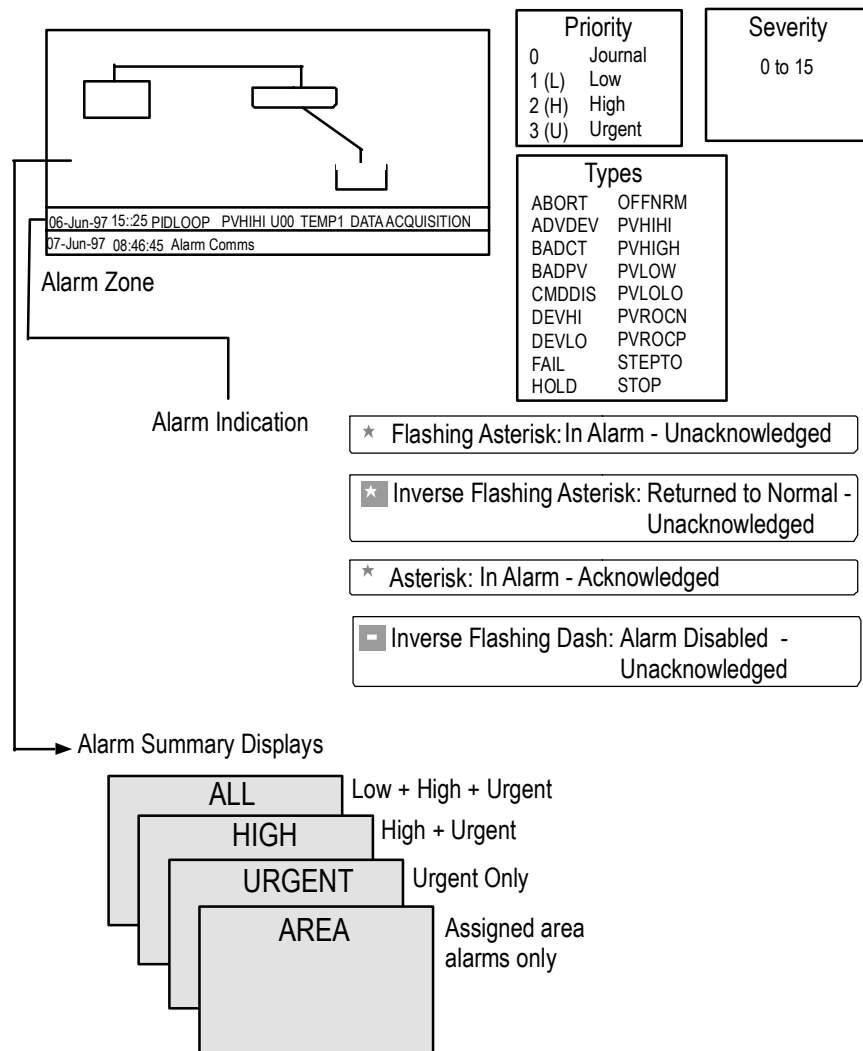


Figure 6 Alarm indication summary.

Alarm types

The alarm types associated with function blocks in a Control Module or Sequential Control Module are listed below.

Alarm Definition	Alarm Type Text	Function Block
Control Module Alarms		
No alarm exists	blank	All
Deviation from Advisory Setpoint	ADVDEV	PID
High Deviation from Setpoint	DEVHI	PID
Low Deviation from Setpoint	DEVLO	PID
PV Rate of Change Exceeded configured rate in negative (descending) direction	PVROCN	DATAACQ
PV Rate of Change Exceeded configured rate in positive (ascending) direction	PVROCP	DATAACQ
PV is Off Normal	OFFNRM	DEVCTL
PV High Trip Point Exceeded	PVHIGH	DATAACQ
PV Low Trip Point Exceeded	PVLOW	DATAACQ
PV High High Trip Point Exceeded	PVHIHI	DATAACQ
PV Low Low Trip Point Exceeded	PVLOLO	DATAACQ
Bad Control	BADCTL	PID
PV is BAD	BADPV	DATAACQ, DEVCTL

Alarm Definition	Alarm Type Text	Function Block
SCM Alarms		
No alarm exists	blank	All
Abnormal Execution Failure	FAIL	SCM Container
Step Timeout Alarm	STEPTO	SCM Container
STATE Alarm - Hold	HOLD	SCM Container
STATE Alarm - Stop	STOP	SCM Container
STATE Alarm - Abort	ABORT	SCM Container

NDM Overview

The Network Diagnostic Manager (NDM) monitors the devices physically present that are not explicitly configured as part of the user's control strategy. NDM employs a periodic 2 pass cycle to automatically {1} add/remove devices of interest to the scan list and {2} monitor the devices of interest on the scan list. More specifically, the first pass is used to search the Supervisory network, all slots within supervisory chassis, all downlinks, and all slots within remote chassis for devices of interest. As devices of interest are found, they are added to the scan list. The second pass is used to monitor the devices on the scan list and generate notifications for noteworthy events (e.g. cable status, fault status, etc). Note that while a device of interest is trivially detected in the first pass, a device is removed from the scan list if NDM is unable to communicate with the particular device 4 times in a row.

NDM detects various Critical Communication Failures and stops the CDA Server service to allow for redundant server failover. Prior to stopping the CDA Server service, an indication of the specific communication fault detected is both appended to the Error Log and posted as a diagnostic alarm. Note that NDM does not have the facilities to determine whether or not it is running in a primary synchronized server. As a consequence, stopping the CDA Server service applies equally to a non-redundant server, the primary server (synchronized or not), and the backup server (synchronized or not). The rationale being that at least this consistent behavior results in user notification of the problem. NDM interprets the following conditions as critical communication failures:

PCIC/KTC Reports Major Fault Status

Error Log	"<PCIC/KTC> reported MajorFault"
Detection	NDM immediately begins processing to stop the CDA Server service. When scanning a PCIC/KTC device, the attributes of the device's Device Object are obtained. A NDM Critical Communication Fault has occurred if the PCIC/KTC Device Object Status attribute indicates a Major Recoverable/Unrecoverable Fault.

Lost Communication With PCIC/KTC

Error Log	"Lost comms with <PCIC/KTC>"
Detection	NDM is unable to communicate with the PCIC/KTC after 3 consecutive attempts. In general, a device is removed from the scan list if NDM is unable to communicate with the particular device 4 times in a row. However, a NDM Critical Communication Fault has occurred if the device is a PCIC/KTC.

Unable to communicate through configured RSLinx driver

Error Log	"No comms through <driver>"
Detection	NDM is unable to communicate with the PCIC/KTC after 3 consecutive attempts. Similar to trigger 2 in that NDM cannot communicate with the PCIC/KTC, but this trigger covers the rare scenario for which NDM cannot communicate with a PCIC/KTC for the first time (i.e. after NDM has detected a new ControlNet RSLinx driver).

Lost communication with RSLinx

Error Log	"Lost comms with RSLinx"
Detection	After NDM verifies that the RSLinx process is not running, NDM immediately begins processing to stop the CDA Server service. After every unsuccessful attempt to perform either an Unconnected Send or a Driver List Query to RSLinx, NDM checks to ensure that the RSLinx process is still running.

PCIC/KTC Lonely on ControlNet

Error Log	"<PCIC/KTC> lonely on CNet"
Detection	NDM detects PCIC/KTC lonely condition after 3 consecutive attempts. When scanning a PCIC/KTC device, the attributes of the device's ControlNet Object are obtained. A NDM Critical Communication Fault has occurred if the PCIC/KTC indicates a lonely on ControlNet condition after 3 attempts.

NDM tag-coding scheme

The NDM uses an automatic tag-coding scheme to identify physical devices that are resident in the Experion PKS system architecture. The NDM scans all devices present in the system even if the associated Control Processor module (CPM) is not configured.

The format of the automatic tag coding scheme is based on the display abbreviation for a given device, as noted previously, appended with appropriate numeric code data. The following table describes the numeric codes in terms of lower case letter pairs that represent a value from 00 to 99. (Note that the user-configured RSLinx driver name appears as a prefix in the description field for all device/driver notifications in the Event display.)

If the displayed tag format is . . .	Then, it means the notification is for the . . .
dd	The RSLinx driver ID. It identifies the RSLinx driver when more than one driver of the same type exists. It is not related to the user-configured driver name, but rather a numerical value derived from the zero-based offset into the RSLinx configured driver list. It can also identify a device residing on the Supervisory Ethernet segment. For example, if ENETDRVR05 is the NDM generated tag for a RSLinx Ethernet driver, the tag generated for the ENI associated with this driver is ENET05. Any event/alarm is reported against this driver.
ddss	The RSLinx driver ID (dd) plus the chassis slot (ss). It identifies a device residing in the same chassis as an ENI attached to the Supervisory Ethernet segment. For example, if ENETDRVR05 is the NDM generated tag for an RSLinx Ethernet driver, the tag generated for the downlink CNI residing in slot 03 of the same Controller chassis as the ENI with tag ENET05 is CNI0503. Where 05 equals the Ethernet driver ID (dd) and 03 equals the slot location (ss) of the downlink CNI. (Note that CNI0503 forms the head of an I/O subnet against which the event/alarm is reported.)
ddssmm	RSLinx Driver ID (dd) plus chassis slot (ss) and MAC ID (mm). It identifies a device residing on a ControlNet subnet with a path through a chassis attached to the Supervisory Ethernet. For example, if ENETDRVR05 is the NDM generated tag for an RSLinx Ethernet driver and CNI0503 is the generated tag for the downlink CNI residing in the Controller chassis with ENI tag ENET05, then the tag generated for the Fieldbus Linking Device with a MAC ID of 02 on the ControlNet subnet node is LD050302 against which the event/alarm is reported.
ddmm	RSLinx driver ID (dd) plus MAC ID (mm). It identifies a device residing on the Supervisory ControlNet segment. For example, If CNETDRVR00 is the NDM generated tag for RSLinx ControlNet driver, the tag generated for the PCIC card with an address of 24 in this Supervisory ControlNet segment is PCIC0024. The PCIC card is installed in the PC.

If the displayed tag format is . . .	Then, it means the notification is for the . . .
ddmmss	RSLink driver ID (dd) plus MAC ID (mm) and chassis slot (ss). It identifies a device residing in the same chassis as the supervisory CNI. For example, , If CNETDRVR00 is the NDM generated tag for RSLink ControlNet driver and the tag generated for the supervisory CNI with a MAC ID of 01 is CNI0001, then the tag generated for the downlink CNI located in slot 3 of the Controller chassis is CNI000103 against which the event/alarm is reported.
ddmmsscc	RSLink driver ID (dd) plus MAC ID (mm), chassis slot (ss), and ControlNet subnet MAC ID (cc). It identifies a device residing on a ControlNet subnet with a path through a chassis connected to the supervisory ControlNet. For example, If CNETDRVR00 is the NDM tag generated for the ControlNet driver, CNI0001 is the tag generated for the supervisory CNI, and CNI000103 is the tag generated for the downlink CNI that is connected to supervisory CNI, then the tag generated for a Fieldbus Linking Device with a MAC ID of 02 residing on the ControlNet subnet is LD00010302 against which the event/alarm is reported.

Figure 7 shows an example of NDM generated notification tags in the Alarm Summary display for reference.

Date	Time	Area	Point ID	Alarm	Priority	Description	Value
08-Jun-00	13:54:13		CNI000211	CNET	H 00	AB_PCIC-1: UMAX Exceeded	
08-Jun-00	13:51:18		NDM	COMMS	H 00	Intermittent Comm Failure	
08-Jun-00	13:47:19		CNI0002	CNET	H 00	AB_PCIC-1: Cable Failure	CableA
08-Jun-00	13:47:17		CNI00011309	CNET	H 00	AB_PCIC-1: Cable Failure	CableA
08-Jun-00	13:47:10		PCIC0022	CNET	H 00	AB_PCIC-1: Cable Failure	CableA
08-Jun-00	13:47:08		CNI0001	CNET	H 00	AB_PCIC-1: Cable Failure	CableA
08-Jun-00	13:41:38		CNI00011310	CNET	H 00	AB_PCIC-1: Invalid Keeper	

Figure 7 Example of NDM generated tags in the Alarm Summary display.

Figure 8 shows an example of NDM generated notification tags in the Event Summary display for reference.

Date	Time	Point ID	Event	Level	Description	Value
08-Jun-00	13:23:58	CNI00011308	CNET	H 00	AB_PCIC-1: Cable Failure	CableA
08-Jun-00	13:22:51	CNI00011308	INFO		AB_PCIC-1: Card Added	
08-Jun-00	13:22:28	CPM0203	OFFNET	U 15	Connection FAILED	
08-Jun-00	13:22:25	CPM0203	OFFNET	U 15	Connection TIMEOUT	
08-Jun-00	13:22:19	RM000209	INFO		AB_PCIC-1: Card Added	
08-Jun-00	13:21:38	CNI000213	FAIL	OK	AB_PCIC-1: Comm Failure	
08-Jun-00	13:21:38	RM000209	INFO	OK	AB_PCIC-1: Card Removed	
08-Jun-00	13:21:05	NDM	COMMS	H 00	Intermittent Comm Failure	
08-Jun-00	13:20:49	CNI000213	FAIL		AB_PCIC-1: Comm Failure	
08-Jun-00	13:20:44	CNI00011310	CNET	H 00	AB_PCIC-1: Cable Failure	CableA

Figure 8 Example of NDM generated tags in the Event Summary display.



REFERENCE - INTERNAL

Refer to the *Experion PKS Troubleshooting and Maintenance Guide* for a detailed description of all the NDM generated notifications.

Example of NDM tag decoding

Figure 9 illustrates the CNI tag codes that would be generated to report a notification against the given partial network topology. Since this topology has a ControlNet Supervisory network, only one RSLinx driver is configured and by default its ID is zero. This diagram contains a Redundant Chassis Pair (RCP): the chassis on the left is currently operating as the Primary and the chassis on the right is the Secondary. If an error is detected in the downlink CNI in the Primary chassis, the notification is reported for tag CNI000303. This identifies the CNI as being located in the number 3 slot in the Controller chassis that has a CNI with a MAC address of 03 for the Supervisory ControlNet (with RSLinx driver ID #0).



ATTENTION

In a system with an RCP, a redundant Controller switchover can result in a changed tag code reference for a Bad CNI.

For example, if the downlink CNI in the Primary chassis in Figure 9 is Bad before switchover, it is displayed as a notification with a tag reference CNI000303. After switchover occurs, the network path to the Bad downlink CNI changes to go through the Secondary chassis and its notification tag reference changes to CNI000403. It is the same downlink CNI but the network path reflects the switch between the Primary and Secondary chassis including the logical change in the MAC address assignments.

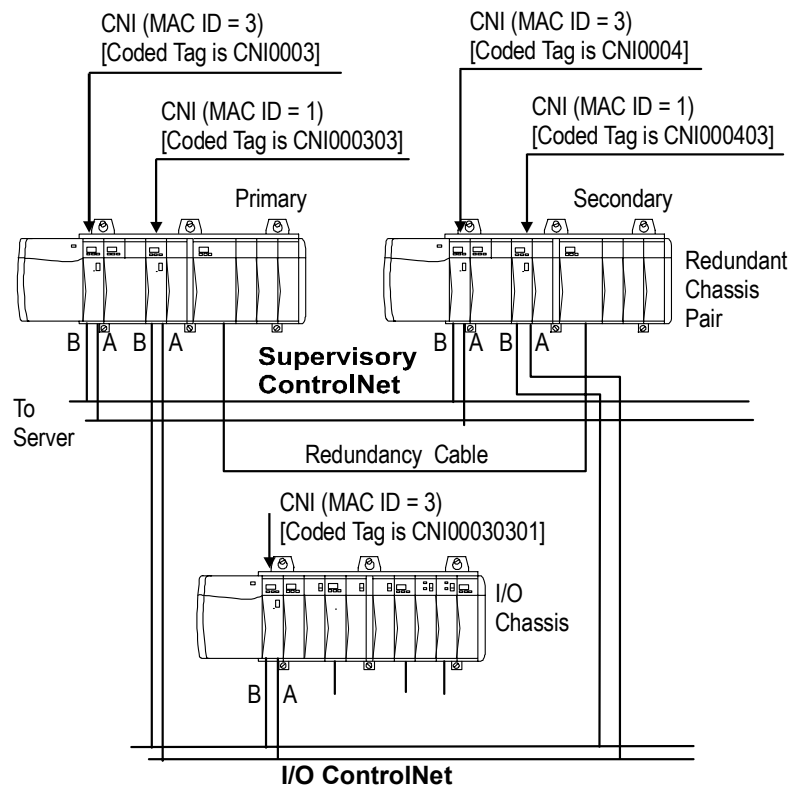


Figure 9 Typical NDM coded tag references for CNIs.

Multiple displays

The Station application provides several standard operator displays as well as the ability to view user built schematics. These displays allow an operator to quickly access more information about a point in alarm with just a keystroke or a command field entry. An operator can also invoke the Control Builder application to monitor the loaded Control Module, if limited access to configurable function block parameters is required. .

Figure 10 is a general graphic representation of how an operator may interact with the given Station displays to progressively reveal more data about a given point in alarm, including the invoking of the Control Builder application.

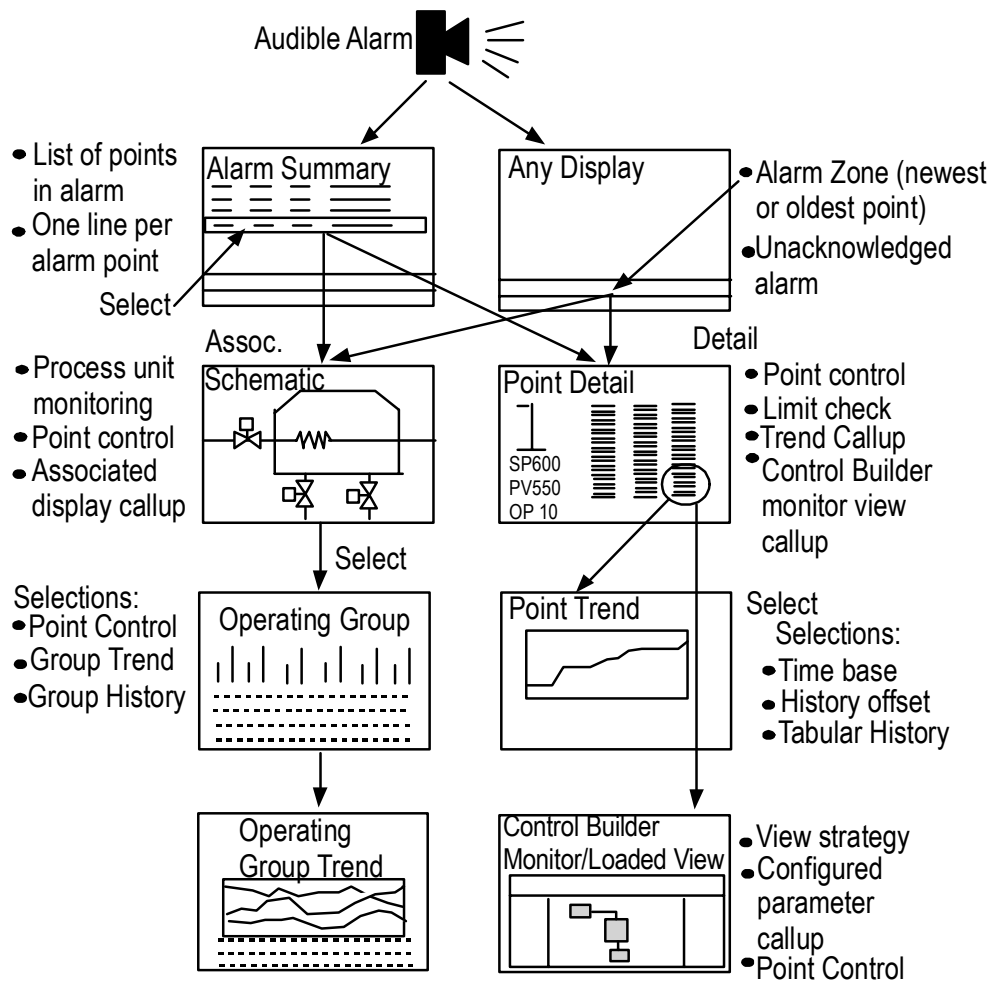


Figure 10 Station display interaction for data disclosure.

Alarm suppression

You can suppress the display of alarm information on an individual Control Module (CM) and Sequential Control Module (SCM), area wide, or system wide basis through the respective Enable/Disable function. The general actions associated with a given enable or disable function are summarized below.

If you enable or disable alarms on. . .	Then, enable action is. . .	Or disable action is. . .
Per CM or SCM basis (This means you must enable/disable each contained CM and/or SCM individually.)	CM/SCM recreates all outstanding alarms. Server forwards any alarms for that point to the alarm summary and the alarm zone. Server journals the alarms.	CM/SCM generates a "disable" event to cover all alarms for contained function blocks. Server treats current alarms reported against the CM/SCM as "disabled", so once they are acknowledged, they are removed from the alarm summary. CM/SCM stops reporting any alarms against the CM/SCM. Server does not report new alarms from the disabled point to the alarm summary. Server does not receive alarms from disabled CMs/SCMs. Alarms are disabled at the source. Server does not journal alarms for a disabled point.

If you enable or disable alarms on. . .	Then, enable action is. . .	Or disable action is. . .
Area Wide basis	Server forwards any alarms for that area to the alarm summary, event journal, and alarm zone. Recreates alarms for all points still in alarm for the affected area on the alarm summary, event journal, and alarm zone.	Server treats alarms for CMs /SCMs assigned to the disabled area as “disabled”, so once they are acknowledged, they are removed from the alarm summary. Server does not report new alarms from the disabled area to the alarm summary. Server does not journal alarms for a disabled area.
System Wide basis	Server forwards any alarms to the alarm summary and alarm zone. Recreates alarms for all points still in alarm on the alarm summary, event journal, and alarm zone.	Server treats alarms for CMs as “disabled”, so once they are acknowledged, they are removed from the alarm summary. Server does not report new alarms to the alarm summary. Server does not journal alarms.

Notification System

Alarm and Event Processing

Display of alarms is also impacted by state changes in the Control Module and Control Processor as well as the alarm itself, as noted below.

If. . .	Then, . . .
Alarm priority is changed to NONE or JOURNAL	The alarm is removed from the Alarm Summary. (Note that a change in priority to NONE is recorded as an event in the Event Summary.)
Alarm returns to normal	Alarm is shown as "returned to normal" and is removed from the Alarm Summary when acknowledged.
Control Processor (CPM) transitions from Run to Idle	All alarms for all CMs/SCMs in the associated CEE are shown as "disabled", so once they are acknowledged, they are removed from the alarm summary.
CPM fails	All alarms for all CMs/SCMs in the associated CEE are frozen at their last state until the CEE recovers.
CPM transitions from Idle to Run (warm start)	The points recreate any currently outstanding alarm. For warm start transition, the points reexecute the alarm algorithms to create alarms as new.
User deletes CM	All CM points are deleted from the Experion PKS Server, which also deletes any associated alarms from the Alarm Summary.
User inactivates CM	All the alarms for this CM are shown as "disabled", so once they are acknowledged, they are removed from the Alarm Summary.
User reactivates CM	Server reports any alarms detected during the first execution of each function block the same as steady-state detection of block alarm condition.

Event indication

The Station application provides an Event Summary display that lets you view a journal of events that have occurred over time listed in reverse chronological order.

Event indications are displayed in this general format on the Summary display:

- <Time><Point ID><Event><Level>< Description><Value><Field Time>

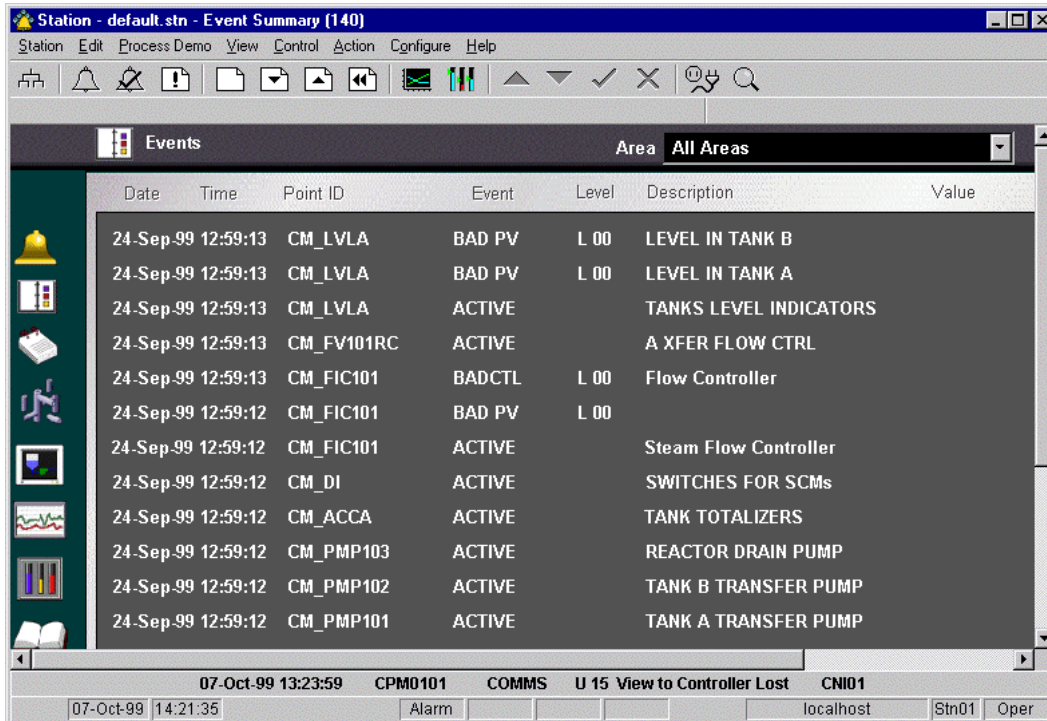
Where:

Display Field	Description
Time	Shows the date and time the event was journalized.
Point ID	The name of the device reporting the event. Up to 16 characters can be displayed.
Event	The event being indicated. Up to 6 characters can be displayed.
Level	Shows the priority level assignment of the event. U for urgent, H for high, and L for low. Or, the User Name entered for Control Builder login for Control Builder journal events.
Description	The description associated with the event. Up to 30 characters.
Value	For Experion PKS, additional descriptor data. Up to 14 characters. Or, the new value, for Control Builder journal events.
Field Time	Not Used

Figure 11 shows an example of an Event Summary display. Its format is similar to the Alarm Summary display.

Notification System

Alarm and Event Processing



Date	Time	Point ID	Event	Level	Description	Value
24-Sep-99	12:59:13	CM_LVLA	BAD PV	L 00	LEVEL IN TANK B	
24-Sep-99	12:59:13	CM_LVLA	BAD PV	L 00	LEVEL IN TANK A	
24-Sep-99	12:59:13	CM_LVLA	ACTIVE		TANKS LEVEL INDICATORS	
24-Sep-99	12:59:13	CM_FV101RC	ACTIVE		A XFER FLOW CTRL	
24-Sep-99	12:59:13	CM_FIC101	BADCTL	L 00	Flow Controller	
24-Sep-99	12:59:12	CM_FIC101	BAD PV	L 00		
24-Sep-99	12:59:12	CM_FIC101	ACTIVE		Steam Flow Controller	
24-Sep-99	12:59:12	CM_DI	ACTIVE		SWITCHES FOR SCMs	
24-Sep-99	12:59:12	CM_ACCA	ACTIVE		TANK TOTALIZERS	
24-Sep-99	12:59:12	CM_PMP103	ACTIVE		REACTOR DRAIN PUMP	
24-Sep-99	12:59:12	CM_PMP102	ACTIVE		TANK B TRANSFER PUMP	
24-Sep-99	12:59:12	CM_PMP101	ACTIVE		TANK A TRANSFER PUMP	

Figure 11 Event summary display example.

Event types

Events can be broadly identified as either a system or an operator change type. If a MESSAGE function block client executes its message, the sent message is also logged as an event. The acknowledgement and confirmation of a Confirmation type message are also logged as events. A system type event can represent one of these three system notification classes previously described in this document.

- System State
- System Diagnostic
- System Information



ATTENTION

SCM status events are reported as alarm exceptions related to the execution of the STEP and TRANSITION function blocks as listed under the Alarm Types heading.

Basic system state changes that can trigger an event are summarized below.

State Change	Applies to Equipment	Comment
Fail to Idle	CPM	CPM must go from Fail to Idle state, and requires a database load prior to going to Run.
Fail to Run	IOM	Configured IOMs are either Failed (not okay) or Run (okay).
Fail, Idle, Run to Offnet	Hybrid Controller	Only the controller as a whole can be Offnet. Server has lost visibility of Hybrid controller.
Idle to Run	Hybrid Controller	
Idle, Run to Fail	CPM, IOM	Transition to failure is on a per card basis. For Control Processor, this is treated as "Offnet".
Offnet to Idle, Run, Fail	Hybrid controller	Offnet to Fail is the same as staying Offnet. No new notification is sent.
Run to Idle	Hybrid Controller	
Switchover	Redundant Chassis Pair – Hybrid Controllers	Secondary chassis becomes Primary on switchover. Related events are posted between Recovery Begin (RCVBGN) and Recovery End (RECVEND) events.

Notification System

Alarm and Event Processing

System diagnostic and system information events are closely related and may also be thought of as equipment failure and Control Processor exception status reporting. Some basic diagnostic classifications are summarized below for general reference.

Diagnostic Class	Where Detected	Description
CPM Fail	NM	The Notifications Manager detects that the CPM is in the Failed State. This means the CPM can respond to CNET messages but not CDA messages.
CPM Not Responding	NM	The Notification Manager detects the CPM is not responding to CDA or CNET messages. This could be caused by CPM hard failure, such as watchdog timeout, or loss of communications on ControlNet.
Hardware Failure	CPM	Indicates a partial hardware failure detected by the diagnostic manager or some other diagnostic agent. The failure may or may not be recoverable without manual intervention. An example would be a RAM parity circuit failure.
IOM Failure	CPM	Control Processor's I/O Manager detects a failed IOM. Note that a state change event (IOM state changes to Fail) is generated in addition to the system diagnostic event.
IOM Partial Failure	IOM	For those IOMs that can self-detect faults, the IOM produces a status indication. The CPM generates the IOM diagnostic notification.
Network Diagnostic Notification	NDM	The Network Diagnostic Manager detects status of all CNIs in the Supervisory and I/O networks and reports status to the NM.

Diagnostic Class	Where Detected	Description
Poor Execution	CEE	The CEE detects that its execution of function blocks is degraded. An example would be execution cycle overruns.
System Diagnostic Notification	CPM	Indicates a system diagnostic failure where there is a detected problem, which is not related to the CPM or the Hybrid Controller, and the CPM remains in an "OK" state. An example would be CPU free too low. If the CDA server fails, Experion PKS produces a "NOTCLI subscription failed" system diagnostic.

Operators can make changes to selected parameters through these displays.

- User built schematics in Station
- Station Group and Detail displays
- Run-time monitoring displays in Control Builder

Changes made through Station displays are recorded in the event journal along with the operator's name.

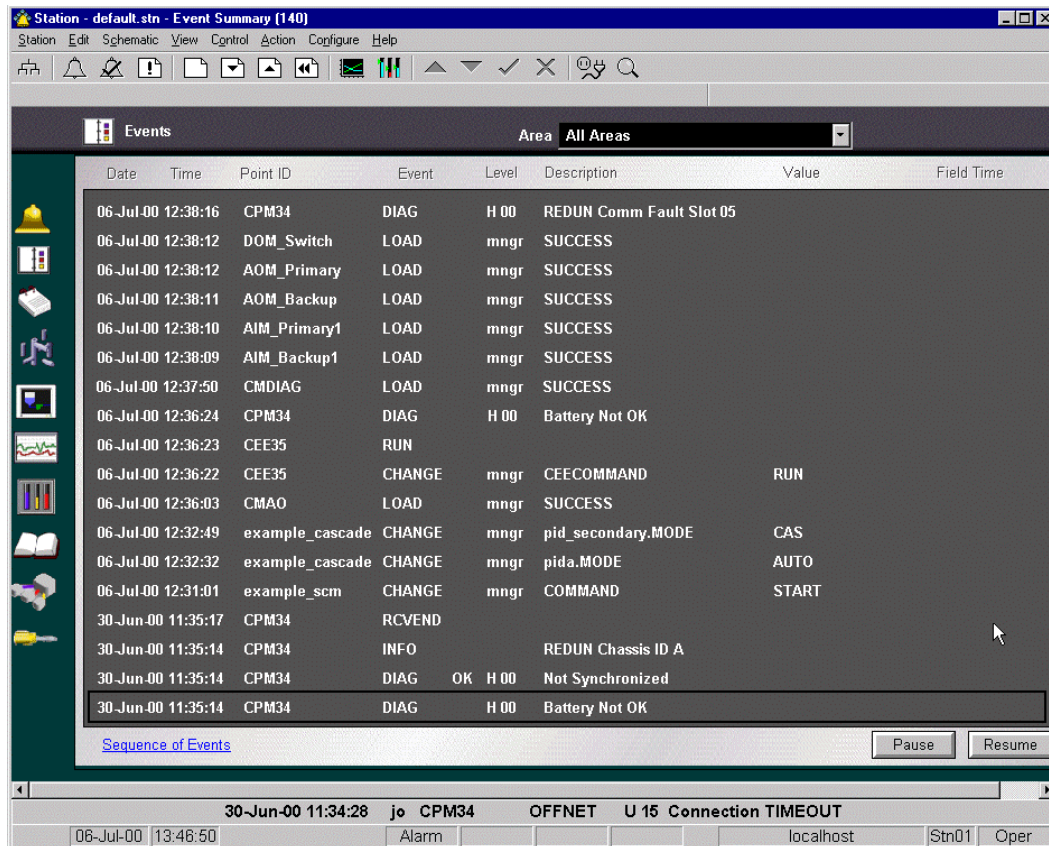
Changes made through Control Builder are recorded in the event journal along with the User Name specified during Control Builder login. The following types of operator initiated actions through Control Builder trigger event journal entries.

- All 'successful' parameter writes to the Controller including activate/inactivate.
- All 'attempts' of Controller loads/deletes. A failed load or delete is also entered in the event log.
- Successful restoration of Controller from snapshot.

Figure 12 shows an example of Control Builder events recorded for a user named "mngr" in the Events Summary display.

Notification System

Alarm and Event Processing



Station - default.stn - Event Summary (140)

Station Edit Schematic View Control Action Configure Help

Events Area All Areas

Date	Time	Point ID	Event	Level	Description	Value	Field Time
06-Jul-00	12:38:16	CPM34	DIAG	H 00	REDUN Comm Fault Slot 05		
06-Jul-00	12:38:12	DOM_Switch	LOAD	mngr	SUCCESS		
06-Jul-00	12:38:12	AOM_Primary	LOAD	mngr	SUCCESS		
06-Jul-00	12:38:11	AOM_Backup	LOAD	mngr	SUCCESS		
06-Jul-00	12:38:10	AIM_Primary1	LOAD	mngr	SUCCESS		
06-Jul-00	12:38:09	AIM_Backup1	LOAD	mngr	SUCCESS		
06-Jul-00	12:37:50	CMDIAG	LOAD	mngr	SUCCESS		
06-Jul-00	12:36:24	CPM34	DIAG	H 00	Battery Not OK		
06-Jul-00	12:36:23	CEE35	RUN				
06-Jul-00	12:36:22	CEE35	CHANGE	mngr	CEECOMMAND	RUN	
06-Jul-00	12:36:03	CMAO	LOAD	mngr	SUCCESS		
06-Jul-00	12:32:49	example_cascade	CHANGE	mngr	pid_secondary.MODE	CAS	
06-Jul-00	12:32:32	example_cascade	CHANGE	mngr	pid_a.MODE	AUTO	
06-Jul-00	12:31:01	example_scm	CHANGE	mngr	COMMAND	START	
30-Jun-00	11:35:17	CPM34	RCVEND				
30-Jun-00	11:35:14	CPM34	INFO		REDUN Chassis ID A		
30-Jun-00	11:35:14	CPM34	DIAG	OK H 00	Not Synchronized		
30-Jun-00	11:35:14	CPM34	DIAG	H 00	Battery Not OK		

Sequence of Events

Pause Resume

30-Jun-00 11:34:28 jo CPM34 OFFNET U 15 Connection TIMEOUT

06-Jul-00 13:46:50 Alarm localhost Stn01 Oper

Figure 12 Control Builder event journal entries example

A word about errorhandling log

If for some reason a Control Builder event journal entry cannot be made in the event log, an error is recorded in an Errlog_n.txt file, where n equals a number, in this directory location *c:\errohandling*.

An example of the entries in the Errlog_n.txt file is shown below for reference:

5/19/00 10:54:06 AM PS_E_WARNING (1LI .101.12394)	CB Journaling failed, CN2_Test2.IN_F32.PV[0] changed to 5555
5/19/00 10:55:21 AM PS_E_WARNING (1LI .101.12394)	CB Journaling failed, CN2_N08.START.IN changed to OFF
5/19/00 10:55:23 AM PS_E_WARNING (1LI .101.12394)	CB Journaling failed, CN2_N08.START.IN changed to ON
5/19/00 10:56:29 AM PS_E_WARNING (1LI .101.12394)	CB Journaling failed, CN2_N04.EXECSTATE changed to ACTIVE
5/19/00 11:00:09 AM PS_E_WARNING (1LI .101.12394)	CB Journaling failed, CN2_Test2.IN_TEXT.STR[0] changed to mari
5/19/00 11:00:15 AM PS_E_WARNING (1LI .101.12394)	CB Journaling failed, CN2_Test2.IN_FLAG.PVFL[0] changed to OFF
5/19/00 11:00:24 AM PS_E_WARNING (1LI .101.12394)	CB Journaling failed, CN2_Test2.IN_INT.PV[0] changed to 0
5/19/00 11:00:55 AM PS_E_WARNING (1LI .101.12394)	CB Journaling failed, CN2_Test2.IN_INT.PV[0] changed to 0
5/19/00 11:01:12 AM PS_E_WARNING (1LI .101.12394)	CB Journaling failed, CN2_Test2.IN_F32.PV[0] changed to 0
5/19/00 11:02:32 AM PS_E_WARNING (1LI .101.12394)	CB Journaling failed, CN2_Test2.IN_F32.PV[0] changed to 66.66
5/19/00 11:04:21 AM PS_E_WARNING (1LI .101.12394)	CB Journaling failed, CN2_Test2.Constant.PV changed to 5550

Message Processing

Message indication

The Station application provides a Message Summary display that lets you view all the current messages that have been sent. A message is displayed along with the time it was generated and the name of the Control Module that contains the MESSAGE function block associated with the message as shown in Figure 13.

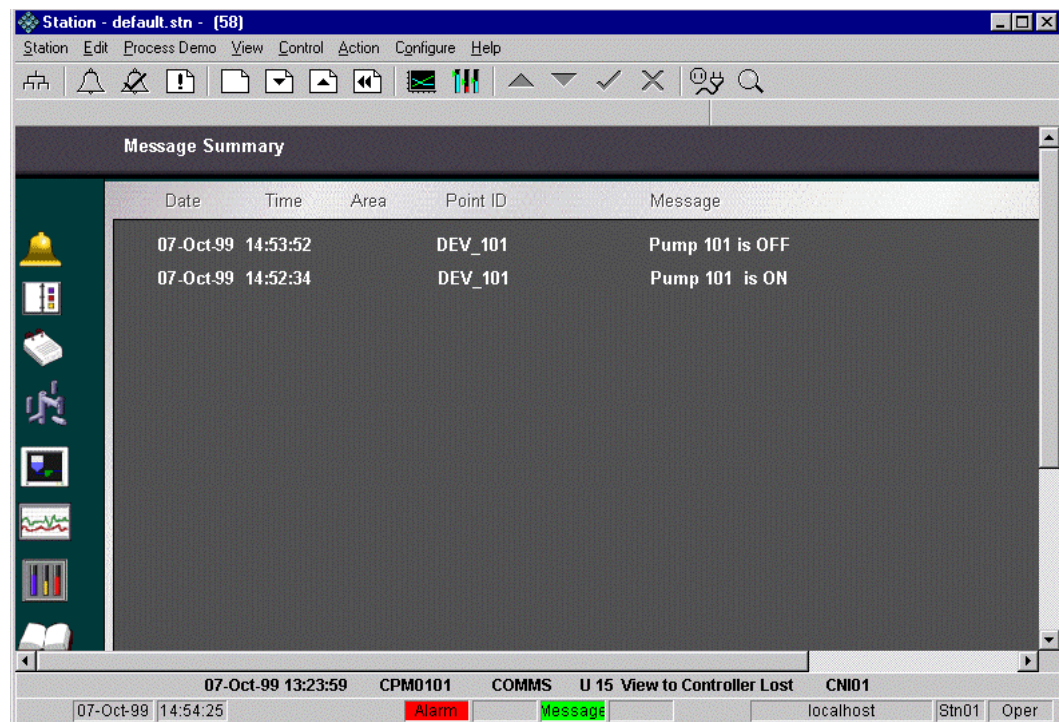


Figure 13 Station Message Summary display format.

Message types

A client-triggered message is configured to provide operation information or a request for operator action as part of the normal or abnormal execution of the control strategy configured within Control Builder. A message can be one of these two types.

Message Type	Description
Information	Requires no specific action from the operator. It is provided for information only.
Confirmation	Waits for operator to confirm. A client trigger pulses the corresponding SENDFLAG[n] to send the MESSAGE[n] to the Experion PKS Server. The client of the MESSAGE block checks for the confirmed parameter (CONFIRMED[n]) to be set to True. The CONFIRMED[n] parameter indicates whether the MESSAGE block has received a confirmation. An operator must acknowledge a confirmation message twice to confirm it through the Message Summary display.

A typical confirmation message would have a STEP output client for the corresponding MESSAGE block SENDFL[n] parameter and a TRANSITION condition client for the corresponding MESSAGE block CONFIRMED[n] parameter. Where “n” is the number of the configured message (0 to 15). The following Figures 14 and 15 show sample configuration entries for STEP and TRANSITION blocks in an SCM that are used as clients for a MESSAGE block (MESSAGE5) in a Control Module (DEV_101). The number 2 message in the MESSAGE block is configured as a confirmation type.

#	Step Outputs
1	- example_pid.pida.modeattr:=2
2	- example_pid.pida.sp:=75.
3	- example_pid.pida.sp:=example_som.rectarget[1]
4	- example_som.histvalue[1]:=example_pid.pida.pv
5	- DEV_101.MESSAGE2.SENDFL[2] := 2
6	- DEV_101.MESSAGE2.SENDFL[3] := 3

Figure 14 Typical STEP output configuration for configured message in a MESSAGE block.

#	Transition Conditions
1	example_som.command=3
2	example_pid.pida.pv>50.
3	example_motor.DEVCTLA.gpv=4
4	DEV_101.MESSAGE2.CONFIRMED[2] = 2

Figure 15 Typical TRANSITION condition configuration for configured confirmation message in a MESSAGE block.

The configured message string is stored in the System Repository instead of the Controller. This means that you must configure the messages before loading the Control Module to the Controller. You can not change a message through the Monitoring tab after the Control Module is loaded to the Controller. To change a message, you must reconfigure it through the Project tab and reload the Control Module to the Controller.

Since the SENDFLAG[n] is a pulse trigger, it is automatically set to False during the next execution cycle. This means the MESSAGE block is ready to send the same message again in the next cycle. To avoid continuous messaging, you can use a FLAG block in conjunction with a PULSE block in the loop to the client trigger.

Notification Reporting and Acknowledgment

Reporting scheme

How the predetermined reporting structure correlates with the notification classes is summarized below. This directs a notification to the end-point client or clients that provide the appropriate user view of the notification.

Notification Class	Reporting Scheme
Operator Change (Station Displays only)	Sent to the operator change journal and the printer journal. Note that these events bypass the Notification Manager.
Process Alarm	Sent to the alarm journal, printer journal, and event journal.
System Diagnostic	Sent to the alarm journal, printer journal, and event journal.
System Information	Sent to the event journal and the printer journal.
System State	Sent to the event journal and the printer journal. System states themselves are accessible as parametric attributes of a block and are available on system status displays.
Third-Party Alarms	Sent to the alarm journal, printer journal, and event journal.

Acknowledgment scheme

Operators can acknowledge receipt of a notification related to one of these three notification classes through Station displays.

- Process Alarms with priority level other than journal
- System diagnostics with priority level other than none or journal
- Messages

Alarm and diagnostic acknowledge functions are managed entirely within the server and Station subsystems. The acknowledge status of a detected condition is stored on a per condition basis within the tagged point in the server RTDB. The tagged point maps to the CM tagged block of the same name. Operators can acknowledge alarms through the Station Alarm Summary display. In systems with multiple operator stations, you only have to acknowledge a condition through the alarm summary in one station to have the condition acknowledged in all alarm summaries on all stations.

The message acknowledge function is managed entirely within the server and Station subsystems. Operators can acknowledge both information and confirmation type messages through the Station Message Summary display. The information messages are removed upon acknowledgement. The confirmation messages are confirmed upon a second acknowledgement.

Notification Recovery

Recovery initiators

The Experion PKS notification system automatically initiates a notification recovery routine in response to the following system conditions:

- Server Startup
- Control Processor Startup
- ControlNet Failure and Recovery (Dual Cable or CNETB)
- Area and System Wide Enable



ATTENTION

The server receives notifications that are assigned to a different area than one being recovered, but it discards these as duplicates of current information. Recovered notifications for a currently disabled area are also discarded.

Recovery routine

In general, a notification recovery routine consists of the Notification Manager commanding each CEE to recover all currently outstanding alarms and system diagnostics. It recreates data as required for these notification classes:

- Process Alarms for those conditions, which are in alarm at the time of recovery, are recreated.
- CPM and CEE System States are recreated as part of a full recovery.
- System Diagnostic conditions that are true at the time of recovery are recreated.
- Third-Party Alarms are handled the same as Process Alarms above.

Frequently Asked Questions About Experion PKS Alarming

Question	Answer
How many alarms can the Experion PKS System handle concurrently?	1000 active alarms.
Is this the size of the alarm page buffer?	Yes
What happens if there is an alarm overflow?	In general, newer alarms replace older alarms. The Experion PKS Server can request full alarm regeneration, if needed, and the ultimate data owning alarm blocks hold alarm information, in case of a system alarm overflow.
If the alarm summary page is full, what happens with the alarms that follow?	Newer alarms replace older alarms on the alarm summary. An algorithm is applied to discard the least important alarm from the list according to this schema. • Discard any duplicate alarm; then, • Discard lower priority alarm; then, • Discard oldest alarm.
Are overflow alarms stored in the Event log file?	All alarms received are placed in the Event journal.

Question	Answer
What happens to alarms when the Experion PKS Server is not running?	When the Server is restarted, all alarms generated by the Hybrid Controller are resent to the Server database. Alarms will be regenerated in response to any of the following events. • Server startup • Redundant Server failover • Re-enabling alarms system-wide, by area or by point • Control Processor startup • Redundant Controller switchover • ControlNet communications failure and recovery • Setting a CEE's state to RUN • Setting a CM's state to Active
When are alarms timestamped?	Alarms are timestamped when received at the Server database.
Can alarms be disabled during maintenance periods, plant commissioning or other circumstances?	Alarms can be disabled through several means, depending upon the extent of the silencing required. You can disable alarms on a point (CM or SCM), area, or system wide basis through Station.

Notification System

Frequently Asked Questions About Experion PKS Alarming

Fax Transmittal

Fax No.: (602) 313-3293

Reader Comments Ftw

To: Solution Support Center

From: Name

Date:

Title:

Company:

Address:

City:

State:

Zip:

Telephone:

Fax:

Experion PKS® Theory, Release 100, 1/03

Comments:

You may also call the Solution Support Center at 800-343-0228,
Email SSC at tac-response@honeywell.com, or write to:

Solution Support Center
Honeywell
MS-P13
2500 West Union Hills Drive
Phoenix, AZ 85027

Experion PKS®
Knowledge Building Tools

Honeywell

Honeywell
ACS – Industry Solutions
1100 Virginia Drive
Fort Washington, PA 19034