Logix 5000 Advanced Process Control and Drives Instructions

1756 ControlLogix, 1756 GuardLogix, 1769 CompactLogix, 1769 Compact GuardLogix, 1789 SoftLogix, 5069 CompactLogix, Emulate 5570

Publication# 1756-RM006L-EN-P

Rockwell Automation

Reference Manual

Original Instructions

Important User Information

Read this document and the documents listed in the additional resources section about installation, configuration, and operation of this equipment before you install, configure, operate, or maintain this product. Users are required to familiarize themselves with installation and wiring instructions in addition to requirements of all applicable codes, laws, and standards.

Activities including installation, adjustments, putting into service, use, assembly, disassembly, and maintenance are required to be carried out by suitably trained personnel in accordance with applicable code of practice.

If this equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

In no event will Rockwell Automation, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Rockwell Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by Rockwell Automation, Inc. with respect to use of information, circuits, equipment, or software described in this manual.

Reproduction of the contents of this manual, in whole or in part, without written permission of Rockwell Automation, Inc., is prohibited.

Throughout this manual, when necessary, we use notes to make you aware of safety considerations.

WARNING: Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.

ATTENTION: Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence.

IMPORTANT Identifies information that is critical for successful application and understanding of the product.

Labels may also be on or inside the equipment to provide specific precautions.

SHOCK HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.

BURN HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.

ARC FLASH HAZARD: Labels may be on or inside the equipment, for example, a motor control center, to alert people to potential Arc Flash. Arc Flash will cause severe injury or death. Wear proper Personal Protective Equipment (PPE). Follow ALL Regulatory requirements for safe work practices and for Personal Protective Equipment (PPE).

This manual includes new and updated information. Use these reference tables to locate changed information.

Global changes

None for this release.

New or enhanced features

Subject	Reason
PlantPAx instructions on page 257	Added PlantPAx instructions.

Use this locator to find the applicable Logix5000 controllers instruction manual for each instruction.

Logix5000 Controllers General Instructions Reference Manual 1756-RM003	Logix5000 Controllers Advanced Process Control and Drives and Equipment Phase and Sequence Instructions Reference Manual 1756-RM006	Logix5000 Controllers Motion Instructions Reference Manual MOTION-RM002
Absolute Value (ABS)	Alarm (ALM)	Master Driven Coordinated Control (MDCC)
Add (ADD)	Attach to Equipment Phase (PATT)	Motion Apply Axis Tuning (MAAT)
Analog Alarm (ALMA)	Attach to Equipment Sequence (SATT)	Motion Apply Hookup Diagnostics (MAHD)
Always False (AFI)	Coordinated Control (CC)	Motion Arm Output Cam (MAOC)
Arc Cosine (ACS, ACOS)	D Flip-Flop (DFF)	Motion Arm Registration (MAR)
Arc Sine (ASN, ASIN)	Deadtime (DEDT)	Motion Arm Watch (MAW)
Arc Tangent (ATN, ATAN)	Derivative (DERV)	Motion Axis Fault Reset (MAFR)
ASCII Chars in Buffer (ACB)	Detach from Equipment Phase (PDET)	Motion Axis Gear (MAG)
ASCII Clear Buffer (ACL)	Detach from Equipment Sequence (SDET)	Motion Axis Home (MAH)
ASCII Handshake Lines (AHL)	Discrete 3-State Device (D3SD)	Motion Axis Jog (MAJ)
ASCII Read (ARD)	Discrete 2-State Device (D2SD)	Motion Axis Move (MAM)
ASCII Read Line (ARL)	Enhanced PID (PIDE)	Motion Axis Position Cam (MAPC)
ASCII Test for Buffer Line (ABL)	Enhanced Select (ESEL)	Motion Axis Stop (MAS)
ASCII Write (AWT)	Equipment Phase Clear Failure (PCLF)	Motion Axis Time Cam (MATC)
ASCII Write Append (AWA)	Equipment Phase Command (PCMD)	Motion Axis Shutdown (MASD)
Bit Field Distribute (BTD)	Equipment Phase External Request (PXRQ)	Motion Axis Shutdown Reset (MASR)
Bit Field Distribute with Target (BTDT)	Equipment Phase Failure (PFL)	Motion Calculate Cam Profile (MCCP)
Bit Shift Left (BSL)	Equipment Phase New Parameters (PRNP)	Motion Coordinated Path Move (MCPM)
Bit Shift Right (BSR)	Equipment Phase Override Command (POVR)	Motion Calculate Slave Values (MCSV)
Bitwise And (AND)	Equipment Phase Paused (PPD)	Motion Coordinated Transform with Orientation (MCTO)
Bitwise (NOT)	Equipment Sequence Assign Sequence Identifier (SASI)	Motion Calculate Transform Position (MCTP)
Bitwise (OR)	Equipment Sequence Clear Failure (SCLF)	Motion Calculate Transform Position with Orientation (MCTPO)
Boolean AND (BAND)	Equipment Sequence command (SCMD)	Motion Change Dynamics (MCD)
Boolean Exclusive OR (BXOR)	Equipment Sequence Override (SOVR)	Motion Coordinated Change Dynamics (MCCD)
Boolean NOT (BNOT)	Function Generator (FGEN)	Motion Coordinated Circular Move (MCCM)
Boolean OR (BOR)	High Pass Filter (HPF)	Motion Coordinated Linear Move (MCLM)
Break (BRK)	High/Low Limit (HLL)	Motion Coordinated Shutdown (MCSD)
Breakpoints (BPT)	HMI Button Control (HMIBC)	Motion Coordinated Shutdown Reset (MCSR)
Clear (CLR)	Integrator (INTG)	Motion Coordinated Stop (MCS)
Compare (CMP)	Internal Model Control (IMC)	Motion Coordinated Transform (MCT)
Convert to BCD (TOD)	JK Flip-Flop (JKFF)	Motion Direct Drive Off (MDF)
Convert to Integer (FRD)	Lead-Lag (LDLG)	Motion Direct Drive On (MDO)
Copy File (COP), Synchronous Copy File (CPS)	Low Pass Filter (LPF)	Motion Direct Start (MDS)
Cosine (COS)	Maximum Capture (MAXC)	Motion Disarm Output Cam (MDOC)
Compute (CPT)	Minimum Capture (MINC)	Motion Disarm Registration (MDR)
Count down (CTD)	Modular Multivariable Control (MMC)	Motion Disarm Watch (MDW)

Logix5000 Controllers General Instructions Reference Manual 1756-RM003	Logix5000 Controllers Advanced Process Control and Drives and Equipment Phase and Sequence Instructions Reference Manual 1756-RM006	Logix5000 Controllers Motion Instructions Reference Manual MOTION-RM002
Count up (CTU)	Moving Average (MAVE)	Motion Group Shutdown (MGSD)
Count up/down CTUD	Moving Standard Deviation (MSTD)	Motion Group Shutdown Reset (MGSR)
Data Transition (DTR)	Multiplexer (MUX)	Motion Group Stop (MGS)
Degrees (DEG)	Notch Filter (NTCH)	Motion Group Strobe Position (MGSP)
Diagnostic Detect (DDT)	Phase State Complete (PSC)	Motion Redefine Position (MRP)
Digital Alarm (ALMD)	Position Proportional (POSP)	Motion Run Axis Tuning (MRAT)
DINT To String (DTOS)	Process Analog HART (PAH)	Motion Run Hookup Diagnostics (MRHD)
Divide (DIV)	Process Analog Input (PAI)	Motion Servo Off (MSF)
End of Transition (EOT)	Process Dual Sensor Analog Input (PAID)	Motion Servo On (MSO)
Equal to (EQU)	Process Multi Sensor Analog Input (PAIM)	
File Arithmetic (FAL)	Process Analog Output (PAO)	
File Bit Comparison (FBC)	Process Boolean Logic (PBL)	
FIFO Load (FFL)	Process Command Source (PCMDSRC)	
FIFO Unload (FFU)	Process Deadband Controller (PDBC)	
File Average (AVE)	Process Discrete Input (PDI)	
File Standard Deviation (STD)	Process Discrete Output (PDO)	
File Fill (FLL)	Process Dosing (PDOSE)	
File Sort (SRT)	Process Analog Fanout (PFO)	
Find String (FIND)	Process High or Low Selector (PHLS)	
For (FOR)	Process Interlocks (PINTLK)	
File Search and Compare (FSC)	Process Lead Lag Standby Motor Group (PLLS)	
Get System Value (GSV) and Set System Value (SST)	Process Motor (PMTR)	
Greater Than or Equal to (GEQ)	Process Permissives (PPERM)	
Greater than (GRT)	Process Proportional + Integral + Derivative (PPID)	
Insert String (INSERT)	Process Pressure/Temperature Compensated Flow (PPTC)	
Immediate Output (IOT)	Process Restart Inhibit (PRI)	
Is Infinity (IsINF)	Process Run Time and Start Counter (PRT)	
Is Not a Number (IsNAN)	Process Tank Strapping Table (PTST)	
Jump to Label (JMP) and Label (LBL)	Process Valve (PVLV)	
Jump to Subroutine (JSR), Subroutine (SBR), and Return (RET)	Process Valve Statistics (PVLVS)	
Jump to External Routine (JXR)	Proportional + Integral (PI)	
Less Than (LES)	Pulse Multiplier (PMUL)	
Less Than or Equal to (LEQ)	Ramp/Soak (RMPS)	
LIFO Load (LFL)	Rate Limiter (RLIM)	
LIFO Unload (LFU)	Reset Dominant (RESD)	
License Validation (LV)	Scale (SCL)	
Limit (LIM)	S-Curve (SCRV)	
Log Base (LOG)	Second-Order Controller (SOC)	
Lower to Case (LOWER)	Second-Order Lead Lag (LDL2)	
Masked Move (MVM)	Select (SEL)	
Masked Move with Target (MVMT)	Selected Negate (SNEG)	

Logix5000 Controllers General Instructions Reference Manual 1756-RM003	Logix5000 Controllers Advanced Process Control and Drives and Equipment Phase and Sequence Instructions Reference Manual 1756-RM006	Logix5000 Controllers Motion Instructions Reference Manual MOTION-RM002
Master Control Reset (MCR)	Selected Summer (SSUM)	
Masked Equal to (MEQ)	Set Dominant (SETD)	
Message (MSG)	Split Range Time Proportional (SRTP)	
Middle String (MID)	Totalizer (TOT)	
Modulo (MOD)	Up/Down Accumulator (UPDN)	
Move (MOV)		
Multiply (MUL)		
Natural Log (LN)		
Negate (NEG)		
Not Equal to (NEQ)		
No Operation (NOP)		
One Shot (ONS)		
One Shot Falling (OSF)		
One Shot Falling with Input (OSFI)		
One Shot Rising (OSR)		
One Shot Rising with Input (OSRI)		
Output Energize (OTE)		
Output Latch (OTL)		
Output Unlatch (OTU)		
Proportional Integral Derivative (PID)		
Radian (RAD)		
Real to String (RTOS)		
Reset (RES)		
Reset SFC (SFR)		
Return (RET)		
Retentive Timer On (RTO)		
Retentive Timer On with Reset (RTOR)		
Pause SFC (SFP)		
Size In Elements (SIZE)		
Sequencer Input (SQI)		
Sequencer Load (SQL)		
Sequencer Output (SQO)		
Sine (SIN)		
Square Roost (SQR/SQRT)		
String Concatenate (CONCAT)		
String Delete (DELETE)		
String to DINT (STOD)		
String to REAL (STOR)		
Swap Byte (SWPB)		
Subtract (SUB)		
Tangent (TAN)		
Timer Off Delay (TOF)		
Timer Off Delay with Reset (TOFR)		
Timer On Delay (TON)		

Logix5000 Controllers General Instructions Reference Manual 1756-RM003	Logix5000 Controllers Advanced Process Control and Drives and Equipment Phase and Sequence Instructions Reference Manual 1756-RM006	Logix5000 Controllers Motion Instructions Reference Manual MOTION-RM002
Timer On Delay with Reset (TONR)		
Temporary End (TND)		
Tracepoints (TPT)		
Trigger Event Task (EVENT)		
Truncate (TRN)		
Unknown Instruction (UNK)		
Upper Case (UPPER)		
User Interrupt Disable (UID)/User Interrupt Enable (UIE)		
X to the Power of Y (XPY)		
Examine if Closed (XIC)		
Examine If Open (XIO)		
Bitwise Exclusive (XOR)		

Summary of changes
Instruction Locator
Preface
Studio 5000 environment 15
Additional resources 15
Purpose of this manual 16
Legal Notices 16

Chapter 1

Process Control Instructions
Process Control Instructions 17
Alarm (ALM) 18
Discrete 3-State Device (D3SD) 24
Discrete 2-State Device (D2SD) 38
Deadtime (DEDT) 48
Function Generator (FGEN) 54
Lead-Lag (LDLG) 60
Enhanced PID (PIDE) 65
Position Proportional (POSP) 99
Ramp/Soak (RMPS) 107
Scale (SCL) 122
Split Range Time Proportional (SRTP) 126
Totalizer (TOT) 134
Coordinated Control (CC) 143
- CC Function Block Configuration 180
- CC Function Block Model Initialization 181
- CC Function Block Tuning 181
- CC Function Block Tuning Errors 182
- CC Function Block Tuning Procedure 182

Internal Model Control (IMC) 182
- IMC Function Block Configuration 200
- IMC Function Block Model Initialization 201
- IMC Function Block Tuning 201
- IMC Function Block Tuning Errors 202
- IMC Function Block Tuning Procedure 202

Modular Multivariable Control (MMC) 203
- MMC Function Block Configuration 241
- MMC Function Block Model Initialization 243
- MMC Function Block Tuning 243
- Use an MMC Function Block for Splitter Control 244
- MMC Function Block Tuning Errors 244
- MMC Function Block Tuning Procedure 244

Current SP 245
- Use the Coordinated Control Function Block to Control 245

CV High/Low Limiting 247

CV Percent Limiting 248
CV Rate-of-Change Limiting 249
CV Windup Limiting 249
Execution 250
Switch Between Program Control and Operator Control 250
Operating Modes 251
Convert the PV and SP Values to Percent 252
Primary Loop Control 252
Processing Faults 254
Select the Control Variable 254
Update the CVOper and CVProg Values 255
Select the Setpoint 255
SP High/Low Limiting 255

Chapter 2

PlantPAx

PlantPAx instructions 257
Process Analog HART (PAH) 259
Process Analog Input (PAI) 274
Process Dual Sensor Analog Input (PAID) 304
Process Multi Sensor Analog Input (PAIM) 321
Process Analog Output (PAO) 344
Process Analog Output feedback processing 377
Process Boolean Logic (PBL) 387
Process Command Source (PCMDSRC) 403
Process Command Source operating model 413
Process Deadband Controller (PDBC) 414
Process Discrete Input (PDI) 432
Process Discrete Output (PDO) 448
Process Dosing (PDOSE) 474
Process Analog Fanout (PFO) 509
Process High or Low Selector (PHLS) 521
Process Interlocks (PINTLK) 531
Process Lead Lag Standby Motor Group (PLLS) 542
Motor Sort Algorithm for Process Lead Lag Standby Motor Group (PLLS) instructions 572
Process Motor (PMTR) 574
Process Motor (PMTR) Command Source 609
Process Permissives (PPERM) 611
Process Proportional + Integral + Derivative (PPID) 620
Process Pressure/Temperature Compensated Flow (PPTC) 707
Process Restart Inhibit (PRI) 715
Process Run Time and Start Counter (PRT) 722

Process Tank Strapping Table (PTST) 729
Process Valve (PVLV) 737
Process Valve (PVLV) Command Source 767
Process Valve Statistics (PVLVS) 769
Process Variable Speed Drive (PVSD) 781
Process Variable Speed Drive (PVSD) Command Source 816

Chapter 3

Drives

Drives Instructions 823
Integrator (INTG) 823
Proportional + Integral (PI) 830
Pulse Multiplier (PMUL) 840
S-Curve (SCRV) 848
Second-Order Controller (SOC) 856
Up/Down Accumulator (UPDN) 865
HMI Button Control (HMIBC) 870

Chapter 4

Filter

Filter Instructions 875
Derivative (DERV) 875
High Pass Filter (HPF) 879
Low Pass Filter (LPF) 885
Notch Filter (NTCH) 890
Second-Order Lead Lag (LDL2) 896

Chapter 5

Select_Limit Instructions

Select/Limit Instructions 903
Enhanced Select (ESEL) 903
High/Low Limit (HLL) 912
Multiplexer (MUX) 916
Rate Limiter (RLIM) 919
Select (SEL) 923
Selected Negate (SNEG) 926
Selected Summer (SSUM) 930

Chapter 6

Statistical Instructions

Statistical Instructions 935
Moving Average (MAVE) 935
Maximum Capture (MAXC) 942
Minimum Capture (MINC) 946
Moving Standard Deviation (MSTD) 949

Chapter 7

Logical and Move

Logical and Move Instructions 955
D Flip-Flop (DFF) 955
JK Flip-Flop (JKFF) 959
Reset Dominant (RESD) 962
Set Dominant (SETD) 965

Chapter 8

Equipment Phase Instructions

Equipment Phase Instructions 969
Attach to Equipment Phase (PATT) 970
Detach from Equipment Phase (PDET) 975
Equipment Phase Clear Failure (PCLF) 978
Equipment Phase Command (PCMD) 980
Equipment Phase External Request (PXRQ) 987
Equipment Phase Failure (PFL) 997
Equipment Phase New Parameters (PRNP) 1001
Equipment Phase Override Command (POVR) 1004
Equipment Phase Paused (PPD) 1008
Phase State Complete (PSC) 1012

Chapter 9

Equipment Sequence

Equipment Sequence instructions 1019
Attach to Equipment Sequence (SATT) 1019
Detach from Equipment Sequence (SDET) 1021
Equipment Sequence Assign Sequence Identifier (SASI) 1023
Equipment Sequence Clear Failure (SCLF) 1025
Equipment Sequence command (SCMD) 1027
Equipment Sequence Diagram instructions 1031
Equipment Sequence Override (SOVR) 1031
Guidelines for SATT instructions 1033
Guidelines for SCMD instructions 1034
Guidelines for SOVR instructions 1035
Result codes for SATT instructions 1036
Result codes for SCLF instructions 1036
Result codes for SCMD instructions 1037
Result codes for SOVR instructions 1038
SASI instruction examples 1038
SATT instruction examples 1039
SCLF instruction examples 1040
SCMD instruction examples 1040
SDET instruction examples 1041
SOVR instruction examples 1041

When should I use an SOVR instruction instead of an SCMD instruction? 1042

Chapter 10

Function Block Attributes

Choose the Function Block Elements 1043
Latching Data 1044
Function Block Responses to Overflow Conditions 1045
Order of Execution 1046
Timing Modes 1050
Program/Operator Control 1052
Function Block States 1055

Chapter 11

Structured Text Programming

Structured Text Syntax 1057
Structured Text Components: Comments 1058
Structured Text Components: Assignments 1059
- Specify a non-retentive assignment 1060
- Assign an ASCII character to a string data member 1061

Structured Text Components: Expressions 1061
- Use arithmetic operators and functions 1063
- Use bitwise operators 1064
- Use logical operators 1064
- Use relational operators 1065

Structured Text Components: Instructions 1066
Structured Text Components: Constructs 1067
Character string literals 1068
- String Types 1069

CASE_OF 1069
FOR_DO 1071
IF_THEN 1074
REPEAT_UNTIL 1077
WHILE_DO 1080
Structured Text Attributes 1082

Chapter 12

Common Attributes for Advanced Process Control and Drives Instructions

Common Attributes 1083
Math Status Flags 1083
Immediate values 1085
Data Conversions 1086
Elementary data types 1089
Floating Point Values 1092
Index Through Arrays 1094

Bit Addressing 1094
Function Block Faceplate Controls 1095
Faceplate Control Properties Dialog - General Tab 1097
Faceplate Control Properties Dialog - Display Tab 1097
Faceplate Control Properties Dialog - Font Tab 1098
Faceplate Control Properties Dialog - LocaleTab 1100
ASCII Character Codes 1101
ASCII character codes 1101

Index

This manual provides a programmer with details about the available General, Motion, Process, and Drives instruction set for a Logix-based controller.

If you design, program, or troubleshoot safety applications that use GuardLogix controllers, refer to the GuardLogix Safety Application Instruction Set Safety Reference Manual, publication 1756-RM095.

This manual is one of a set of related manuals that show common procedures for programming and operating Logix 5000 controllers.

For a complete list of common procedures manuals, refer to the Logix 5000 Controllers Common Procedures Programming Manual, publication 1756-PM001.

The term Logix 5000 controller refers to any controller based on the Logix 5000 operating system.

Studio 5000 environment

The Studio 5000 Automation Engineering & Design Environment® combines engineering and design elements into a common environment. The first element is the Studio 5000 Logix Designer® application. The Logix Designer application is the rebranding of RSLogix 5000® software and will continue to be the product to program Logix 5000™ controllers for discrete, process, batch, motion, safety, and drive-based solutions.

The Studio 5000® environment is the foundation for the future of Rockwell Automation® engineering design tools and capabilities. The Studio 5000 environment is the one place for design engineers to develop all elements of their control system.

Additional resources

These documents contain additional information concerning related Rockwell Automation products.

Resource	Description
Industrial Automation Wiring and Grounding Guidelines, publication 1770-4.1	Provides general guidelines for installing a Rockwell Automation industrial system.
Product Certifications webpage, available at http://ab.rockwellautomation.com	Provides declarations of conformity, certificates, and other certification details.

View or download publications at http://www.rockwellautomation.com/literature. To order paper copies of technical documentation, contact the local Rockwell Automation distributor or sales representative.

Purpose of this manual

This manual provides a programmer with details about each available instruction for a Logix-based controller. This manual also gives you guidance and examples to use equipment phase instructions to transition to different state, handle faults, set up break points, and so forth.

Legal Notices

Rockwell Automation publishes legal notices, such as privacy policies, license agreements, trademark disclosures, and other terms and conditions on the Legal Notices page of the Rockwell Automation website.

End User License Agreement (EULA)

You can view the Rockwell Automation End-User License Agreement ("EULA") by opening the License.rtf file located in your product's install folder on your hard drive.

Open Source Licenses

The software included in this product contains copyrighted software that is licensed under one or more open source licenses. Copies of those licenses are included with the software. Corresponding Source code for open source packages included in this product are located at their respective web site(s).

Alternately, obtain complete Corresponding Source code by contacting Rockwell Automation via the Contact form on the Rockwell Automation website: http://www.rockwellautomation.com/global/about-us/contact/contact.page
Please include "Open Source" as part of the request text.

A full list of all open source software used in this product and their corresponding licenses can be found in the OPENSOURCE folder. The default installed location of these licenses is `C:\Program Files (x86)\Common Files\Rockwell\Help\FactoryTalk Services Platform\Release Notes\OPENSOURCE\index.htm`.

Process Control Instructions

Process Control Instructions

The Process Control instructions include these instructions:

Available Instructions

Ladder Diagram

Not available

Function Block and Structured Text

ALM	SCL	PIDE	RMPS	POSP	SRTP	LDLG	FGEN

TOT	DEDT	D2SD	D3SD	IMC	CC	MMC

If you want to	Use this instruction
Provide alarming for any analog signal.	ALM
Control discrete devices, such as solenoid valves, pumps, and motors, that have only two possible states (e.g., on/off, open/closed, etc.).	D2SD
Control discrete devices, such as high/low/off feeders that have three possible states (e.g., fast/slow/off, forward/stop/reverse, etc.).	D3SD
Perform a delay of a single input. You select the amount of deadtime delay.	DEDT
Convert an input based on a piece-wise linear function.	FGEN
Provide a phase lead-lag compensation for an input signal.	LDLG
Regulate an analog output to maintain a process variable at a certain setpoint, using a PID algorithm.	PIDE
Raise/lower or open/close a device, such as a motor-operated valve, by pulsing open or close contacts.	POSP
Provide for alternating ramp and soak periods to follow a temperature profile.	RMPS
Convert an unscaled input value to a floating point value in engineering units.	SCL
Take the 0-100% output of a PID loop and drive heating and cooling digital output contacts with a periodic pulse.	SRTP

Provide a time-scaled accumulation of an analog input value, such as a volumetric flow.	TOT
Control a single process variable by maintaining a single controller output.	IMC
Control a single process variable by manipulating as many as three different control variables.	CC
Control two process variables to their setpoints using up to three control variables.	MMC

See also

Filter Instructions on page 875

Logical and Move Instructions on page 955

Drives Instructions on page 823

Select/Limit Instructions on page 903

Statistical Instructions on page 935

Alarm (ALM)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The ALM instruction provides alarming for any analog signal.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block

Structured Text

ALM(ALM_tag)

Operands

Function Block

Operand	Type	Format	Description
ALM tag	ALARM	structure	ALM structure

Structured Text

Operand	Type	Format	Description
ALM tag	ALARM	structure	ALM structure

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

ALARM Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
In	REAL	The analog signal input. Valid = any float Default = 0.0
HHLimit	REAL	The high-high alarm limit for the input. Valid = any real value Default = maximum positive value

Input Parameter	Data Type	Description
HLimit	REAL	The high alarm limit for the input. Valid = any real value Default = maximum positive value
LLimit	REAL	The low alarm limit for the input. Valid = any real value Default = maximum negative value
LLLimit	REAL	The low-low alarm limit for the input. Valid = any real value Default = maximum negative value
≤ Deadband	REAL	The alarm deadband for the high-high to low-low limits Valid = any real value ≥ 0.0 Default = 0.0
ROCPosLimit	REAL	The rate-of-change alarm limit in units per second for a positive (increasing) change in the input. Set ROCPosLimit = 0 to disable ROC positive alarming. If invalid, the instruction assumes a value of 0.0 and sets the appropriate bit in Status. Valid = any real value ≤ 0.0 Default = 0.0
ROCNegLimit	REAL	The rate-of-change alarm limit in units per second for a negative (decreasing) change in the input. Set ROCNegLimit = 0 to disable ROC negative alarming. If invalid, the instruction assumes a value of 0.0 and sets the appropriate bit in Status. Valid = any real value ≤ 0.0 Default = 0.0
ROCPeriod	REAL	Time period in seconds for calculation (sampling interval) of the rate of change value. Each time the sampling interval expires, a new sample of In is stored, and ROC is re-calculated. Instead of an enable bit like other conditions in the analog alarm, the rate-of-change detection is enabled by putting any non-zero value in the ROCPeriod. Valid = 0.0 to 32767.0 Default = 0.0.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if ROC overflows.
HHAlarm	BOOL	The high-high alarm indicator. Default = false
HAlarm	BOOL	The high alarm indicator. Default = false
LAlarm	BOOL	The low alarm indicator. Default = false
LLAlarm	BOOL	The low-low alarm indicator. Default = false
ROCPosAlarm	BOOL	The rate-of-change positive alarm indicator. Default = false
ROCNegAlarm	BOOL	The rate-of-change negative alarm indicator. Default = false

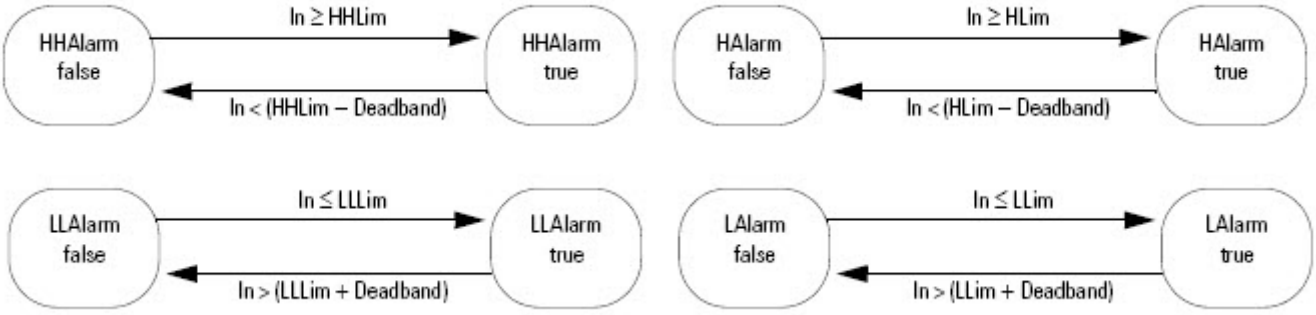
Output Parameter	Data Type	Description
ROC	REAL	The rate-of-change output.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
DeadbandInv (Status.1)	BOOL	Invalid Deadband value.
ROCPosLimitInv (Status.2)	BOOL	Invalid ROCPosLimit value.
ROCNegLimitInv (Status.3)	BOOL	Invalid ROCNegLimit value.
ROCPeriodInv (Status.4)	BOOL	Invalid ROCPeriod value.

Description

The ALM instruction provides alarm indicators for high-high, high, low, low-low, rate-of-change positive, and rate-of-change negative. An alarm deadband is available for the high-high to low-low alarms. A user-defined period for performing rate-of-change alarming is also available.

High-high to Low-low Alarm

The high-high and low-low alarm algorithms compare the input to the alarm limit and the alarm limit plus or minus the deadband.



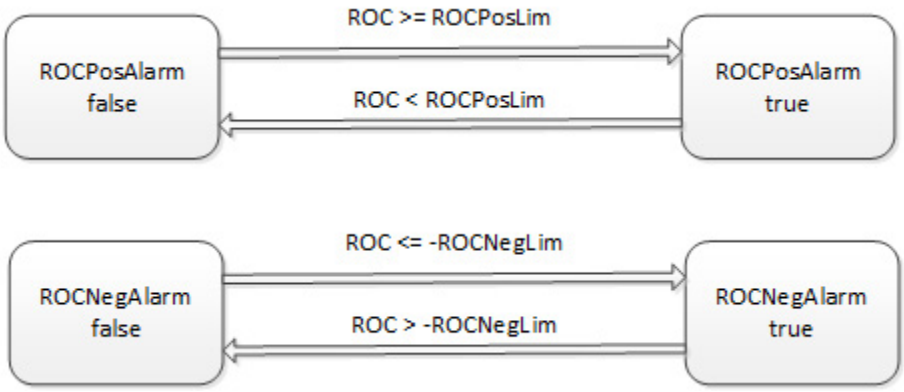
Rate-of-change Alarm

The rate-of-change (ROC) alarm compares the change of the input over the ROCPeriod to the rate-of-change limits. The ROCPeriod provides a type of deadband for the rate-of-change alarm. For example, define an ROC alarm limit of 2°F/second with a period of execution of 100 ms. If you use an analog input module with a resolution of 1°F, every time the input value changes, an ROC alarm is generated because the instruction calculates an effective rate of 10°F/second. However, enter an ROCPeriod of 1 sec and the instruction only generates an alarm if the rate truly exceeds the 2°F/second limit.

The ROC alarm calculates the rate-of-change as:

$$ROC = \frac{In(Now) - In(EndofpreviousROCPeriod)}{ROCPeriod}$$

The instruction performs this calculation when the ROCPeriod expires. Once the instruction calculates the ROC, it determines alarms as:



Monitoring the ALM Instruction

There is an operator faceplate available for the ALM instruction.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	Rung-condition-in bits are cleared to false.
Rung-condition-in is false	Rung-condition-in bits are cleared to false.
Rung-condition-in is true	Rung-condition-in bits are set to true. The instruction executes.
Postscan	Rung-condition-in bits are cleared to false.

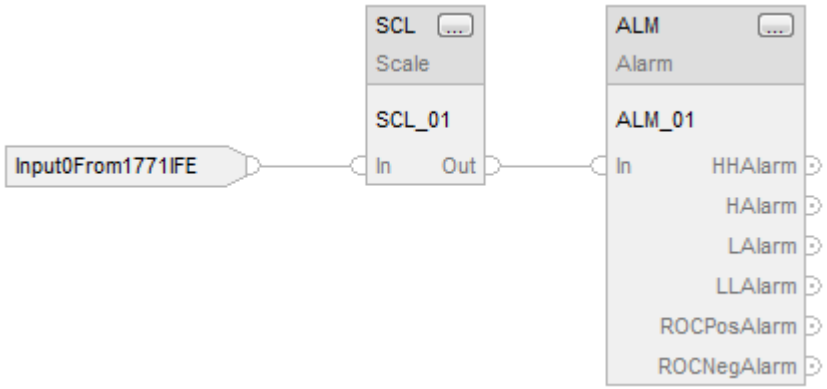
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Rung-condition-in is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

The ALM instruction is typically used either with analog input modules (such as 1771 I/O modules) that do not support on-board alarming, or to generate alarms on a calculated variable. In this example, an analog input from a 1771-IFE module is first scaled to engineering units using the SCL instruction. The Out of the SCL instruction is an input to the ALM instruction to determine whether to set an alarm. The resulting alarm output parameters could then be used in your program and/or viewed on an operator interface display.

Function Block



Structured Text

```
SCL_01.IN := Input0From1771IFE;

SCL(SCL_01);

ALM_01.IN := SCL_01.Out;

ALM(ALM_01);
```

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Discrete 3-State Device (D3SD)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The D3SD instruction controls a discrete device having three possible states, such as fast/slow/off or forward/stop/reverse.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block

Structured Text

D3SD(D3SD_tag)

Operands

Structured Text

Operand	Type	Format	Description
D3SD tag	DISCRETE_3STATE	structure	D3SD structure

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

Function Block

Operand	Type	Format	Description
D3SD tag	DISCRETE_3STATE	structure	D3SD structure

DISCRETE_3STATE Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
Prog0Command	BOOL	Program state 0 command. This input determines the device state when the device is in Program control. If true, the device is commanded to the 0 state. Default is false.
Prog1Command	BOOL	Program state 1 command. This input determines the device state when the device is in Program control. If true, the device is commanded to the 1 state. Default is false.
Prog2Command	BOOL	Program state 2 command. This input determines the device state when the device is in Program control. If true, the device is commanded to the 2 state. Default is false.
Oper0Req	BOOL	Operator state 0 request. Set to true by the operator interface to place the device into the 0 state when the device is in Operator control. Default is false.
Oper1Req	BOOL	Operator state 1 request. Set true by the operator interface to place the device into the 1 state when the device is in Operator control. Default is false.

Input Parameter	Data Type	Description
Oper2Req	BOOL	Operator state 2 request. Set to true by the operator interface to place the device into the 2 state when the device is in Operator control. Default is false.
State0Perm	BOOL	State 0 permissive. Unless in Hand or Override mode, this input must be true for the device to enter the 0 state. This input has no effect if the device is already in the 0 state. Default is true.
State1Perm	BOOL	State 1 permissive. Unless in Hand or Override mode, this input must be true for the device to enter the 1 state. This input has no effect if the device is already in the 1 state. Default is true.
State2Perm	BOOL	State 2 permissive. Unless in Hand or Override mode, this input must be true for the device to enter the 2 state. This input has no effect if the device is already in the 2 state. Default is true.
FB0	BOOL	The first feedback input available to the instruction. Default is false.
FB1	BOOL	The second feedback input available to the instruction. Default is false.
FB2	BOOL	The third feedback input available to the instruction. Default is false.
FB3	BOOL	The fourth feedback input available to the instruction. Default is false.
HandFB0	BOOL	Hand feedback state 0. This input from a field hand/off/auto station shows the requested state of the field device. True indicates the field device is being requested to enter the 0 state; false indicates the field device is being requested to enter some other state. Default is false.
HandFB1	BOOL	Hand feedback state 1. This input from a field hand/off/auto station shows the requested state of the field device. True indicates the field device is being requested to enter the 1 state; false indicates the field device is being requested to enter some other state. Default is false.

Input Parameter	Data Type	Description
HandFB2	BOOL	Hand feedback state 2. This input from a field hand/off/auto station shows the requested state of the field device. True indicates the field device is being requested to enter the 2 state; false indicates the field device is being requested to enter some other state. Default is false.
FaultTime	REAL	Fault time value. Configure the value in seconds of the time to allow the device to reach a newly commanded state. Set FaultTime = 0 to disable the fault timer. If this value is invalid, the instruction assumes a value of zero and sets the appropriate bit in Status. Valid = any float ≥ 0.0 Default = 0.0
FaultAlarmLatch	BOOL	Fault alarm latch input. When true and FaultAlarm is true, latch FaultAlarm. To unlatch FaultAlarm, set FaultAlmUnlatch to true or clear FaultAlarmLatch to false. Default is false.
FaultAlmUnLatch	BOOL	Fault alarm unlatch input. Set this input to true when FaultAlarmLatch is set to unlatch FaultAlarm. The instruction clears this input to false. Default is false.
OverrideOnInit	BOOL	Override on initialization request. If this bit is true, then during instruction first scan, the instruction is placed in Operator control with Override true and Hand false. If ProgHandReq is true, then Override is cleared to false and Hand is set to true. Default is false.
OverrideOnFault	BOOL	Override on fault request. Set this value to true if the device should go to Override mode and enter the Override State on a fault alarm. After the fault alarm is removed, the instruction is placed in Operator control. Default is false.
Out0State0	BOOL	Output 0 state 0 input. This value determines the value of Output0 when the device is in the 0 state. Default is false.
Out0State1	BOOL	Output 0 state 1 input. This value determines the value of Output0 when the device is in the 1 state. Default is false.

Input Parameter	Data Type	Description
Out0State2	BOOL	Output 0 state 2 input. This value determines the value of Output0 when the device is in the 2 state. Default is false.
Out1State0	BOOL	Output 1 state 0 input. This value determines the value of Output1 when the device is in the 0 state. Default is false.
Out1State1	BOOL	Output 1 state 1 input. This value determines the value of Output1 when the device is in the 1 state. Default is false.
Out1State2	BOOL	Output 1 state 2 input. This value determines the value of Output1 when the device is in the 2 state. Default is false.
Out2State0	BOOL	Output 2 state 0 input. This value determines the value of Output2 when the device is in the 0 state. Default is false.
Out2State1	BOOL	Output 2 state 1 input. This value determines the value of Output2 when the device is in the 1 state. Default is false.
Out2State2	BOOL	Output 2 state 2 input. This value determines the value of Output2 when the device is in the 2 state. Default is false.
OverrideState	DINT	Override state input. Set this input to indicate the state of the device when in Override mode. 2 = Device should go to the 2 state 1 = Device should go to the 1 state 0 = Device should go to the 0 state An invalid value sets the appropriate bit in Status. Valid = 0 to 2 Default = 0
FB0State0	BOOL	Feedback 0 state 0 input. This value determines the expected value of FB0 when the device is in the 0 state. Default is false.
FB0State1	BOOL	Feedback 0 state 1 input. This value determines the expected value of FB0 when the device is in the 1 state. Default is false.
FB0State2	BOOL	Feedback 0 state 2 input. This value determines the expected value of FB0 when the device is in the 2 state. Default is false.
FB1State0	BOOL	Feedback 1 state 0 input. This value determines the expected value of FB1 when the device is in the 0 state. Default is false.

Input Parameter	Data Type	Description
FB1State1	BOOL	Feedback 1 state 1 input. This value determines the expected value of FB1 when the device is in the 1 state. Default is false.
FB1State2	BOOL	Feedback 1 state 2 input. This value determines the expected value of FB1 when the device is in the 2 state. Default is false.
FB2State0	BOOL	Feedback 2 state 0 input. This value determines the expected value of FB2 when the device is in the 0 state. Default is false.
FB2State1	BOOL	Feedback 2 state 1 input. This value determines the expected value of FB2 when the device is in the 1 state. Default is false.
FB2State2	BOOL	Feedback 2 state 2 input. This value determines the expected value of FB2 when the device is in the 2 state. Default is false.
FB3State0	BOOL	Feedback 3 state 0 input. This value determines the expected value of FB3 when the device is in the 0 state. Default is false.
FB3State1	BOOL	Feedback 3 state 1 input. This value determines the expected value of FB3 when the device is in the 1 state. Default is false.
FB3State2	BOOL	Feedback 3 state 2 input. This value determines the expected value of FB3 when the device is in the 2 state. Default is false.
ProgProgReq	BOOL	Program program request. Set to true by the user program to request Program control. Ignored if ProgOperReq is true. Holding this true and ProgOperReq false locks the instruction in Program control. Default is false.
ProgOperReq	BOOL	Program operator request. Set to true by the user program to request operator control. Holding this true locks the instruction in Operator control. Default is false.
ProgOverrideReq	BOOL	Program override request. Set to true by the user program to request the device to enter Override mode. Ignored if ProgHandReq is true. Default is false.
ProgHandReq	BOOL	Program hand request. Set to true by the user program to request the device to enter Hand mode. Default is false.

Input Parameter	Data Type	Description
OperProgReq	BOOL	Operator program request. Set to true by the operator interface to request Program control. The instruction clears this input to false. Default is false.
OperOperReq	BOOL	Operator operator request. Set to true by the operator interface to request Operator control. The instruction clears this input to false. Default is false.
ProgValueReset	BOOL	Reset program control values. When true, all the program request inputs are cleared to false at each execution of the instruction. Default is false.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled.
Out0	BOOL	The first output of the instruction.
Out1	BOOL	The second output of the instruction.
Out2	BOOL	The third output of the instruction.
Device0State	BOOL	Device state 0 output. True when the device is commanded to the 0 state and the feedback indicates the device really is in the 0 state.
Device1State	BOOL	Device state 1 output. True when the device is commanded to the 1 state and the feedback indicates the device really is in the 1 state.
Device2State	BOOL	Device state 2 output. True when the device is commanded to the 2 state and the feedback indicates the device really is in the 2 state.
Command0Status	BOOL	Device state 0 command status. True when the device is being commanded to the 0 state; false when the device is being commanded to some other state.
Command1Status	BOOL	Device state 1 command status. True when the device is being commanded to the 1 state; false when the device is being commanded to some other state.
Command2Status	BOOL	Device state 2 command status. True when the device is being commanded to the 2 state; false when the device is being commanded to some other state.

Output Parameter	Data Type	Description
FaultAlarm	BOOL	Fault alarm output. True if the device has been commanded to a new state, and the FaultTime has expired without the feedback indicating that the new state has actually been reached. Also set to true if, after reaching a commanded state, the feedbacks suddenly indicate that the device is no longer in the commanded state.
ModeAlarm	BOOL	Mode alarm output. True if the device is in operator control and a ProgxCommand input requests a state which is different from the state currently commanded by the operator. This alarm is intended as a reminder that a device was left in Operator control.
ProgOper	BOOL	Program/operator control indicator. True when in Program control. False when in Operator control.
Override	BOOL	Override mode. True when the device is in the Override mode.
Hand	BOOL	Hand mode. True when the device is in the Hand mode.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
FaultTimeInv (Status.1)	BOOL	Invalid FaultTime value. The instruction sets FaultTime = 0.
OverrideStateInv (Status.2)	BOOL	The Override value is out of range. It prevents the instruction from entering the Override state.
ProgCommandInv (Status.3)	BOOL	Multiple program state command bits are set at the same time. Refer to Commanded State in Program Control section.
OperReqInv (Status.4)	BOOL	Multiple operator state request bits are set at the same time. Refer to Commanded State in Program Control section.
HandCommandInv (Status.5)	BOOL	Multiple hand feedback state request bits are set at the same time.

Description

The D3SD instruction controls a discrete device having three possible states, such as fast/slow/off or forward/stop/reverse. Typical discrete devices of this nature include feeder systems, and reversible motors.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes
Instruction first run	Set ProgOper to Operator Mode. Set Command0Status to True. Set Command1Status to False. Set Command2Status to False.
Instruction first scan	The fault timer is cleared. ModeAlarm is cleared to false. All the operator request inputs are cleared to false. If ProgValueReset is true, all the program request inputs are cleared to false. When OverrideOnInit is true, ProgOper is cleared to false(Operator control). If ProgHandReq is false and OverrideOnInit is true, Hand is cleared to false and Override is set to true (Override mode). If ProgHandReq is true, Hand is set to true and Override is cleared to false(Hand mode).
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

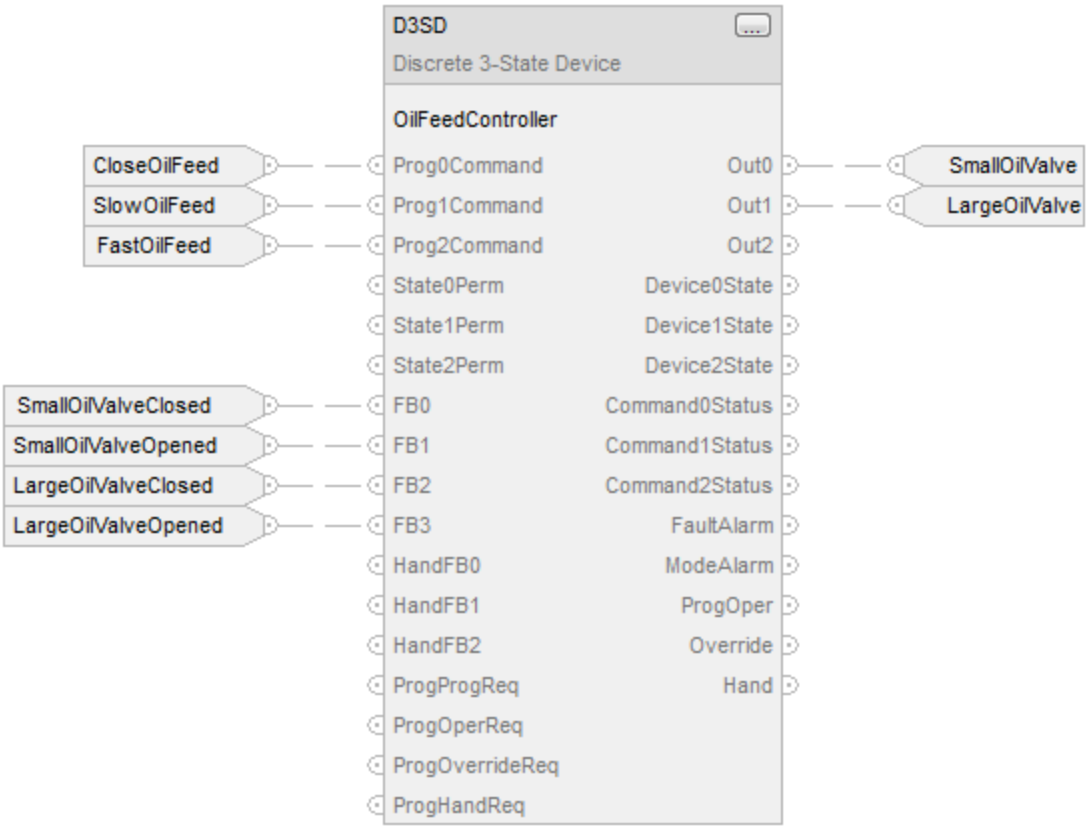
In Structured Text, EnableIn is always true during normal scan. Therefore, if the instruction is in the control path activated by the logic it will execute.

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Examples

The D3SD instruction is typically used to control 3-state devices such as high/low/off feed systems. In this example, the D3SD instruction controls a feed system consisting of a pair of solenoid valves adding vegetable oil to a batch tank. One of the valves is on a large diameter feed pipe into the batch tank, and the other valve is plumbed in parallel on a small diameter feed pipe. When oil is first added, the D3SD instruction is commanded to the fast feed state (state 2) where both valves are opened. When the oil added approaches the target amount, the D3SD instruction is commanded to the slow feed state (state 1) where the "large valve" is closed and the "small valve" is kept open. When the target is reached, the D3SD instruction is commanded to go to the off state (state 0) and both valves are closed. As long as the D3SD instruction is in Program control, the valves open according to the CloseOilFeed, SlowOilFeed, and FastOilFeed inputs. The operator can also take Operator control of the feed system if necessary. The solenoid valves in this example have limit switches which indicate when the valves are fully closed or opened. These switches are wired into the FB0, FB1, FB2, and FB3 feedback inputs. This allows the D3SD instruction to generate a FaultAlarm if the solenoid valves do not reach their commanded states within the configured FaultTime.

Function Block

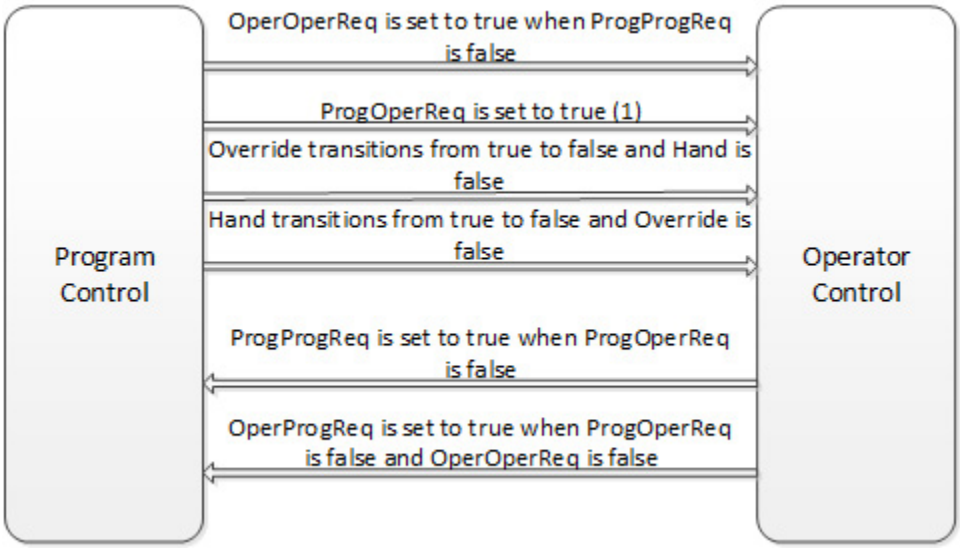


Structured Text

```
OilFeedController.Prog0Command := ClosedOilFeed;
OilFeedController.Prog1Command := SlowOilFeed;
OilFeedController.Prog1Command := FastOilFeed;
OilFeedController.FB0 := SmallOilValveClosed;
OilFeedController.FB1 := SmallOilValveOpened;
OilFeedController.FB2 := LargeOilValveClosed;
OilFeedController.FB3 := LargeOilValveOpened;
D3SD(OilFeedController);
SmallOilValve := OilFeedController.Out0;
LargeOilValve := OilFeedController.Out1;
```

Switch Between Program Control and Operator Control

The following diagram shows how the D3SD instruction changes between Program control and Operator control.



(1) The instruction remains in Operator control mode when ProgOperReq is true.

Commanded State in Program Control

The following table shows how the D3SD instruction operates when in Program control.

Prog0 Command	Prog1 Command	Prog2 Command	State0 Perm	State1 Perm	State2 Perm	Description
false	false	true	either	either	true	Command0Status is cleared to false Command1Status is cleared to false Command2Status is set to true
false	true	false	either	true	either	Command0Status is cleared to false Command1Status is set to true Command2Status is cleared to false
true	false	false	true	either	either	Command0Status is set to true Command1Status is cleared to false Command2Status is cleared to false

If more than one program command input is true:

- The instruction sets the appropriate bit in Status
- If Override and Hand are cleared to false, the instruction holds the previous state

Commanded State in Operator Control

The following table shows how the D3SD instruction operates when in Operator control.

Oper0Req	Oper1Req	Oper2Req	State0 Perm	State1 Perm	State2 Perm	Description
false	false	true	either	either	true	Command0Status is cleared to false Command1Status is cleared to false Command2Status is set to true
false	true	false	either	true	either	Command0Status is cleared to false Command1Status is set to true Command2Status is cleared to false
true	false	false	true	either	either	Command0Status is set to true Command1Status is cleared to false Command2Status is cleared to false

If more than one operator command input is true:

- The instruction sets the appropriate bit in Status
- If Override and Hand are cleared to false, the instruction holds the previous state

After every instruction execution, the instruction:

- Clears all the operator request inputs
- If ProgValueReset is true, clears all the program request inputs to false

Hand Mode or Override Mode

The following table describes how the D3SD instruction determines whether to operate in Hand or Override mode.

ProgHandReq	ProgOverrideReq	FaultAlarm and OverrideOnFault	Description
true	either	either	Hand mode Hand is set to true Override is cleared to false
false	true	either	Override mode Hand is cleared to false Override is set to true
false	either	true	Override mode Hand is cleared to false Override is set to true

When Override is set, it takes precedence over Program and Operator control. The following table describes how the Override mode affects the commanded state.

Override	Override State	Description
true	2	Command0Status is cleared to false Command1Status is cleared to false Command2Status is set to true
true	1	Command0Status is cleared to false Command1Status is set to true Command2Status is cleared to false
true	0	Command0Status is set to true Command1Status is cleared to false Command2Status is cleared to false

If OverrideState is invalid, the instruction sets the appropriate bit in Status and does not enter the override state.

When Hand is true, it takes precedence over Program and Operator control. The following table describes how the Hand mode affects the commanded state.

Hand	HandFB0	HandFB1	HandFB2	Description
true	false	false	true	Command0Status is cleared to false Command1Status is cleared to false Command2Status is set to true
true	false	true	false	Command0Status is cleared to false Command1Status is set to true Command2Status is cleared to false
true	true	false	false	Command0Status is set to true Command1Status is cleared to false Command2Status is cleared to false

If more than one HandFB input is true, the instruction sets the appropriate bit in Status and, if Hand is true, the instruction holds the previous state.

Output State

The D3SD output state is based on the state of the command status.

CommandStatus	Output State
Command0Status is true	Out0 = Out0State0 Out1 = Out1State0 Out2 = Out2State0
Command0Status is true and FB0 = FB0State0 and FB1 = FB1State0 and FB2 = FB2State0 and FB3 = FB3State0	Stop and clear the fault timer. Device0State is set to true
Command1Status is true	Out0 = Out0State1 Out1 = Out1State1 Out2 = Out2State1
Command1Status is true and FB0 = FB0State1 and FB1 = FB1State1 and FB2 = FB2State1 and FB3 = FB3State1	Stop and clear the fault timer. Device1State is set to true
Command2Status is true	Out0 = Out0State2 Out1 = Out1State2 Out2 = Out2State2
Command2Status is true and FB0 = FB0State2 and FB1 = FB1State2 and FB2 = FB2State2 and FB3 = FB3State2	Stop and clear the fault timer. Device2State is set to true

Fault Alarm Conditions

The D3SD instruction checks for these fault alarm conditions.

Fault alarm condition resulting from	Rules
Device state was commanded to change, but the feedback did not indicate that the desired state was actually reached within the FaultTime	Start the fault timer when $Command0Status_n \neq Command0Status_{n-1}$ or $Command1Status_n \neq Command1Status_{n-1}$ or $Command2Status_n \neq Command2Status_{n-1}$ Set FaultAlarm when the fault timer done and FaultTime > 0.0
The device unexpectedly left a state (according to the feedback) without being commanded to	Set FaultAlarm to true when the fault timer is not timing and one of the following conditions is satisfied: Command0Status is true and Device0State is false Command1Status is true and Device1State is false Command2Status is true and Device2State is false

If there is no fault present, FaultAlarm is cleared to false if one of the following conditions is met:

- Command0Status is true and Device0State is true
- Command1Status is true and Device1State is true
- Command2Status is true and Device2State is true
- FaultTime ≤ 0

FaultAlarm cannot be cleared to false when FaultAlarmLatch is true, unless FaultAlmUnlatch is true and no fault is present.

Mode Alarm Conditions

The mode alarm reminds an operator that a device has been left in Operator control. The mode alarm only turns on when, in Operator control mode, the program tries to change the state of the device from the operator's commanded state. The alarm does not turn on if an operator places a device in Operator control mode and changes the state. The D3SD instruction checks for mode alarm conditions, using these rules.

ModeAlarm is	When
true	Prog2Command ≠ $Prog2Command_{n-1}$ **and** Prog2Command ≠ Command2Status **or** Prog1Command ≠ $Prog1Command_{n-1}$ **and** Prog1Command ≠ Command1Status **or** Prog0Command ≠ $Prog1Command_{n-1}$ **and** Prog0Command ≠ Command0Status
false	Prog2Command = Command2Status and Prog1Command = Command1Status and Prog0Command = Command0Status or The device is in Override, Hand, or Program control mode

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Discrete 2-State Device (D2SD)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The D2SD instruction controls a discrete device which has only two possible states (such as on/off or open/closed).

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block

Structured Text

D2SD(D2SD_tag)

Operands

There are data conversion rules for mixed data types within an instruction. See *Data Conversion*.

Structured Text

Operand	Type	Format	Description
D2SD tag	DISCRETE_2STATE	Structure	D2SD structure

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

Function Block

Operand	Type	Format	Description
D2SD tag	DISCRETE_2STATE	Structure	D2SD structure

DISCRETE_2STATE Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.

Input Parameter	Data Type	Description
ProgCommand	BOOL	Used to determine CommandStatus when the device is in Program control. When true, the device is commanded to the 1 state; when false, the device is commanded to the 0 state. Default is false.
Oper0Req	BOOL	Operator state 0 request. Set by the operator interface to place the device in the 0 state when the device is in Operator control. Default is false.
Oper1Req	BOOL	Operator state 1 request. Set by the operator interface to place the device in the 1 state when the device is in Operator control. Default is false.
State0Perm	BOOL	State 0 permissive. Unless in Hand or Override mode, this input must be set for the device to enter the 0 state. This input has no effect for a device already in the 0 state. Default is true.
State1Perm	BOOL	State 1 permissive. Unless in the Hand or Override mode, this input must be set for the device to enter the 1 state. This input has no effect for a device already in the 1 state. Default is true.
FB0	BOOL	The first feedback input available to the D2SD instruction. Default is false.
FB1	BOOL	The second feedback input available to the D2SD instruction. Default is false.
HandFB	BOOL	Hand feedback input. This input is from a field hand/off/auto station and it shows the requested state of the field device. When true, the field device is being requested to enter the 1 state; when false, the field device is being requested to enter the 0 state. Default is false.
FaultTime	REAL	Fault time value. Configure the value in seconds of the time to allow the device to reach a newly commanded state. Set FaultTime = 0 to disable the fault timer. If this value is invalid, the instruction assumes a value of zero and sets the appropriate bit in Status. Valid = any float $\geq$ 0.0 Default = 0.0
FaultAlarmLatch	BOOL	Fault alarm latch input. When true and FaultAlarm is true, latch FaultAlarm. To unlatch FaultAlarm set FaultAlmUnlatch to true or clear FaultAlarmLatch to false. Default is false.
FaultAlmUnLatch	BOOL	Fault alarm unlatch input. Set FaultAlmUnLatch when FaultAlarmLatch is set to unlatch FaultAlarm. The instruction clears this input to false. Default is false.

Input Parameter	Data Type	Description
OverrideOnInit	BOOL	Override on initialization request. If this bit is true, then during instruction first scan, the 2-state device is placed in Operator control, Override is set to true, and Hand is cleared to false. If ProgHandReq is true, then Override is cleared to false and Hand is set to true. Default is false.
OverrideOnFault	BOOL	Override on fault request. Set OverrideOnFault to true if the device should go to Override mode and enter the OverrideState on a fault alarm. After the fault alarm is removed, the 2-state device is placed in Operator control. Default is false.
OutReverse	BOOL	Reverse default out state. The default state of Out is cleared to false when commanded to state 0, and set to true when commanded to state 1. When OutReverse is true, Out is set to true when commanded to state 0, and cleared to false when commanded to state 1. Default is false.
OverrideState	BOOL	Override state input. Configure this value to specify the state of the device when the device is in Override mode. True indicates the device should go to the 1 state; false indicates the device should go to the 0 state. Default is false.
FB0State0	BOOL	Feedback 0 state 0 input. Configure the state of the FB0 when the device is in the 0 state. Default is false.
FB0State1	BOOL	Feedback 0 state 1 input. Configure the state of the FB0 when the device is in the 1 state. Default is false.
FB1State0	BOOL	Feedback 1 state 0 input. Configure the state of the FB1 when the device is in the 0 state. Default is false.
FB1State1	BOOL	Feedback 1 state 1 input. Configure the state of the FB1 when the device is in the 1 state. Default is false.
ProgProgReq	BOOL	Program program request. Set to true by the user program to request Program control. Ignored if ProgOperReq is true. Holding this true and ProgOperReq false locks the instruction into Program control. Default is false.
ProgOperReq	BOOL	Program operator request. Set to true by the user program to request Operator control. Holding this true locks the instruction into Operator control. Default is false.
ProgOverrideReq	BOOL	Program override request. Set to true by the user program to request the device to enter Override mode. Ignored if ProgHandReq is true. Default is false.

Input Parameter	Data Type	Description
ProgHandReq	BOOL	Program hand request. Set to true by the user program to request the device to enter Hand mode. Default is false.
OperProgReq	BOOL	Operator program request. Set to true by the operator interface to request Program control. The instruction clears this input to false. Default is false.
OperOperReq	BOOL	Operator operator request. Set to true by the operator interface to request Operator control. The instruction clears this input to false. Default is false.
ProgValueReset	BOOL	Reset program control values. When true, all the program request inputs are cleared to false at each execution of the instruction. Default is false.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled.
Out	BOOL	The output of the 2-state instruction.
Device0State	BOOL	Device 0 state output. Set to true when the device is commanded to the 0 state and the feedbacks indicate the device really is in the 0 state.
Device1State	BOOL	Device 1 state output. Set to true when the device is commanded to the 1 state and the feedbacks indicate the device really is in the 1 state.
CommandStatus	BOOL	Command status output. Set to true when the device is being commanded to the 1 state and cleared when the device is being commanded to the 0 state.
FaultAlarm	BOOL	Fault alarm output. Set to true if the device was commanded to a new state and the FaultTime has expired without the feedbacks indicating that the new state has actually been reached. Also set to true if, after reaching a commanded state, the feedbacks suddenly indicate that the device is no longer in the commanded state.
ModeAlarm	BOOL	Mode alarm output. Set to true if the device is in Operator control and a program command changes to a state which is different from the state currently commanded by the operator. This alarm is intended as a reminder that a device was left in Operator control.
ProgOper	BOOL	Program/Operator control indicator. True when in Program control. False when in Operator control.
Override	BOOL	Override mode. True when the device is in the Override mode.
Hand	BOOL	Hand mode. True when the device is in the Hand mode.
Status	DINT	Status of the function block.

Output Parameter	Data Type	Description
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
FaultTimeInv (Status.1)	BOOL	Invalid FaultTime value. The instruction sets FaultTime = 0.
OperReqInv (Status.2)	BOOL	Both operator state request bits are true.

Description

The D2SD instruction controls a discrete device which has only two possible states (such as on/off or open/closed). Typical discrete devices of this nature include motors, pumps, and solenoid valves.

Monitoring the D2SD Instruction

There is an operator faceplate available for the D2SD instruction.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	Set ProgOper to Operator Mode. Set CommandStatus to False.

Condition/State	Action Taken
Instruction first scan	Set EnableOut to true. ModeAlarm and operator request inputs are cleared to false, If ProgValueReset is true, all the program request inputs are cleared to false. When OverrideOnInit is true, ProgOper is cleared to false (Operator control). If ProgHandReq is cleared and OverrideOnInit is set, clear Hand and set Override (Override mode). If ProgHandReq is set, set Hand and clear Override (Hand mode).
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

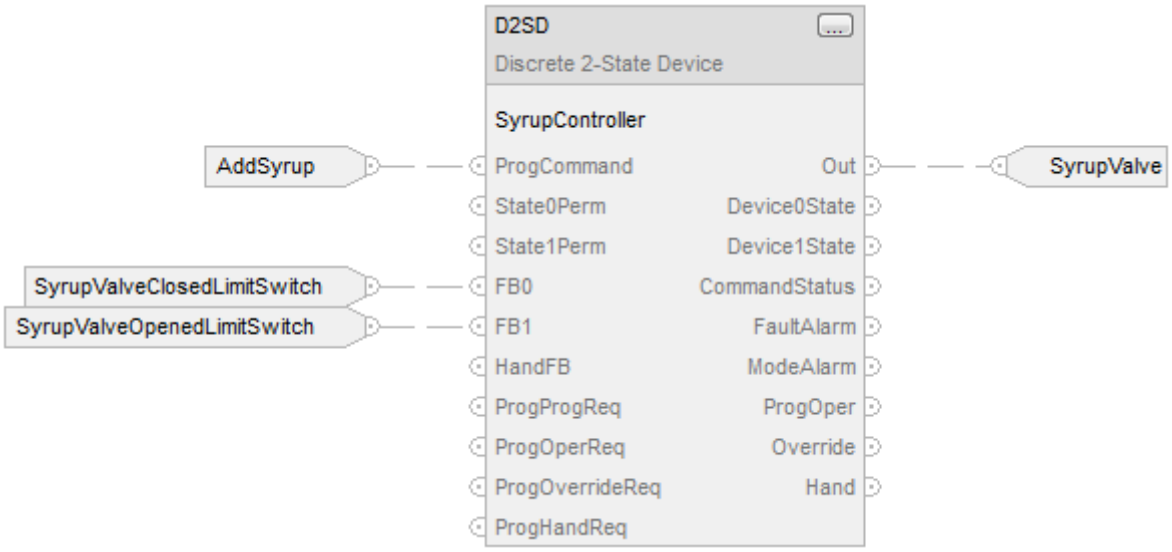
In Structured Text, EnableIn is always true during normal scan. Therefore, if the instruction is in the control path activated by the logic it will execute.

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Examples

SD instruction is typically used to control on-off or open-close devices such as pumps or solenoid valves. In this example, the D2SD instruction controls a solenoid valve adding corn syrup to a batch tank. As long as the D2SD instruction is in Program control, the valve opens when the AddSyrup input is set. The operator can also take Operator control of the valve to open or close it if necessary The solenoid valve in this example has limit switches that indicate when the valve is fully closed or opened. These switches are wired into the FB0 and FB1 feedback inputs. This allows the D2SD instruction to generate a FaultAlarm if the solenoid valve does not reach the commanded state within the configured FaultTime.

Function Block

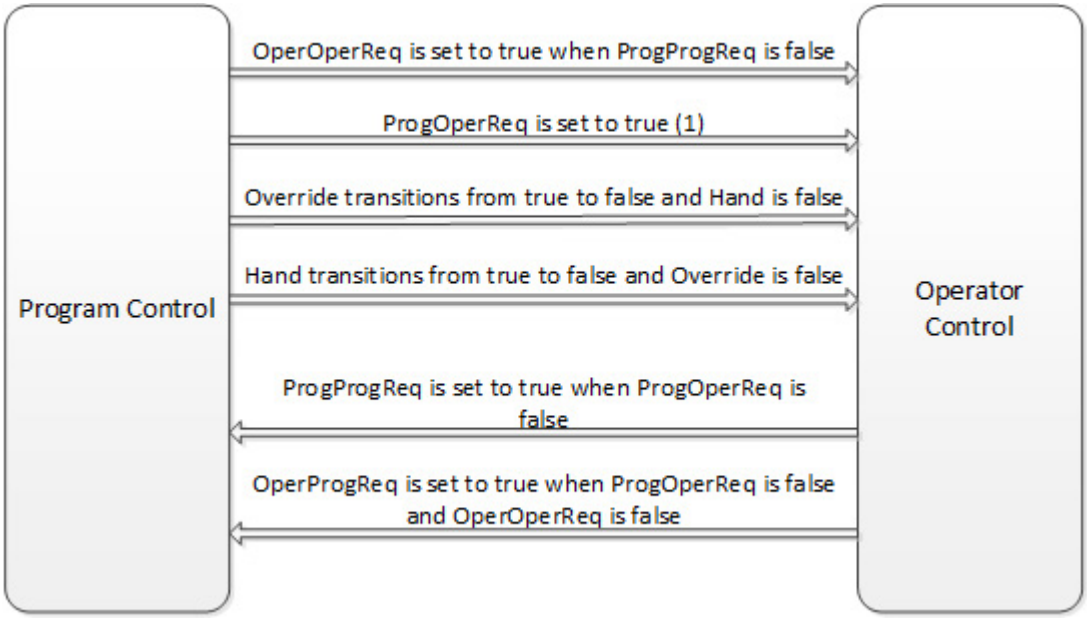


Structured Text

```
SyrupController.ProgCommand := AddSyrup;

SyrupController.FB0 := SyrupValveClosedLimitSwitch;

SyrupController.FB1 := SyrupValveOpenedLimitSwitch;

D2SD(SyrupController);

SyrupValve := SyrupController.Out;
```

Switch Between Program Control and Operator Control

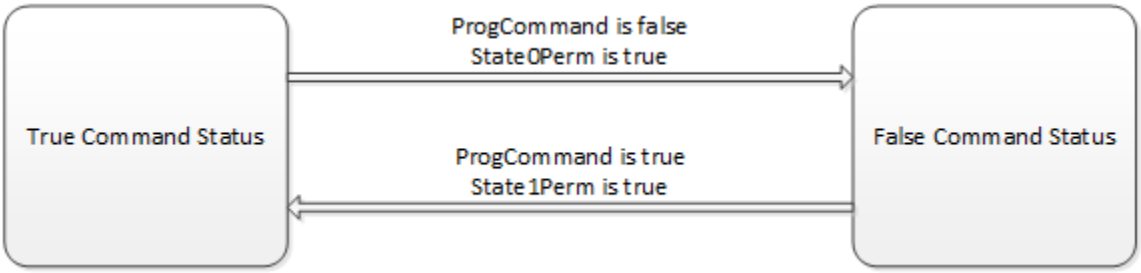
The following diagram shows how the D2SD instruction changes between Program control and Operator control.



(1) The instruction remains in Operator control mode when ProgOperReq is true.

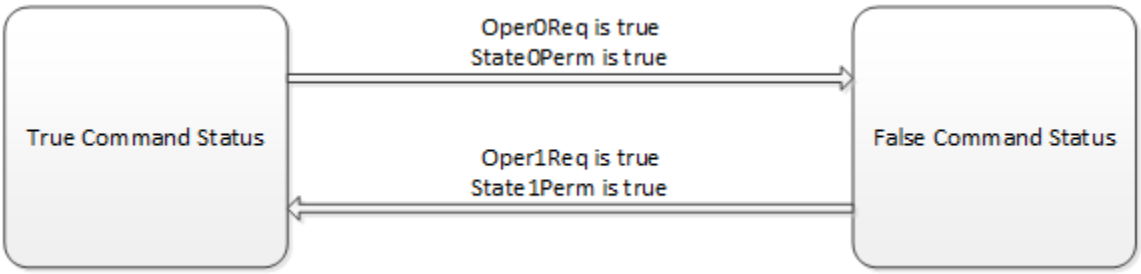
Commanded State in Program Control

The following diagram shows how the D2SD instruction operates when in Program control.



Commanded State in Operator Control

The following diagram shows how the D2SD instruction operates when in Operator control.



If both OperoReq and Oper1Req are true:

- The instruction sets the appropriate bit in Status to true
- If Override and Hand are false, the instruction holds the previous state.

After every instruction execution, the instruction:

- Clears all the operator request inputs to false
- If ProgValueReset is true, clears all the program request inputs to false

Hand Mode or Override Mode

The following table describes how the D2SD instruction determines whether to operate in Hand or Override mode.

ProgHandReq	ProgOverrideReq	FaultAlarm and OverrideOnFault	Description
true	either	either	Hand mode Hand is set to true Override is cleared to false
false	true	either	Override mode Hand is cleared to false Override is set to true
false	either	true	Override mode Hand is cleared to false Override is set to true

When the instruction is in Override mode, CommandStatus = OverrideState.

When the instruction is in Hand mode, CommandStatus = HandFB.

Output State

The D2SD output state is based on the state of the command status.

CommandStatus	Output State
false	If OutReverse is false, Out is cleared to false If OutReverse is true, Out is set to true
true	If OutReverse is false, Out is set to rue If OutReverse is true, Out is cleared to false
false and FB0 = FB0State0 and FB1 = FB1State0	The fault timer is stopped and cleared to 0 Device0State is set to true
true and FB0 = FB0State1 and FB1 = FB1State1	The fault timer is stopped and cleared to 0 Device1State is set to true

Fault Alarm Conditions

The D2SD instruction checks for these fault alarm conditions.

Fault alarm condition resulting from	Rules
Device state was commanded to change, but the feedback did not indicate that the desired state was actually reached within the FaultTime	Start the fault timer when $CommandStatus_n \neq CommandStatus_{n-1}$ Set FaultAlarm when faulttimer is done and FaultTime > 0.0
The device unexpectedly left a state (according to the feedback) without being commanded to	Set FaultAlarm to true when the fault timer is not timing and one of the following conditions is satisfied: CommandStatus is false and Device0State is false CommandStatus is true and Device1State is false

FaultAlarm is cleared to false if one of the following conditions is met:

- CommandStatus is false and Device0State is true
- CommandStatus is true and Device1State is true
- FaultTime $\leq$ 0

FaultAlarm cannot be cleared to false when FaultAlarmLatch is true, unless FaultAlmUnlatch is true and no fault is present.

Mode Alarm Conditions

The mode alarm reminds an operator that a device has been left in Operator control. The mode alarm only turns on when, in Operator control mode, the program tries to change the state of the device from the operator's commanded state. The alarm does not turn on if an operator places a device in Operator control mode and changes the state. The D2SD instruction checks for mode alarm conditions, using these rules.

ModeAlarm	When
True	$ProgCommand_n \neq$ ProgCommandn-1 and $ProgCommand_n \neq$ CommandStatus
False	ProgCommand = CommandStatus or the device is in Override, Hand, or Program control mode

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Data Conversions on page 1086

Deadtime (DEDT)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

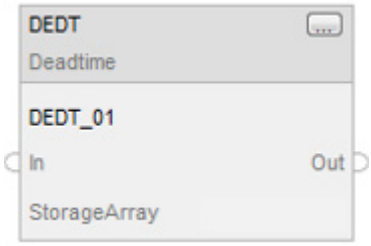
The DEDT instruction performs a delay of a single input. You select the amount of deadtime delay.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

DEDT(DEDT_tag,storage);

Operands

Structured Text

Operand	Type	Format	Description
DEDT tag	DEADTIME	structure	DEDT structure
storage	REAL	array	deadtime buffer

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

Function Block

Operand	Type	Format	Description
DEDT tag	DEADTIME	structure	DEDT structure
storage	REAL	array	deadtime buffer

DEADTIME Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0
InFault	BOOL	Bad health indicator for the input. If the input value is read from an analog input, then InFault is controlled by fault status on the analog input. If true, InFault indicates the input signal has an error, the instruction sets the appropriate bit in Status, the control algorithm is not executed, and Out is held. Default is false. false = good health
Deadtime	REAL	Deadtime input to the instruction. Enter the deadtime in seconds. If this value is invalid, the instruction assumes a value of zero and sets the appropriate bit in Status. Valid = 0.0 to (StorageArray size * DeltaT) Default = 0.0
Gain	REAL	Gain input to the instruction. The value of In is multiplied by this value. This allows simulation of a process gain. Valid = any float Default = 1.0
Bias	REAL	Bias input to the instruction. The value of In multiplied by the Gain is added to this value. This allows simulation of an ambient condition. Valid = any float Default = 0.0
TimingMode	DINT	Selects timing execution mode. 0 = Period mode 1 = oversample mode 2 = real time sampling mode Valid = 0 to 2 Default = 0 For more information about timing modes, see Function Block Attributes.
OversampleDT	REAL	Execution time for oversample mode. Valid = 0 to 4194.303 seconds Default = 0
RTSTime	DINT	Module update period for real time sampling mode Valid = 1 to 32,767ms Default = 1

Input Parameter	Data Type	Description
RTSTimeStamp	DINT	Module time stamp value for real time sampling mode. Valid = 0 to 32,767ms Default = 0

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if Out overflows.
Out	REAL	The calculated output of the deadtime algorithm.
DeltaT	REAL	Elapsed time between updates. This is the elapsed time in seconds used by the control algorithm to calculate the process output.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
InFaulted (Status.1)	BOOL	In health is bad.
DeadtimeInv (Status.2)	BOOL	Invalid Deadtime value.
TimingMode (Status.27)	BOOL	Invalid TimingMode value. For more information about timing modes, see Function Block Attributes.
RTSMissed (Status.28)	BOOL	Only used in real time sampling mode. Set when ABS(DeltaT - RTSTime) > 1 millisecond.
RTSTimeInv (Status.29)	BOOL	Invalid RTSTime value.
RTSTimeStampInv (Status.30)	BOOL	Invalid RTSTimeStamp value.
DeltaTInv (Status.31)	BOOL	Invalid DeltaT value.

Description

The DEDT instruction uses a data buffer to store delayed data, thereby allowing any length deadtime desired. The DEDT instruction is designed to execute in a task where the scan rate remains constant.

To use the DEDT instruction, create a storage array to store the deadtime buffer to hold the samples of (In x Gain) + Bias. The storage array should be large enough to hold the largest desired deadtime, using this formula:

StorageArray Size Needed = Maximum Deadtime (secs) / DeltaT (secs)

Servicing the Deadtime Buffer

During runtime, the instruction checks for a valid Deadtime. Deadtime must be between 0.0 and (StorageArray Size x DeltaT).

If the Deadtime is invalid, the instruction sets an appropriate Status bit and sets Out = (In x Gain) + Bias.

The deadtime buffer functions as a first-in, first-out buffer. Every time the deadtime algorithm executes, the oldest value in the deadtime buffer is moved into Out. The remaining values in the buffer shift downward and the value ((In x Gain) + Bias) is moved to the beginning of the deadtime buffer. A new value that is placed in the deadtime buffer appears in the Out after Deadtime seconds.

The number of array elements required to perform the programmed delay is calculated by dividing Deadtime by DeltaT. If Deadtime is not evenly divisible by DeltaT, then the number of array elements and the programmed delay are rounded to the nearest increment of DeltaT. For example, to find the number of array elements required to perform the programmed delay given Deadtime = 4.25s and DeltaT = 0.50s:

4.25s / 0.50s = 8.5
rounds up to 9 array elements required

The actual delay applied to the input in this example is:

number of array elements x DeltaT = programmed delay or
9 x 0.5s = 4.5s

Runtime changes to either Deadtime or DeltaT change the point in which values are moved out of the buffer. The number of elements required to perform the programmed delay can either increase or decrease. Prior to servicing the deadtime buffer, the following updates occur:

If the number of required elements needs to increase, the new buffer elements are populated with the oldest value in the current deadtime buffer.

If the number of required elements needs to decrease, the oldest elements of the current deadtime buffer are discarded.

Instruction Behavior on InFault Transition

When InFault is true (bad), the instruction suspends execution, holds the last output, and sets the appropriate bit in Status.

When InFault transitions from true to false, the instruction sets all values in the deadtime buffer equal to In x Gain + Bias.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	N/A The instruction does not execute, but does validate input parameters.
Postscan	EnableIn and EnableOut bits are cleared to false.

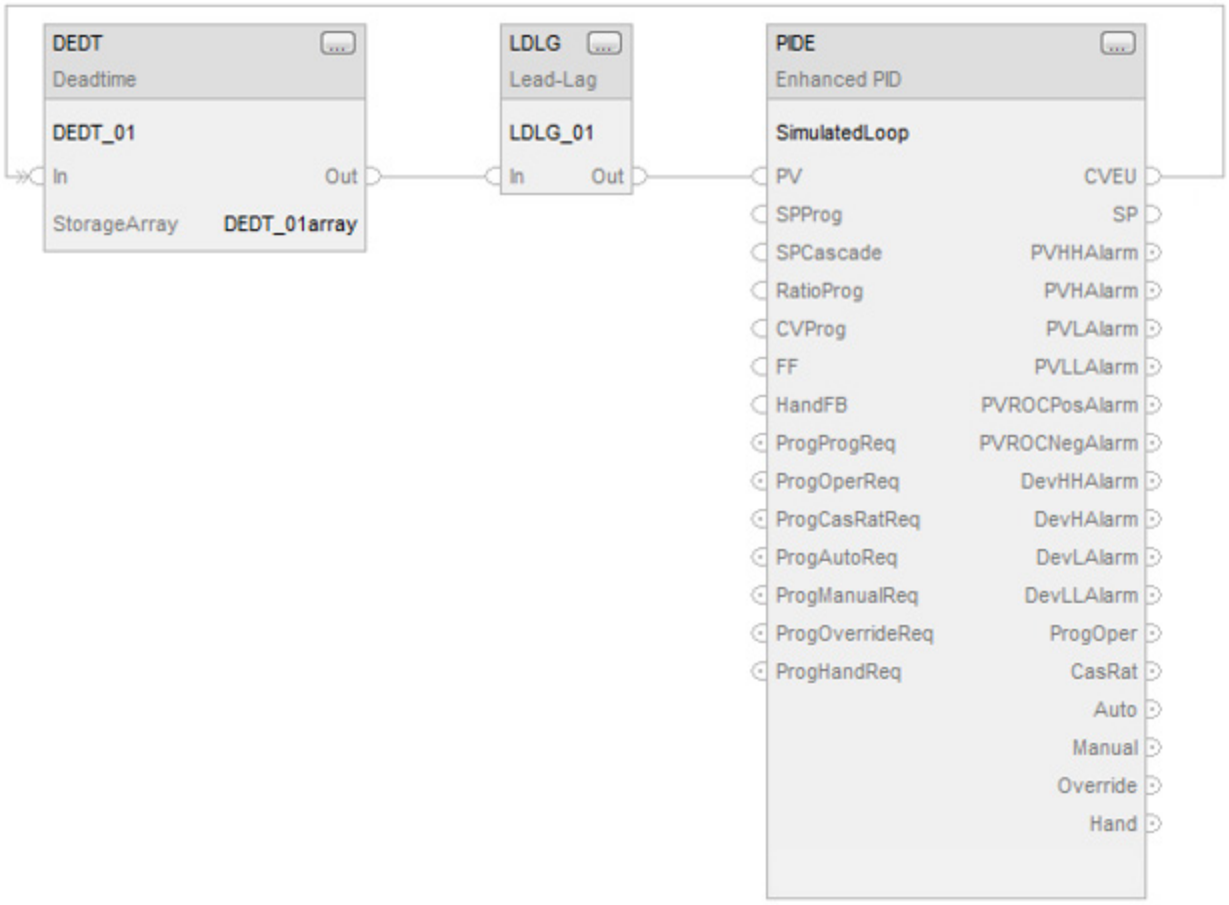
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

In this example, the DEDT instruction simulates a deadtime delay in a simulated process. The output of the PIDE instruction is passed through a deadtime delay and a first-order lag to simulate the process. The array DEDT_01array is a REAL array with 100 elements to support a deadtime of up to 100 samples. For example, if this routine executes every 100 msec, the array would support a deadtime of up to 10 seconds.

Function Block



See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Function Generator (FGEN)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

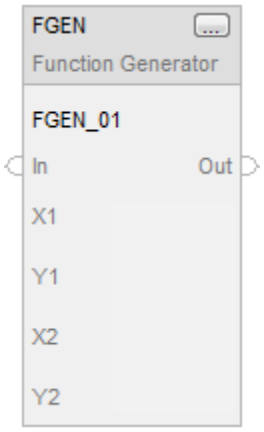
The FGEN instruction converts an input based on a piece-wise linear function.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram.

Function Block



Structured Text

FGEN(FGEN_tag,X1,Y1,X2,Y2);

Operands

Function Block

Operand	Type	Format	Description
FGEN tag	FUNCTION_ GENERATOR	structure	FGEN structure
X1	REAL	array	X-axis array, table one. Combine with the Y-axis array, table one to define the points of the first piece-wise linear curve. Valid = any float
Y1	REAL	array	Y-axis array, table one. Combine with the X-axis array, table one to define the points of the first piece-wise linear curve. Valid = any float
X2	REAL	array	(optional) X-axis array, table two. Combine with the Y-axis array, table two to define the points of the second piece-wise linear curve. Valid = any float
Y2	REAL	array	(optional) Y-axis array, table two. Combine with the X-axis array, table two to define the points of the second piece-wise linear curve. Valid = any float

Structured Text

Operand	Type	Format	Description
FGEN tag	FUNCTION_ GENERATOR	structure	FGEN structure
X1	REAL	array	X-axis array, table one. Combine with the Y-axis array, table one to define the points of the first piece-wise linear curve. Valid = any float
Y1	REAL	array	Y-axis array, table one. Combine with the X-axis array, table one to define the points of the first piece-wise linear curve. Valid = any float
X2	REAL	array	(optional) X-axis array, table two. Combine with the Y-axis array, table two to define the points of the second piece-wise linear curve. Valid = any float
Y2	REAL	array	(optional) Y-axis array, table two. Combine with the X-axis array, table two to define the points of the second piece-wise linear curve. Valid = any float

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

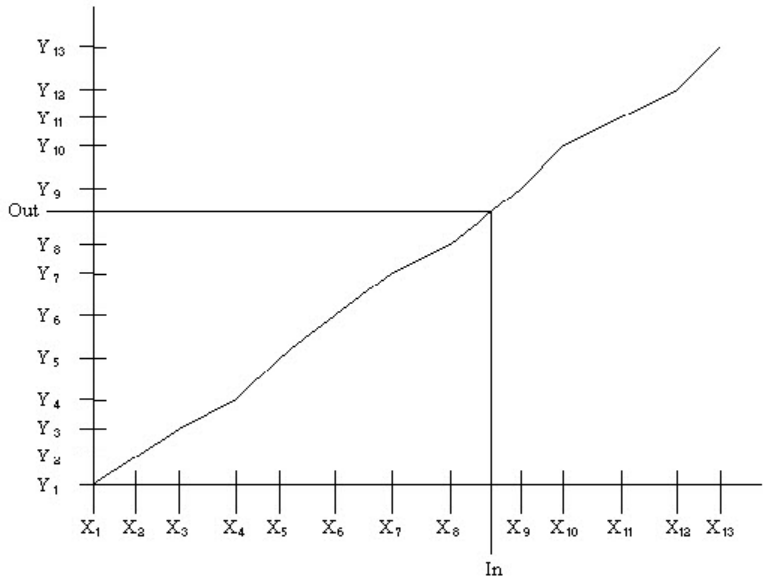
FUNCTION_GENERATOR Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0
XY1Size	DINT	Number of points in the piece-wise linear curve to use from table one. If the value is less than one and Select is cleared, the instruction sets the appropriate bit in Status and the output is not changed. Valid = 1 to (smallest of X1 and Y1 array sizes) Default = 1
XY2Size	DINT	Number of points in the piece-wise linear curve to use from table two. If the value is less than one and Select is set, the instruction sets the appropriate bit in Status and the output is not changed. Valid = 0 to (smallest of X2 and Y2 array sizes) Default = 0
Select	BOOL	This input determines which table to use. When cleared, the instruction uses table one; when set, the instruction uses table two. Default is cleared.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false on overflow
Out	REAL	Output of the instruction.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	Instruction generated a fault.
XY1SizeInv (Status.1)	BOOL	Size of table 1 is invalid or not compatible with the array size.
XY2SizeInv (Status.2)	BOOL	Size of table 2 is invalid or not compatible with the array size.
XisOutofOrder (Status.3)	BOOL	The X parameters are not sorted.

Description

The following illustration shows how the FGEN instruction converts a twelve-segment curve.



The X-axis parameters must follow the relationship:

X[1] < X[2] < X[3] < ... < X[XY<n>Size],

where XY<n>Size > 1 and is a number of points in the piece-wise linear curve and where n is 1 or 2 for the table selected. You must create sorted X-axis elements in the X arrays.

The Select input determines which table to use for the instruction. When the instruction is executing on one table, you can modify the values in the other table. Change the state of Select to execute with the other table.

Before calculating Out, the X axis parameters are scanned. If they are not sorted in ascending order, the appropriate bit in Status is set and Out remains

unchanged. Also, if XY1Size or XY2Size is invalid, the instruction sets the appropriate bit in Status and leaves Out unchanged.

The instruction uses this algorithm to calculate Out based on In:

- When In ≤ X[1], set Out = Y[1]
- When In > X[XY<n>Size], set Out = Y[XY<n>Size]
- When X[n] < In ≤ X[n+1], calculate Out = ((Y[n+1]-Yn)/ (X[n+1]-Xn))*(In-Xn)+Yn

Affects Math Status Flags

No

Major/Minor Fault

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Function Block

Condition	Function Block Action
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	EnableIn and EnableOut bits are set to true. The instruction executes.

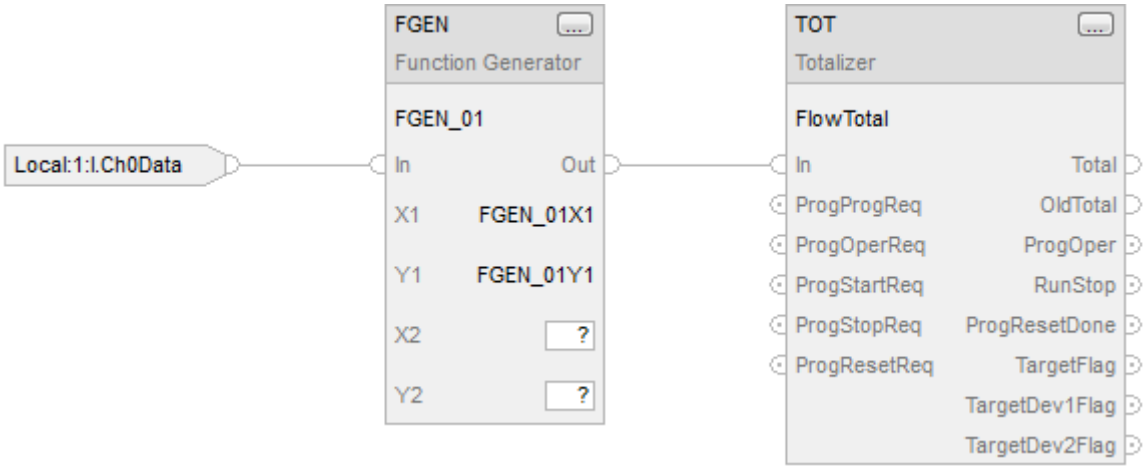
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Examples

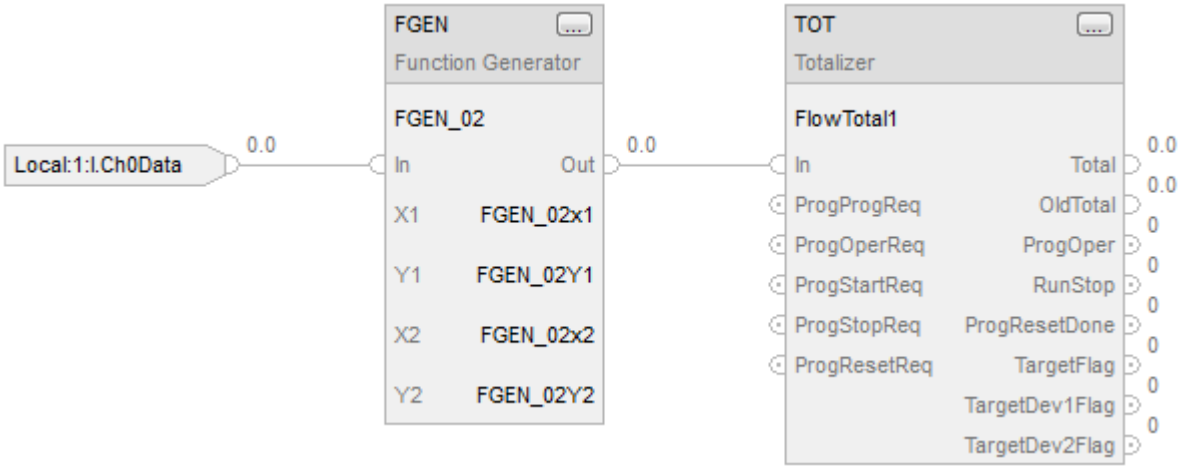
Example 1

In this example, the FGEN instruction characterizes a flow signal which is then totalized using a TOT instruction. The FGEN_01X1 and FGEN_01Y1 arrays are REAL arrays of 10 elements each to support up to a 9 segment curve. You can use arrays of any size to support a curve of any desired number of segments.



Example 2

This example passes optional parameters to FGEN instruction.



See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Lead-Lag (LDLG)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

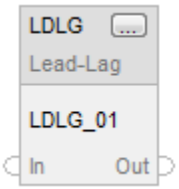
The LDLG instruction provides a phase lead-lag compensation for an input signal. This instruction is typically used for feedforward PID control or for process simulations.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram.

Function Block



Structured Text

LDLG(LDLG_tag);

Operands

Function Block

Operand	Type	Format	Description
LDLG tag	LEAD_LAG	Structure	LDLG structure

Structured Text

Operand	Type	Format	Description
LDLG tag	LEAD_LAG	Structure	LDLG structure

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

LEAD_LAG Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0
Initialize	BOOL	Request to initialize filter control algorithm. When Initialize is set, Out = (In x Gain) + Bias. Default = cleared
Lead	REAL	The lead time in seconds. Set Lead = 0.0 to disable the lead control algorithm. If Lead < 0.0, the instruction sets the appropriate bit in Status and limits Lead to 0.0. If Lead > maximum positive float, the instruction sets the appropriate bit in Status. Valid = any float ≥ 0.0 Default = 0.0
Lag	REAL	The lag time in seconds. The minimum lag time is DeltaT/2. If Lag < DeltaT/2, the instruction sets the appropriate bit in Status and limits Lag to DeltaT/2. If Lag > maximum positive float, the instruction sets the appropriate bit in Status. Valid = any float ≥ DeltaT/2 Default = 0.0
Gain	REAL	The process gain multiplier. This value allows the simulation of a process gain. The In signal is multiplied by this value. I = (In x Gain) + Bias Valid = any float Default = 1.0
Bias	REAL	The process offset level. This value allows the simulation of an ambient condition. This value is summed with the results of the multiplication of In times Gain. I = (In x Gain) + Bias Valid = any float Default = 0.0
TimingMode	DINT	Selects timing execution mode. 0 = Periodic rate 1 = Oversample mode 2 = Real-time sampling mode Valid = 0 to 2 Default = 0 For more information about timing modes, see Function Block Attributes.
OversampleDT	REAL	Execution time for oversample mode. Valid = 0 to 4194.303 seconds Default = 0

Input Parameter	Data Type	Description
RTSTime	DINT	Module update period for real time sampling mode Valid = 1 to 32,767ms Default = 1
RTSTimeStamp	DINT	Module time stamp value for real time sampling mode. Valid = 0 to 32,767ms Default = 0

Output Parameter	Data Type	Description
EnableOut	BOOL	Enable output.
Out	REAL	The calculated output of the algorithm. Math status flags are used for this output.
DeltaT	REAL	Elapsed time between updates. This is the elapsed time in seconds used by the control algorithm to calculate the process output.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
LeadInv (Status.1)	BOOL	Lead < minimum value or Lead > maximum value.
LagInv (Status.2)	BOOL	Lag < minimum value or Lag > maximum value.
TimingModeInv (Status.27)	BOOL	Invalid TimingMode value. For more information about timing modes, see Function Block Attributes.
RTSMissed (Status.28)	BOOL	Only used in real time sampling mode. Set when ABS | DeltaT - RTSTime | > 1 (.001 second).
RTSTimeInv (Status.29)	BOOL	Invalid RTSTime value.
RTSTimeStampInv (Status.30)	BOOL	Invalid RTSTimeStamp value.
DeltaTInv (Status.31)	BOOL	Invalid DeltaT value.

Description

The LDLG instruction supports one lead and lag in series. The instruction also allows configurable gain and bias factors. The LDLG instruction is designed to execute in a task where the scan rate remains constant.

The LDLG instruction uses this equation:

$$H(s) = \left[\frac{1 + Lead \times s}{1 + Lag \times s}\right]$$

with these parameter limits:

Parameter	Limitations
Lead	LowLimit = 0.0 HighLimit = maximum positive float

Parameter	Limitations
Lag	LowLimit = DeltaT/2 (DeltaT is in seconds) HighLimit = maximum positive float

Whenever the value computed for the output is invalid, NAN, or ± INF, the instruction sets Out = the invalid value and sets the Math overflow status flag. When the value computed for the output becomes valid, the instruction initializes the internal parameters and sets Out = (In x Gain) + Bias.

Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Function Block

Condition	Function Block Action
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	Execute "Out = (In * Gain) + Bias". Recalculate Lead/Lag coefficients.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

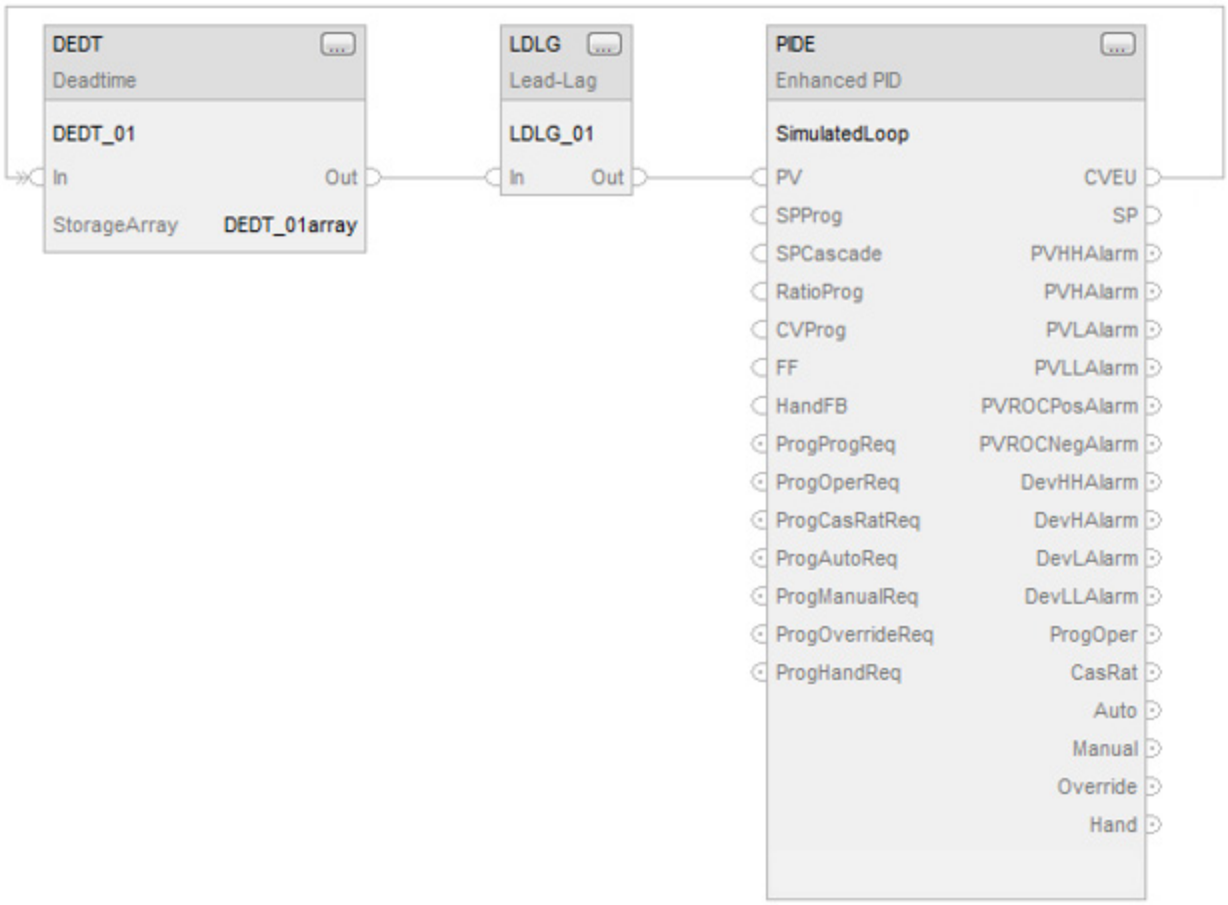
Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

The LDLG instruction in this example adds a first-order lag to a simulated process. Optionally, you could enter a Gain on the LDLG instruction to

simulate a process gain, and you could enter a Bias to simulate an ambient condition.

Function Block



Structured Text

DEDT_01.In := SimulatedLoop.CVEU;

DEDT(DEDT_01,DEDT_01_array);

LDLG_01.In := DEDT_01.Out;

LDLG(LDLG_01);

SimulatedLoop.PV := LDLG_01.Out;

PIDE(SimulatedLoop);

See also

Function Block Attributes on page 1043

Common Attributes on page 1083

Structured Text Syntax on page 1057

Enhanced PID (PIDE)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The PIDE instruction provides enhanced capabilities over the standard PID instruction. The instruction uses the velocity form of the PID algorithm. The gain terms are applied to the change in the value of error or PV, not the value of error or PV.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram.

Function Block

Structured Text

PIDE(PIDE_tag);

Operands

Function Block

Operand	Type	Format	Description
PIDE tag	PID_ENHANCED	structure	PIDE structure
autotune tag	PIDE_AUTOTUNE	structure	(optional) autotune structure

Structured Text

Operand	Type	Format	Description
PIDE tag	PID_ENHANCED	structure	PIDE structure

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

PIDE Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
PV	REAL	Scaled process variable input. This value is typically read from an analog input module. Valid = any float Default = 0.0
PVFault	BOOL	PV bad health indicator. If PV is read from an analog input, then PVFault is normally controlled by the analog input fault status. When PVFault is true, it indicates the input signal has an error. Default is false = "good health"
PVEUMax	REAL	Maximum scaled value for PV. The value of PV and SP which corresponds to 100 percent span of the Process Variable. Valid = PVEUMin < PVEUMax ≤ maximum positive float Default = 100.0
PVEUMin	REAL	Minimum scaled value for PV. The value of PV and SP which corresponds to 0 percent span of the Process Variable. Valid = maximum negative float ≤ PVEUMin < PVEUMax Default = 0.0
SPProg	REAL	SP program value, scaled in PV units. SP is set to this value when in Program control and not Cascade/Ratio mode. If the value of SPProg < SPLLimit or > SPHLimit, the instruction sets the appropriate bit in Status and limits the value used for SP. Valid = SPLLimit to SPHLimit Default = 0.0
SPOper	REAL	SP operator value, scaled in PV units. SP is set to this value when in Operator control and not Cascade/Ratio mode. If the value of SPOper < SPLLimit or > SPHLimit, the instruction sets the appropriate bit in Status and limits the value used for SP. Valid = SPLLimit to SPHLimit Default = 0.0
SPCascade	REAL	SP Cascade value, scaled in PV units. If CascadeRatio is true and UseRatio is false, then SP = SPCascade. This is typically the CVEU of a primary loop. If CascadeRatio and UseRatio are true, then SP = (SPCascade x Ratio). If the value of SPCascade < SPLLimit or > SPHLimit, set the appropriate bit in Status and limit the value used for SP. Valid = SPLLimit to SPHLimit Default = 0.0
SPHLimit	REAL	SP high limit value, scaled in PV units. If SPHLimit > PVEUMax, the instruction sets the appropriate bit in Status. Valid = SPLLimit to PVEUMax Default = 100.0
SPLLimit	REAL	SP low limit value, scaled in PV units. If SPLLimit < PVEUMin, the instruction sets the appropriate bit in Status. If SPHLimit < SPLLimit, the instruction sets the appropriate bit in Status and limits SP using the value of SPLLimit. Valid = PVEUMin to SPHLimit Default = 0.0

Input Parameter	Data Type	Description
UseRatio	BOOL	Allow ratio control permissive. Set to true to enable ratio control when in Cascade/Ratio mode. Default is false.
RatioProg	REAL	Ratio program multiplier. Ratio and RatioOper are set equal to this value when in Program control. If RatioProg < RatioLLimit or > RatioHLimit, the instruction sets the appropriate bit in Status and limits the value used for Ratio. Valid = RatioLLimit to RatioHLimit Default = 1.0
RatioOper	REAL	Ratio operator multiplier. Ratio is set equal to this value when in Operator control. If RatioOper < RatioLLimit or > RatioHLimit, the instruction sets the appropriate bit in Status and limits the value used for Ratio. Valid = RatioLLimit to RatioHLimit Default = 1.0
RatioHLimit	REAL	Ratio high limit value. Limits the value of Ratio obtained from RatioProg or RatioOper. If RatioHLimit < RatioLLimit, the instruction sets the appropriate bit in Status and limits Ratio using the value of RatioLLimit. Valid = RatioLLimit to maximum positive float Default = 1.0
RatioLLimit	REAL	Ratio low limit value. Limits the value of Ratio obtained from RatioProg or RatioOper. If RatioLLimit < 0, the instruction sets the appropriate bit in Status and limits the value to zero. If RatioHLimit < RatioLLimit, the instruction sets the appropriate bit in Status and limits Ratio using the value of RatioLLimit. Valid = 0.0 to RatioHLimit Default = 1.0
CVFault	BOOL	Control variable bad health indicator. If CVEU controls an analog output, then CVFault normally comes from the analog output's fault status. When true, CVFault indicates an error on the output module and the instruction sets the appropriate bit in Status. Default is false = "good health"
CVInitReq	BOOL	CV initialization request. This signal is normally controlled by the "In Hold" status on the analog output module controlled by CVEU or from the InitPrimary output of a secondary PID loop. Default is false.
CVInitValue	REAL	CVEU initialization value, scaled in CVEU units. When CVInitializing is true, CVEU = CVInitValue and CV equals the corresponding percentage value. CVInitValue comes from the feedback of the analog output controlled by CVEU or from the setpoint of a secondary loop. Instruction initialization is disabled when CVFaulted or CVEUSpanInv is true. Valid = any float Default = 0.0
CVProg	REAL	CV program manual value. CV equals this value when in Program Manual mode. If CVProg < 0 or > 100, or < CVLLimit or > CVHLimit when CVManLimiting is true, the instruction sets the appropriate bit in Status and limits the CV value. Valid = 0.0 to 100.0 Default = 0.0
CVOper	REAL	CV operator manual value. CV equals this value when in Operator Manual mode. If not Operator Manual mode, the instruction sets CVOper = CV at the end of each instruction execution. If CVOper < 0 or > 100, or < CVLLimit or > CVHLimit when CVManLimiting is true, the instruction sets the appropriate bit in Status and limits the CV value. Valid = 0.0 to 100.0 Default = 0.0

Input Parameter	Data Type	Description
CVOverride	REAL	CV override value. CV equals this value when in override mode. This value should correspond to a safe state output of the PID loop. If CVOverride < 0 or >100, the instruction sets the appropriate bit in Status and limits the CV value. Valid = 0.0 to 100.0 Default = 0.0
CVPrevious	REAL	CV_{n-1} value. If CVSetPrevious is set, CV_{n-1} equals this value. CV_{n-1} is the value of CV from the previous execution. CVPrevious is ignored when in manual, override or hand mode or when CVInitializing is set. If CVPrevious < 0 or > 100, or < CVLLimit or > CVHLimit when in Auto or cascade/ratio mode, the instruction sets the appropriate bit in Status and limits the CVn-1 value. Valid = 0.0 to 100.0 Default = 0.0
CVSetPrevious	BOOL	Request to use CVPrevious. If true, CVn-1 = CVPrevious Default is false.
CVManLimiting	BOOL	Limit CV in manual mode request. If Manual mode and CVManLimiting is true, CV is limited by the CVHLimit and CVLLimit values. Default is false.
CVEUMax	REAL	Maximum value for CVEU. The value of CVEU which corresponds to 100 percent CV. If CVEUMax = CVEUMin, the instruction sets the appropriate bit in Status. Valid = any float Default = 100.0
CVEUMin	REAL	Minimum value of CVEU. The value of CVEU which corresponds to 0 percent CV. If CVEUMax = CVEUMin, the instruction sets the appropriate bit in Status. Valid = any float Default = 0.0
CVHLimit	REAL	CV high limit value. This is used to set the CVHAlarm output. It is also used for limiting CV when in Auto or Cascade/Ratio mode, or Manual mode if CVManLimiting is true. If CVHLimit > 100 or < CVLLimit, the instruction sets the appropriate bit in Status. If CVHLimit < CVLLimit, the instruction limits CV using the value of CVLLimit. Valid = CVLLimit < CVHLimit $\leq$ 100.0 Default = 100.0
CVLLimit	REAL	CV low limit value. This is used to set the CVLAlarm output. It is also used for limiting CV when in Auto or Cascade/Ratio mode, or Manual mode if CVManLimiting is true. If CVLLimit < 0 or CVHLimit < CVLLimit, the instruction sets the appropriate bit in Status. If CVHLimit < CVLLimit, the instruction limits CV using the value of CVLLimit. Valid = 0.0 $\leq$ CVLLimit < CVHLimit Default = 0.0
CVROCLimit	REAL	CV rate of change limit, in percent per second. Rate of change limiting is only used when in Auto or Cascade/Ratio modes or Manual mode if CVManLimiting is true. Enter 0 to disable CV ROC limiting. If CVROCLimit < 0, the instruction sets the appropriate bit in Status and disables CV ROC limiting. Valid = 0.0 to maximum positive float Default = 0.0
FF	REAL	Feed forward value. The value of feed forward is summed with CV after the zero-crossing deadband limiting has been applied to CV. Therefore changes in FF are always reflected in the final output value of CV. If FF < -100 or > 100, the instruction sets the appropriate bit in Status and limits the value used for FF. Valid = -100.0 to 100.0 Default = 0.0

Input Parameter	Data Type	Description
FFPrevious	REAL	FF_{n-1} value. If FF SetPrevous is set, the instruction sets FF_{n-1} = FFPrevious. FF_{n-1} is the valu eof FF from the previous execution. If FFPrevious < -100 or > 100, the instruction sets the appropriate bit in Status and limits value used for FF_{n-1} Valid = -100.0 to 100.0 Default - 0.0
FFSetPrevious	BOOL	Request to use FFPrevious. If true, FFn-1 = FFPrevious. Default is false.
HandFB	REAL	CV Hand feedback value. CV equals this value when in Hand mode and HandFBFault is false (good health). This value typically comes from the output of a field mounted hand/ auto station and is used to generate a bumpless transfer out of hand mode. If HandFB < 0 or > 100, the instruction sets the appropriate bit in Status and limits the value used for CV. Valid = 0.0 to 100.0 Default = 0.0
HandFBFault	BOOL	HandFB value bad health indicator. If the HandFB value is read from an analog input, then HandFBFault is typically controlled by the status of the analog input channel. When true, HandFBFault indicates an error on the input module and the instruction sets the appropriate bit in Status. Default is false = "good health"
WindupHIn	BOOL	Windup high request. When true, the CV cannot integrate in a positive direction. The signal is typically obtained from the WindupHOut output from a secondary loop. Default is false.
WindupLIn	BOOL	Windup low request. When true, the CV cannot integrate in a negative direction. This signal is typically obtained from the WindupLOut output from a secondary loop. Default is false.
ControlAction	BOOL	Control action request. Set to true to calculate error as E = PV - SP; clear to false to calculate error as E = SP - PV. Default is false.
DependIndepend	BOOL	Dependent/independent control request. When true, use the dependent form of the PID equation; when false, use the independent form of the equations. Default is false.
PGain	REAL	Proportional gain. When the independent form of the PID algorithm is selected, enter the unitless proportional gain into this value. When the dependent PID algorithm is selected, enter the unitless controller gain into this value. Enter 0 to disable the proportional control. If PGain < 0, the instruction sets the appropriate bit in Status and uses a value of PGain = 0. Valid = 0.0 to maximum positive float Default = 0.0
IGain	REAL	Integral gain. When the independent form of the PID algorithm is selected, enter the integral gain in units of 1/minutes into this value. When the dependent PID algorithm is selected, enter the integral time constant in units of minutes/repeat into this value. Enter 0 to disable the integral control. If IGain < 0, the instruction sets the appropriate bit in Status and uses a value of IGain = 0. Valid = 0.0 to maximum positive float Default = 0.0
DGain	REAL	Derivative gain. When the independent form of the PID algorithm is selected, enter the derivative gain in units of minutes into this value. When the dependent PID algorithm is used, enter the derivative time constant in units of minutes into this value. Enter 0 to disable the derivative control. If DGain < 0, the instruction sets the appropriate bit in Status and uses a value of DGain = 0. Valid = 0.0 to maximum positive float Default = 0.0

Input Parameter	Data Type	Description
PVEProportional	BOOL	Proportional PV control request. When true, calculate the proportional term (DeltaPTerm) using the change in process variable (PVPercent). When false, use the change in error (EPercent). Default is false.
PVEDerivative	BOOL	Derivative PV control request. When true, calculate the derivative term (DeltaDTerm) using the change in process variable (PVPercent). When false, use the change in error (EPercent). Default is true.
DSmoothing	BOOL	Derivative Smoothing request. When true, changes in the derivative term are smoothed. Derivative smoothing causes less output "jitters" as a result of a noisy PV signal but also limits the effectiveness of high derivative gains. Default is false.
PVTracking	BOOL	SP track PV request. Set to true to cause SP to track PV when in manual mode. Ignored when in Cascade/Ratio or Auto mode. Default is false.
ZCDeadband	REAL	Zero crossing deadband range, scaled in PV units. Defines the zero crossing deadband range. Enter 0 to disable the zero crossing deadband checking. If ZCDeadband < 0, the instruction sets the appropriate bit in Status and disables zero crossing deadband checking. Valid = 0.0 to maximum positive float Default = 0.0
ZCOff	BOOL	Zero crossing disable request. Set to true to disable zero crossing for the deadband calculation. Default is false.
PVHHLimit	REAL	PV high-high alarm limit value, scaled in PV units. Valid = any float Default = maximum positive float
PVHLimit	REAL	PV high alarm limit value, scaled in PV units. Valid = any float Default = maximum positive float
PVLLimit	REAL	PV low alarm limit value, scaled in PV units. Valid = any float Default = maximum negative float
PVLLLimit	REAL	PV low-low alarm limit value, scaled in PV units. Valid = any float Default = maximum negative float
PVDeadband	REAL	PV alarm limit deadband value, scaled in PV units. Deadband is the delta value between the turn-on and turn-off value for each of the PV alarm limits. If PVDeadband < 0.0, the instruction sets the appropriate bit in Status and limits PVDeadband to zero. Valid = 0.0 to maximum positive float Default = 0.0
PVROCPosLimit	REAL	PV positive rate of change alarm limit. The limit value for a positive (increasing) change in PV, scaled in PV units per seconds. Enter 0.0 to disable positive PVROC alarm checking. If PVROCPosLimit < 0.0, the instruction sets the appropriate bit in Status and disables positive PVROC checking. Valid = 0.0 to maximum positive float Default = 0.0 PV/second
PVROCNegLimit	REAL	PV negative rate of change alarm limit. The limit value for a negative (decreasing) change in PV, scaled in PV units per seconds. Enter 0.0 to disable negative PVROC alarm checking. If PVROCNegLimit < 0, the instruction sets the appropriate bit in Status and disables negative PVROC checking. Valid = 0.0 to maximum positive float Default = 0.0

Input Parameter	Data Type	Description
PVROCPeriod	REAL	PV rate of change sample period. The time period, in seconds, over which the rate of change for PV is evaluated. Enter 0 to disable PVROC alarm checking If PVROCPeriod < 0.0, the instruction sets the appropriate bit in Status, and disables positive and negative PVROC checking. Valid = any float ≥ 0.0 Default = 0.0 seconds
DevHHLimit	REAL	Deviation high-high alarm limit value, scaled in PV units. Deviation is the difference in value between the process variable (PV) and the setpoint (SP). Deviation alarming alerts the operator to a discrepancy between the process variable and the setpoint value. If DevHHLimit < 0.0, the instruction sets the appropriate bits in Status and sets DevHHLimit = 0.0. Valid = 0.0 to maximum positive float Default = maximum positive float
DevHLimit	REAL	Deviation high alarm limit value, scaled in PV units. Deviation is the difference in value between the process variable (PV) and the setpoint (SP). Deviation alarming alerts the operator to a discrepancy between the process variable and the setpoint value. If DevHLimit < 0.0, the instruction sets the appropriate bit in Status and sets DevHLimit = 0.0. Valid = 0.0 to maximum positive float Default = maximum positive float
DevLLimit	REAL	Deviation low alarm limit value, scaled in PV units. Deviation is the difference in value between the process variable (PV) and the setpoint (SP). Deviation alarming alerts the operator to a discrepancy between the process variable and the setpoint value. If DevLLimit < 0.0, the instruction sets the appropriate bit in Status and sets DevLLimit = 0.0. Valid = 0.0 to maximum positive float Default = maximum positive float
DevLLLimit	REAL	Deviation low-low alarm limit value, scaled in PV units. Deviation is the difference in value between the process variable (PV) and the setpoint (SP). Deviation alarming alerts the operator to a discrepancy between the process variable and the setpoint value. If DevLLLimit < 0.0, the instruction sets the appropriate bit in Status and sets DevLLLimit = 0.0. Valid = 0.0 to maximum positive float Default = maximum positive float
DevDeadband	REAL	The deadband value for the Deviation alarm limits, scaled in PV units. Deadband is the delta value between the turn-on and turn-off value for each of the Deviation alarm limits. If DevDeadband < 0.0, the instruction sets the appropriate bit in Status and sets DevDeadband = 0.0. Valid = 0.0 to maximum positive float Default = 0.0
AllowCasRat	BOOL	Allow cascade/ratio mode permissive. Set to true to allow Cascade/Ratio mode to be selected using either ProgCasRatReq or OperCasRatReq. Default is false.
ManualAfterInit	BOOL	Manual mode after initialization request. When true, the instruction is placed in Manual mode when CVInitializing is true, unless the current mode is Override or Hand. When ManualAfterInit is false, the instruction's mode is not changed, unless requested to do so. Default is false.
ProgProgReq	BOOL	Program program request. Set to true by the user program to request Program control. Ignored if ProgOperReq is true. Holding this true and ProgOperReq false locks the instruction in Program control. When ProgValueReset is true, the instruction clears this input to false at each execution. Default is false.

Input Parameter	Data Type	Description
ProgOperReq	BOOL	Program operator request. Set to true by the user program to request Operator control. Holding this true locks the instruction in Operator control. When ProgValueReset is true, the instruction clears this input to false at each execution. Default is false.
ProgCasRatReq	BOOL	Program cascade/ratio mode request. Set to true by the user program to request Cascade/Ratio mode. When ProgValueReset is true, the instruction clears this input to false at each execution. Default is false.
ProgAutoReq	BOOL	Program auto mode request. Set to true by the user program to request Auto mode. When ProgValueReset is true, the instruction clears this input to false at each execution. Default is false.
ProgManualReq	BOOL	Program manual mode request. Set to true by the user program to request Manual mode. When ProgValueReset is true, the instruction clears this input to false ateach execution. Default is false.
ProgOverrideReq	BOOL	Program override mode request. Set to true by the user program to request Override mode. When ProgValueReset is true, the instruction clears this input to false at each execution. Default is false.
ProgHandReq	BOOL	Program hand mode request. Set to true by the user program to request Hand mode. This value is usually read as a digital input from a hand/auto station. When ProgValueReset is true, the instruction clears this input to false at each execution. Default is false.
OperProgReq	BOOL	Operator program request. Set to true by the operator interface to request Program control. The instruction clears this input to false at each execution. Default is false.
OperOperReq	BOOL	Operator operator request. Set to true by the operator interface to request Operator control. The instruction clears this input to false at each execution. Default is false.
OperCasRatReq	BOOL	Operator cascade/ratio mode request. Set to true by the operator interface to request Cascade/ Ratio mode. The instruction clears this input to false at each execution. Default is false.
OperAutoReq	BOOL	Operator auto mode request. Set to true by the operator interface to request Auto mode. The instruction clears this input to false at each execution. Default is false.
OperManualReq	BOOL	Operator manual mode request. Set to true by the operator interface to request Manual mode. The instruction clears this input to false at each execution. Default is false.
ProgValueReset	BOOL	Reset program control values. When true, all the program request inputs are cleared to false by the instruction at each execution. When true and in Operator control, the instruction sets SPProgram = SP and CVProgram = CV. Default is false.
TimingMode	DINT	Selects timing execution mode. 0 = Periodic mode 1 = Oversample mode 2 = Real time sampling mode For more information about timing modes, see Function Block Attributes. Valid = 0 to 2 Default = 0

Input Parameter	Data Type	Description
OversampleDT	REAL	Execution time for oversample mode. Valid = 0 to 4194.303 seconds Default = 0
RTSTime	DINT	Module update period for real time sampling mode Valid = 1 to 32,767ms Default = 1
RTSTimeStamp	DINT	Module time stamp value for real time sampling mode. Valid = 0 to 32,767ms Default = 0
AtuneAcquire	BOOL	Acquire PIDE AtuneData request. Default is false.
AtuneStart	BOOL	Start Autotune request. Default is false.
AtuneUseGains	BOOL	Use Autotune gains request. Default is false.
AtuneAbort	BOOL	Abort Autotune request. Default is false.
AtuneUnacquire	BOOL	Unacquire PIDE AtuneData request. Default is false.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if CVEU overflows.
CVEU	REAL	Scaled control variable output. Scaled using CVEUMax and CVEUMin, where CVEUMax corresponds to 100 percent and CVEUMin corresponds to 0 percent. This output typically controls an analog output module or a secondary loop. CVEU = (CV x CVEUSpan / 100) + CVEUMin CVEU span calculation: CVEUSpan = (CVEUMax - CVEUMin)
CV	REAL	Control variable output. This value is expressed as 0 to 100 percent. CV is limited by CVHLimit and CVLLimit when in auto or cascade/ratio mode or manual mode if CVManLimiting is true. Otherwise this value is limited by 0 and 100 percent.
CVInitializing	BOOL	Initialization mode indicator. CVInitializing is set to true when CVInitReq is true, during instruction first scan, and on a true to false transition of CVHealth (bad to good). CVInitializing is cleared to false after the instruction has been initialized and CVInitReq is false.
CVHAlarm	BOOL	CV high alarm indicator. Set to true when the calculated value of CV > 100 or CVHLimit.
CVLAlarm	BOOL	CV low alarm indicator. Set to true when the calculated value of CV < 0 or CVLLimit.
CVROCAlarm	BOOL	CV rate of change alarm indicator. Set to true when the calculated rate of change for CV exceeds CVROCLimit.
SP	REAL	Current setpoint value. The value of SP is used to control CV when in Auto or Cascade/ Ratio mode.
SPPercent	REAL	The value of SP expressed in percent of span of PV. SPPercent = ((SP - PVEUMin) x 100) / PVSpan PV Span calculation: PVSpan = (PVEUMax - PVEUMin)
SPHAlarm	BOOL	SP high alarm indicator. Set to true when the SP > SPHLimit.

Output Parameter	Data Type	Description
SPLAlarm	BOOL	SP low alarm indicator. Set to true when the SP < SPLLimit.
PVPercent	REAL	PV expressed in percent of span. PVPercent = ((PV- PVEUMin) x 100) / PVSpan PV Span calculation: PVSpan = (PVEUMax - PVEUMin)
E	REAL	Process error. Difference between SP and PV, scaled in PV units.
EPercent	REAL	The error expressed as a percent of span.
InitPrimary	BOOL	Initialize primary loop command. Set to true when not in Cascade/Ratio mode or when CVInitializing is true. This signal is normally used by the CVInitReq input of a primary PID loop.
WindupHOut	BOOL	Windup high indicator. Set to true when CV high or CV low limit (depending on the control action) or SP high limit has been reached. This signal is typically used by the WindupHIn input to prevent the windup of the CV output on a primary loop.
WindupLOut	BOOL	Windup low indicator. Set to true when CV high or CV low limit (depending on the control action) or SP low limit has been reached. This signal is typically used by the WindupLIn input to prevent the windup of the CV output on a primary loop.
Ratio	REAL	Current ratio multiplier.
RatioHAlarm	BOOL	Ratio high alarm indicator. Set to true when Ratio > RatioHLimit.
RatioLAlarm	BOOL	Ratio low alarm indicator. Set to true when Ratio < RatioLLimit.
ZCDeadbandOn	BOOL	Zero crossing deadband indicator. When true the value of CV does not change. If ZCOff is true, then ZCDeadbandOn is set to true when | E | is within the ZCDeadband range. If ZCOff is false, then ZCDeadbandOn is set to true when | E | crosses zero and remains within the ZCDeadband range. ZCDeadbandOn is cleared to false when | E | exceeds the deadband range or when ZCDeadband = 0.
PVHHAlarm	BOOL	PV high-high alarm indicator. Set to true when PV $\geq$ PVHHLimit. Cleared to false when PV < (PVHHLimit - PVDeadband)
PVHAlarm	BOOL	PV high alarm indicator. Set to true when PV $\geq$ PVHLimit. Cleared to false when PV < (PVHLimit - PVDeadband)
PVLAlarm	BOOL	PV low alarm indicator. Set to true when PV $\leq$ PVLLimit. Cleared to false when PV > (PVLLimit + PVDeadband)
PVLLAlarm	BOOL	PV low-low alarm indicator. Set to true when PV $\leq$ PVLLLimit. Cleared to false when PV > (PVLLLimit + PVDeadband)
PVROCPosAlarm	BOOL	PV positive rate-of-change alarm indicator. Set to true when calculated PV rate-of-change $\geq$ PVROCPosLimit.
PVROCNegAlarm	BOOL	PV negative rate-of-change alarm indicator. Set to true when calculated PV rate-of-change $\leq$ (PVROCNegLimit x -1).
DevHHAlarm	BOOL	Deviation high-high alarm indicator. Set to true when PV $\geq$ (SP + DevHHLimit). Cleared to false when PV < (SP + DevHHLimit - DevDeadband)
DevHAlarm	BOOL	Deviation high alarm indicator. Set to true when PV $\geq$ (SP + DevHLimit). Cleared to false when PV < (SP + DevHLimit - DevDeadband)
DevLAlarm	BOOL	Deviation low alarm indicator. Set to true when PV $\leq$ (SP - DevLLimit). Cleared to false when PV > (SP - DevLLimit + DevDeadband)
DevLLAlarm	BOOL	Deviation low-low alarm indicator. Set to true when PV $\leq$ (SP - DevLLLimit). Cleared to false when PV > (SP - DevLLLimit + DevDeadband)
ProgOper	BOOL	Program/operator control indicator. Set to true when in Program control. Cleared to false when in Operator control.
CasRat	BOOL	Cascade/ratio mode indicator. Set to true when in the Cascade/Ratio mode.
Auto	BOOL	Auto mode indicator. Set to true when in the Auto mode.

Output Parameter	Data Type	Description
Manual	BOOL	Manual mode indicator. Set to true when in the Manual mode.
Override	BOOL	Override mode indicator. Set to true when in the Override mode.
Hand	BOOL	Hand mode indicator. Set to true when in the Hand mode.
DeltaT	REAL	Elapsed time between updates. This is the elapsed time in seconds used by the control algorithm to calculate the process output.
AtuneReady	BOOL	Set to true when the specified AtuneData has been acquired by the PIDE instruction.
AtuneOn	BOOL	Set to true when auto tuning has been initiated.
AtuneDone	BOOL	Set to true when auto tuning has completed.
AtuneAborted	BOOL	Set to true when auto tuning has been aborted by the user or due to errors that occurred during the auto tuning operation.
AtuneBusy	BOOL	Set to true when the specified AtuneData could not be acquired because it is currently acquired by another PIDE instruction.
Status1	DINT	Status of the function block.
InstructFault (Status1.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
PVFaulted (Status1.1)	BOOL	Process variable (PV) health bad.
CVFaulted (Status1.2)	BOOL	Control variable (CV) health bad.
HandFBFaulted (Status1.3)	BOOL	HandFB value health bad.
PVSpanInv (Status1.4)	BOOL	Invalid span of PV. PVEUMax £ PVEUMin.
SPProgInv (Status1.5)	BOOL	SPProg < SPLLimit or SPProg > SPHLimit. The instruction uses the limited value for SP.
SPOperInv (Status1.6)	BOOL	SPOper < SPLLimit or SPOper > SPHLimit. The instruction uses the limited value for SP.
SPCascadeInv (Status1.7)	BOOL	SPCascade < SPLLimit or SPCascade > SPHLimit. The instruction uses the limited value for SP.
SPLimitsInv (Status1.8)	BOOL	Limits invalid: SPLLimit < PVEUMin, SPHLimit > PVEUMax, or SPHLimit < SPLLimit. If SPHLimit < SPLLimit, the instruction limits the value using SPLLimit
RatioProgInv (Status1.9)	BOOL	RatioProg < RatioLLimit or RatioProg > RatioHLimit. The instruction limits the value for Ratio.
RatioOperInv (Status1.10)	BOOL	RatioOper < RatioLLimit or RatioOper > RatioHLimit. The instruction limits the value for Ratio.
RatioLimitsInv (Status1.11)	BOOL	RatioLLimit < 0 or RatioHLimit < RatioLLimit.
CVProgInv (Status1.12)	BOOL	CVProg < 0 or CVProg > 100, or CVProg < CVLLimit or CVProg > CVHLimit when CVManLimiting is true. The instruction limits the value for CV.
CVOperInv (Status1.13)	BOOL	CVOper < 0 or CVOper > 100, or CVOper < CVLLimit or CVOper > CVHLimit when CVManLimiting is true. The instruction limits the value for CV.
CVOverrideInv (Status1.14)	BOOL	CVOverride < 0 or CVOverride > 100. The instruction limits the value for CV.
CVPreviousInv (Status1.15)	BOOL	CVPrevious < 0 or CVPrevious > 100, or CVPrevious < CVLLimit or CVPrevious > CVHLimit when in Auto or Cascade/Ratio mode. The instruction limits the value of CVn-1.
CVEUSpanInv (Status1.16)	BOOL	Invalid CVEU span. The instruction uses a value of CVEUMax = CVEUMin.
CVLimitsInv (Status1.17)	BOOL	CVLLimit < 0, CVHLimit > 100, or CVHLimit < CVLLimit. If CVHLimit < CVLLimit, the instruction limits CV using CVLLimit.
CVROCLimitInv (Status1.18)	BOOL	CVROCLimit < 0. The instruction disables ROC limiting.

Output Parameter	Data Type	Description
FFInv (Status1.19)	BOOL	FF < -100 or FF > 100. The instruction uses the limited value for FF.
FFPreviousInv (Status1.20)	BOOL	FFPrevious < - 100 or FFPrevious > 100. The instruction uses the limited value for FF_{n-1}.
HandFBInv (Status1.21)	BOOL	HandFB < 0 or HandFB > 100. The instruction uses the limited value for CV.
PGainInv (Status1.22)	BOOL	PGain < 0. The instruction uses a value of PGain = 0.
IGainInv (Status1.23)	BOOL	IGain < 0. The instruction uses a value of IGain = 0.
DGainInv (Status1.24)	BOOL	DGain < 0. The instruction uses a value of DGain = 0.
ZCDeadbandInv (Status1.25)	BOOL	ZCDeadband < 0. The instruction disables zero crossing deadband.
PVDeadbandInv (Status1.26)	BOOL	PVDeadband < 0. The instruction limits PVDeadband to zero.
PVROCLimitsInv (Status1.27)	BOOL	PVROCPosLimit < 0, PVROCNegLimit < 0, or PVROCPeriod < 0.
DevHLLimitsInv (Status1.28)	BOOL	Deviation high-low limits invalid. Low-low limit < 0, low limit < 0, high limit < 0, or high-high limit < 0. The instruction uses 0 for the invalid limit.
DevDeadbandInv (Status1.29)	BOOL	Deviation deadband < 0. The instruction uses a value of DevDeadband = 0.
Status2	DINT	Timing status of the function block.
TimingModeInv (Status2.27)	BOOL	Invalid TimingMode value. For more information about timing modes, see Function Block Attributes.
RTSMissed (Status2.28)	BOOL	Only used in real time sampling mode. Set to true when ABS | DeltaT - RTSTime | > 1 (.001 second).
RTSTimeInv (Status2.29)	BOOL	Invalid RTSTime value.
RTSTimeStampInv (Status2.30)	BOOL	Invalid RTSTimeStamp value.
DeltaTInv (Status2.31)	BOOL	Invalid DeltaT value.

Description

The PID algorithm regulates the CV output in order to maintain the PV at the SP when the instruction executes in Cascade/Ratio or Auto modes.

When ControlAction is set, the calculated value of EPercent and PVPIDPercent is negated before being used by the control algorithm.

The following table describes how the instruction calculates the PID terms.

PID Term	Method of Calculation
proportional	The proportional term is calculated using: • PV when PVEProportional is set or • Error when PVEProportional is cleared Set PGain = 0 to disable proportional control.

PID Term	Method of Calculation
integral	The integral term is calculated using Error. Set IGain = 0 to disable integral control. Also, setting PGain = 0 when DependIndepend is set will disable integral control.
derivative	The derivative term is calculated using: • PV when PVEDerivative is set or • Error when PVEDerivative is cleared Set DGain = 0 to disable derivative control. Also, setting PGain = 0 when DependIndepend is set will disable derivative control. Derivative smoothing is enabled when DSmoothing is set and disabled when DSmoothing is cleared. Derivative smoothing causes less CV output "jitter" as a result of a noisy PV signal but also limits the effectiveness of high derivative gains.

Computing CV

The PID control algorithm computes the value for CV by summing Delta PTerm, Delta ITerm, Delta

DTerm, and CV from the previous execution of the instruction, for example CV_{n-1}. When CVsetPrevious is set,

CVn-1 is set equal to CVPrevious. This lets you preset CV_{n-1} to a specified value before computing the value of CV.

CalculatedCV = CV_{n-1} + DΔPTerm + DITerm + DDTerm

PIDE Algorithms

The PIDE instruction uses a velocity form PID algorithm similar to that used in most DCS systems. Some advantages to a velocity form algorithm include:

- Bumpless adaptive gain changes – You can change gains on the fly without initializing the algorithm.
- Multi-loop control schemes – You can implement cross limiting between loops by manipulating the CV_{n-1} term.

Independent Gains Form

$$CV_n = CV_{n-1} + K_P\Delta E + \frac{K_I}{60}E\Delta t + 60K_D\frac{E_n - 2E_{n-1} + E_{n-2}}{\Delta t}$$

In this form of the algorithm, each term of the algorithm (proportional, integral, and derivative) has a separate gain. Changing one gain only affects that term and not any of the others, where:

PIDE term:	Description:
CV	Control variable

PIDE term:	Description:
E	Error in percent of span
Δt	Update time in seconds used by the loop
K_p	Proportional gain
K_I	Integral gain in min $^{-1}$ a larger value of K_I causes a faster integral response.
K_D	Derivative gain in minutes

Dependent Gains Form

$$CV_n = CV_{n-1} + K_C\left(\Delta E + \frac{1}{60T_I}E\Delta t + 60T_D\frac{E_n - 2E_{n-1} + E_{n-2}}{\Delta t}\right)$$

This form of the algorithm changes the proportional gain into a controller gain. By changing the controller gain, you change the action of all three terms (proportional, integral, and derivative) at the same time, where:

PIDE term:	Description:
CV	Control variable
E	Error in percent of span
Δt	Update time in seconds used by the loop
K_c	Controller gain
T_I	Integral time constant in minutes per repeat a larger value of T_I causes a slower integral response It takes T_I minutes for the integral term to repeat the action of the proportional term in response to a step change in error.
T_D	Derivative time in constant in minutes

PIDE term:	Description:
CV	Control variable
E	Error in percent of span
Δt	Update time in seconds used by the loop
Kp	Proportional gain
K_I	Integral gain in min $^{-1}$ a larger value of K_I causes a faster integral response.
K_D	Derivative gain in minutes

Determining Which Algorithm to Use

- $K_P = K_C$
- $K_I = \frac{K_C}{T_I}$
- $K_D = K_C T_D$

The PIDE equations above are representative of the algorithms used by the PIDE instruction. You can substitute the change in error values for the change in PV (in percent of span) for the proportional and derivative terms by manipulating the parameters PVEProportional and PVEDerivative. By default, the PIDE instruction uses the change in error for the proportional term and the change in PV for the derivative term. This eliminates large derivative spikes on changes in setpoint.

You can convert the gains used between the different PIDE algorithm forms using these equations:

- $K_P = K_C$
- $K_I = \frac{K_C}{T_I}$
- $K_D = K_C T_D$

Each algorithm provides identical control with the appropriate gains. Some people prefer the independent gains style because they can manipulate individual gains without affecting the other terms. Others prefer the dependent gains style because they can, at least to a certain extent, change just the controller gain and cause an overall change in the aggressiveness of the PID loop without changing each gain separately.

Monitoring the PIDE Instruction

There is an operator faceplate available for the PIDE instruction.

Autotuning the PIDE Instruction

The Logix Designer application PIDE autotuner provides an open-loop autotuner built into the PIDE instruction. You can autotune from PanelView terminal or any other operator interface devices, as well as the Logix Designer application. The PIDE block has an Autotune Tag (type PIDE_AUTOTUNE) that you specify for those PIDE blocks that you want to autotune.

The PIDE autotuner is installed with the application, but you need an activation key to enable it. The autotuner is only supported in function block programming; it is not available in ladder diagram or structured text programming.

Use the Autotune tab to specify and configure the autotune tag for a PIDE block.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	If CVFault and CVEUSpanInv are set,see Processing Faults later in this instruction. If CVFault and CVEUSpanInv are cleared: 1. CVInitializing is set 2. If PVFault is set, PVSpanInv and SPlimitsInv are cleared. See Processing Faults in this instruction. 3. The PID control algorithm is not executed. 4. The instruction sets CVEU = CVInitValue and CV = corresponding percentage. 5. When CVInitializing and Manual AfterInit are set, the instructions misables auto and cascade/ratio modes. If the current mode is not Override or Hand mode, the instruction changes to manual ode.If ManualAfterInit is cleared, the mode is not changed. CVEu = CVInitValue $CV_{n-1} = CV = \frac{CVEU - CVEUMin}{CVEUMax - CVEUMin} \times 100$ CVOper = CV
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

When CVInitReq is set, or during instruction first scan, or on a set to cleared transition of CVFault (bad to good), the instruction initializes the CVEU and CV outputs to the value of CVInitValue. If the timing mode is not oversample and EnableIn transitions from cleared to set, the instruction initializes the CVEU and CV values. CVInitialization is cleared after the initialization and when CVInitReq is cleared.

The CVInitValue normally comes from the analog output's readback value. The CVInitReq value normally comes from the "In Hold" status bit on the analog output controlled by CVEU. The initialization procedure is performed to avoid a bump at startup in the output signal being sent to the field device.

When using cascaded PID loops, the primary PID loop can be initialized when the secondary loop is initialized or when the secondary loop leaves the Cascade/Ratio mode. In this case, move the state of the InitPrimary output and SP output from the secondary loop to the CVInitReq input and CVInitValue input on the primary loop.

The instruction does not initialize and the CVEU and CV values are not updated if CVFault or CVEUSpanInv is set.

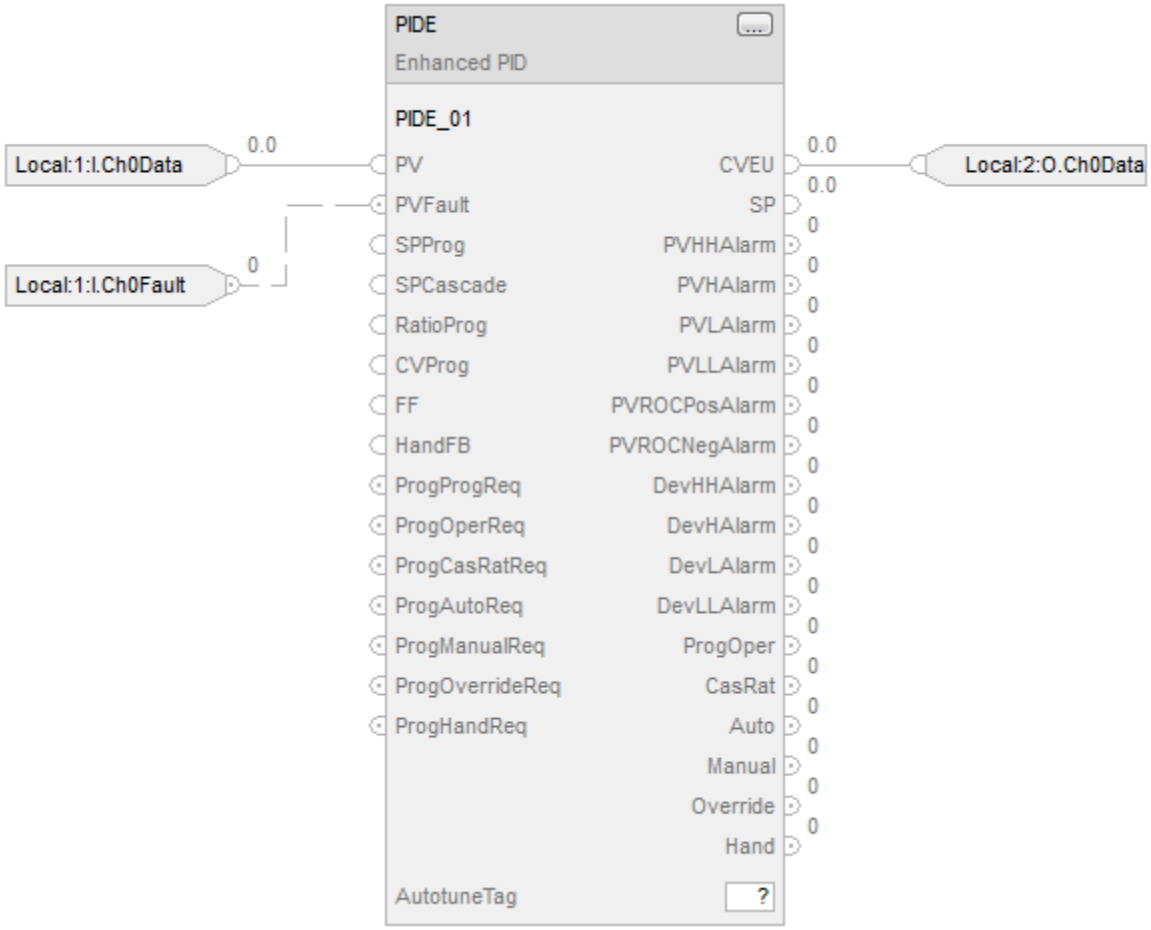
Examples

Example 1

The easiest way to implement a PIDE instruction is to create a function block routine in a program in a periodic task. The default timing mode for the PIDE instruction is periodic. When the PIDE instruction is used in a periodic task and in periodic timing mode, it automatically uses the periodic task's update rate as its delta t update time. All you need to do is wire the process variable analog input into the PV parameter on the PIDE instruction and wire the CVEU out of the PIDE instruction into the controlled variable analog output.

Optionally, you can wire the analog input's fault indicator (if one is available) into the PVFault parameter on the PIDE instruction. This forces the PIDE into Manual mode when the analog input is faulted and stops the PIDE CVEU output from winding up or down when the PV signal is not available.

Function Block



Structured Text

PIDE_01.PV := Local:1:I.Ch0Data;

PIDE_01.PVFault := Local:1:I.Ch0Fault;

PIDE(PIDE_01);

Local:2:)O.Ch0Data :=PIDE_01.CVEU;

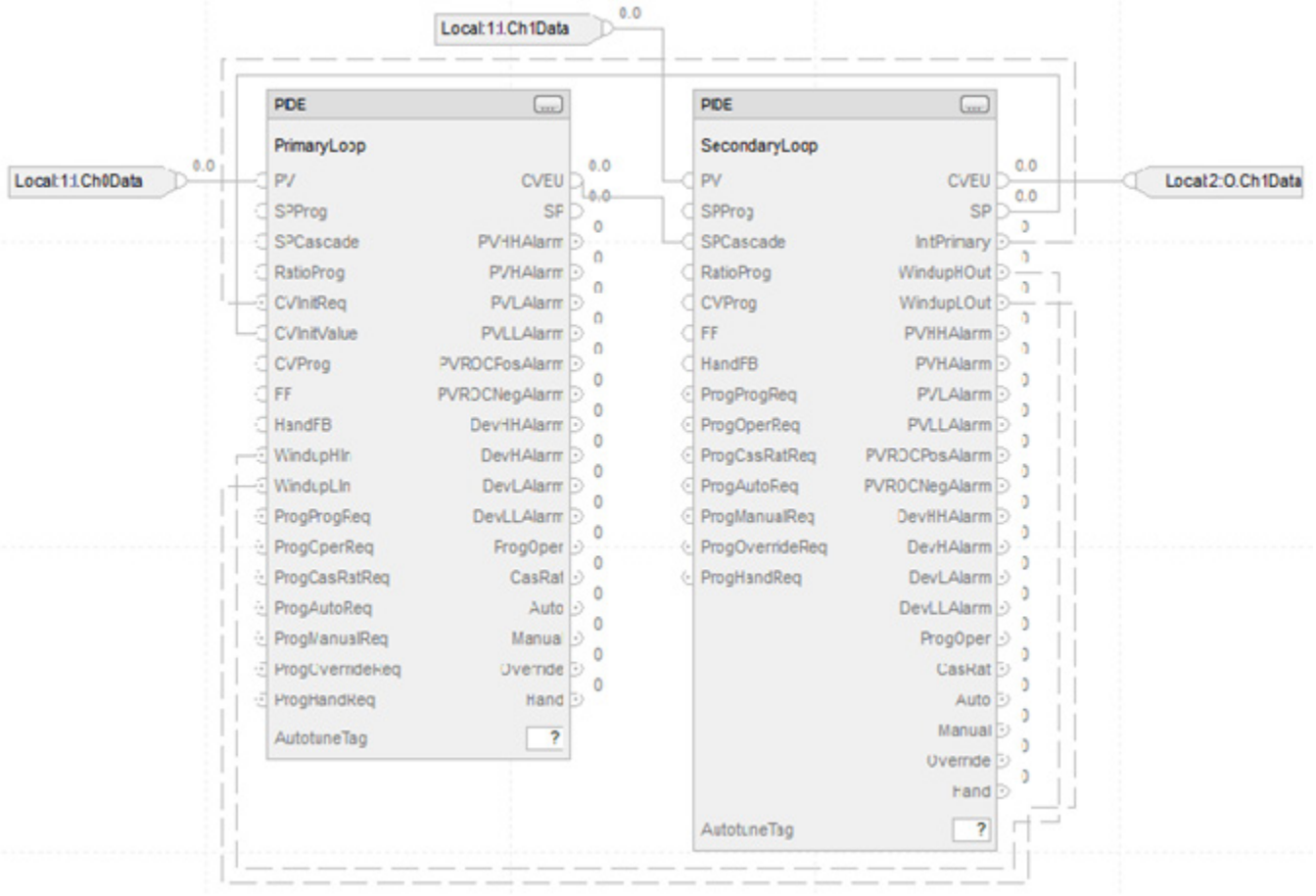
Example 2

Cascade control is useful when externally-caused upsets to the controlled variable occur often, which then cause upsets to the process variable you are trying to control. For example, try to control the temperature of liquid in a tank by varying the amount of steam fed into a heating jacket around the tank. If the steam flow suddenly drops because of an upstream process, the temperature of the liquid in the tank eventually drops and the PIDE

instruction then opens the steam valve to compensate for the drop in temperature.

In this example, a cascaded loop provides better control by opening the steam valve when the steam flow drops before the liquid temperature in the tank drops. To implement a cascaded loop, use a PIDE instruction to control the steam valve opening based on a process variable signal from a steam flow transmitter. This is the secondary loop of the cascaded pair. A second PIDE instruction (called the primary loop) uses the liquid temperature as a process variable and sends its CV output into the setpoint of the secondary loop. In this manner, the primary temperature loop asks for a certain amount of steam flow from the secondary steam flow loop. The steam flow loop is then responsible for providing the amount of steam requested by the temperature loop in order to maintain a constant liquid temperature.

Function Block



Structured Text

```
PrimaryLoop.PV := Local:1:I.CHoData;
PrimaryLoop.CVInitReq := SecondaryLoop.InitPrimary;
```

```
PrimaryLoop.CVInitValue := SecondaryLoop.SP;
PrimaryLoop.WindupHIn := SecondaryLoop.WindupHOut;
PrimaryLoop.WindupLIn := SecondaryLoop.WindupLOut;

PIDE(PrimaryLoop);

SecondaryLoop.PV := Local:1:I.Ch1Data;
SecondaryLoop.SPCascade := PrimaryLoop.CVEU;

PIDE(SecondaryLoop);

Local:2:O.Ch0Data:= SecondaryLoop.CVEU;
```

For a cascaded pair of loops to work correctly, the secondary loop must have a faster process response than the primary loop. This is because the secondary loop's process must be able to compensate for any upsets before these upsets affect the primary loop's process. In this example, if steam flow drops, the steam flow must be able to increase as a result of the secondary controller's action before the liquid temperature is affected.

To set up a pair of cascaded PIDE instructions, set the *AllowCasRat* input parameter in the secondary loop. This allows the secondary loop to be placed into Cascade/Ratio mode. Next, wire the *CVEU* from the primary loop into the *SPCascade* parameter on the secondary loop. The *SPCascade* value is used as the SP on the secondary loop when the secondary loop is placed into Cascade/Ratio mode. The engineering unit range of the CVEU on the primary loop should match the engineering unit range of the PV on the secondary loop. This lets the primary loop scale its 0-100% value of CV into the matching engineering units used for the setpoint on the secondary loop.

The PIDE instruction supports several other features to more effectively support cascade control. Wire the *InitPrimary* output on the secondary loop into the *CVInitReq* input on the primary loop and wire the SP output of the secondary into the *CVInitValue* input on the primary. This sets the CVEU value of the primary loop equal to the SP of the secondary loop when the secondary loop leaves Cascade/Ratio mode. This allows a bumpless transfer when you place the secondary loop back into Cascade/Ratio mode. Also, wire the *WindupHOut* and WindupLOut outputs on the secondary loop into the *WindupHIn* and *WindupLIn* inputs on the primary loop. This causes the primary loop to stop increasing or decreasing, as appropriate, its value of CVEU if the secondary loop hits a SP limit or CV limit and eliminates any windup on the primary loop if these conditions occur.

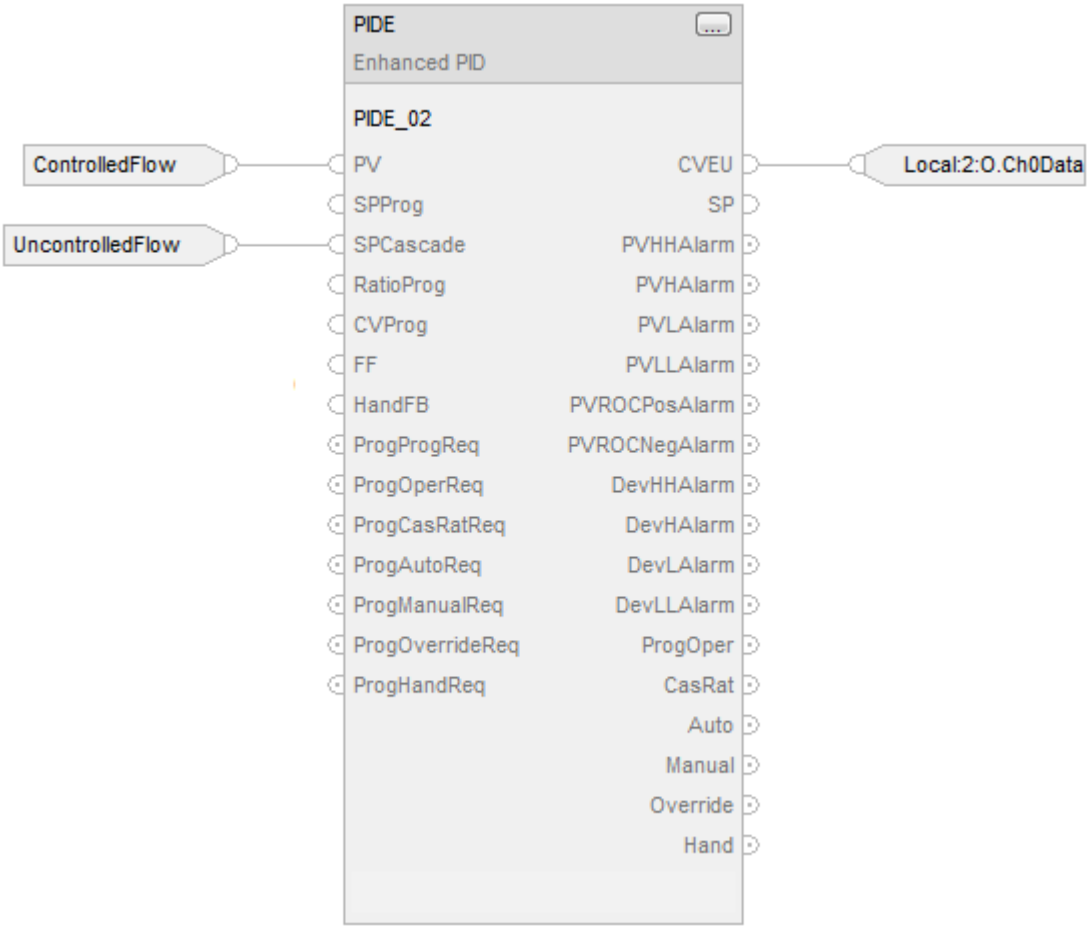
Example 3

Ratio control is typically used to add a fluid in a set proportion to another fluid. For example, if you want to add two reactants (say A and B) to a tank in a constant ratio, and the flow rate of reactant A may change over time because of some upstream process upsets, you can use a ratio controller to

automatically adjust the rate of reactant B addition. In this example, reactant A is often called the "uncontrolled" flow since it is not controlled by the PIDE instruction. Reactant B is then called the "controlled" flow.

To perform ratio control with a PIDE instruction, set the *AllowCasRat* and *UseRatio* input parameters. Wire the uncontrolled flow into the *SPCascade* input parameter. When in Cascade/Ratio mode, the uncontrolled flow is multiplied by either the *RatioOper* (when in Operator control) or the RatioProg (when in Program control) and the resulting value is used by the PIDE instruction as the setpoint.

Function Block



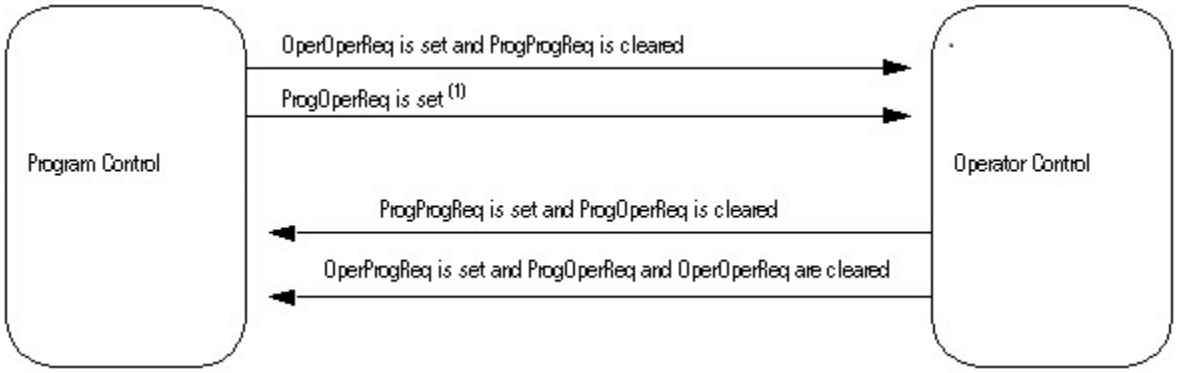
Structured Text

```
PIDE_01.PV := ControlledFlow;
PIDE_01.SPCascade := UncontrolledFlow;

PIDE(PIDE_01);

Local:2:O.Ch0Data := PIDE_01.CVEU;
```

Switching Between Program Control and Operator Control

The PIDE instruction can be controlled by either a user program or an operator interface. You can change the control mode at any time. Program and Operator control use the same ProgOper output. When ProgOper is set, control is Program; when ProgOper is cleared, control is Operator.

The following diagram shows how the PIDE instruction changes between Program control and Operator control.



(1) The instruction remains in Operator control mode when ProgOperReq is set.

Operating Modes

The PIDE instruction supports the following PID modes.

PID Operating Mode	Description
Cascade/Ratio	While in Cascade/Ratio mode the instruction computes the change in CV. The instruction regulates CV to maintain PV at either the SPCascade value or the SPCascade value multiplied by the Ratio value. SPCascade comes from either the CVEU of a primary PID loop for cascade control or from the "uncontrolled" flow of a ratio-controlled loop. Select Cascade/Ratio mode using either OperCasRatReq or ProgCasRatReq: Set OperCasRatReq to request Cascade/Ratio mode. Ignored when ProgOper, ProgOverrideReq, ProgHandReq, OperAutoReq, or OperManualReq is set, or when AllowCasRat is cleared. Set ProgCasRatReq to request Cascade/Ratio mode. Ignored when ProgOper or AllowCasRat is cleared or when ProgOverrideReq, ProgHandReq, ProgAutoReq, or ProgManualReq is set.
Auto	While in Auto mode the instruction computes the change in CV. The instruction regulates CV to maintain PV at the SP value. If in program control, SP = SPProg; if in Operator control, SP = SPOper. Select Auto mode using either OperAutoReq or ProgAutoReq: Set OperAutoReq to request Auto mode. Ignored when ProgOper, ProgOverrideReq, ProgHandReq, or OperManualReq is set. Set ProgAutoReq to request Auto mode. Ignored when ProgOper is cleared or when ProgOverrideReq, ProgHandReq, or ProgManualReq is set.
Manual	While in Manual mode the instruction does not compute the change in CV. The value of CV is determined by the control. If in Program control, CV = CVProg; if in Operator control, CV = CVOper. Select Manual mode using either OperManualReq or ProgManualReq: Set OperManualReq to request Manual mode. Ignored when ProgOper, ProgOverrideReq, or ProgHandReq is set. Set ProgManualReq to request Manual mode. Ignored when ProgOper is cleared or when ProgOverrideReq or ProgHandReq is set.

PID Operating Mode	Description
Override	While in Override mode the instruction does not compute the change in CV. CV = CVOverride, regardless of the control mode. Override mode is typically used to set a "safe state" for the PID loop. Select Override mode using ProgOverrideReq: Set ProgOverrideReq to request Override mode. Ignored when ProgHandReq is cleared.
Hand	While in Hand mode the PID algorithm does not compute the change in CV. CV = HandFB, regardless of the control mode. Hand mode is typically used to indicate that control of the final control element was taken over by a field hand/auto station. Select Hand mode using ProgHandReq: Set ProgHandReq to request hand mode. This value is usually read as a digital input from a hand/auto station.

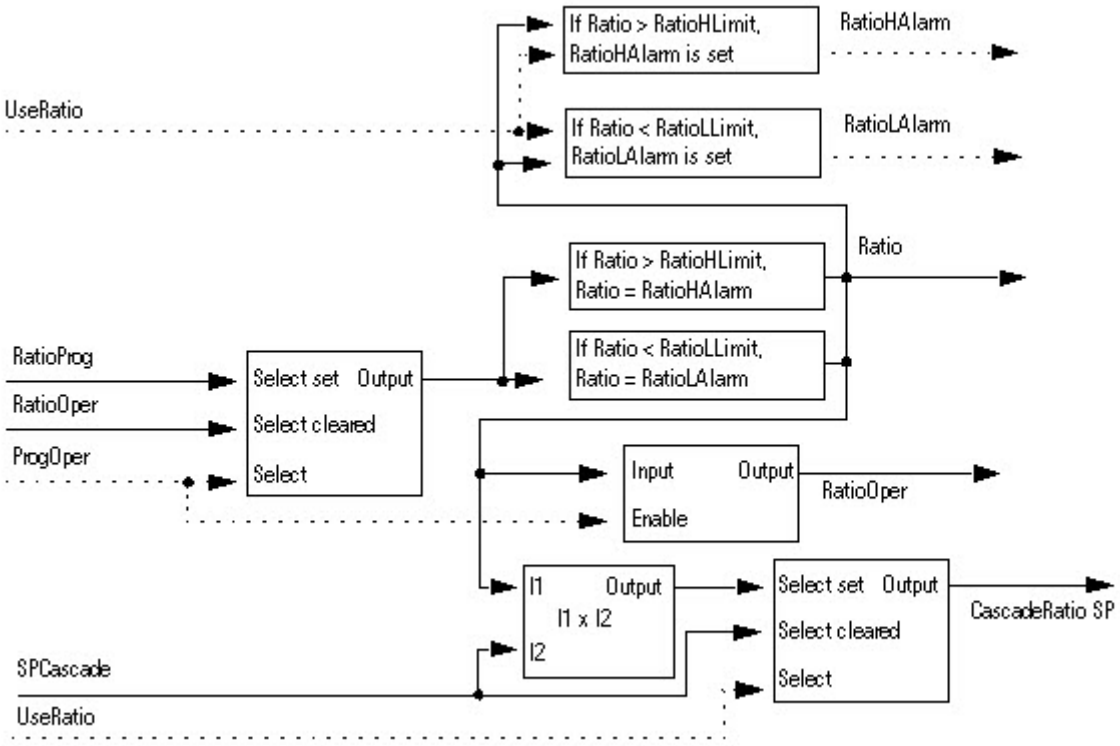
The Cascade/Ratio, Auto, and Manual modes can be controlled by a user program when in Program control or by an operator interface when in Operator control. The Override and Hand modes have a mode request input that can only be controlled by a user program; these inputs operate in both Program and Operator control.

Selecting the Setpoint

Once the instruction determines program or operator control and the PID mode, the instruction can obtain the proper SP value. You can select the cascade/ratio SP or the current SP.

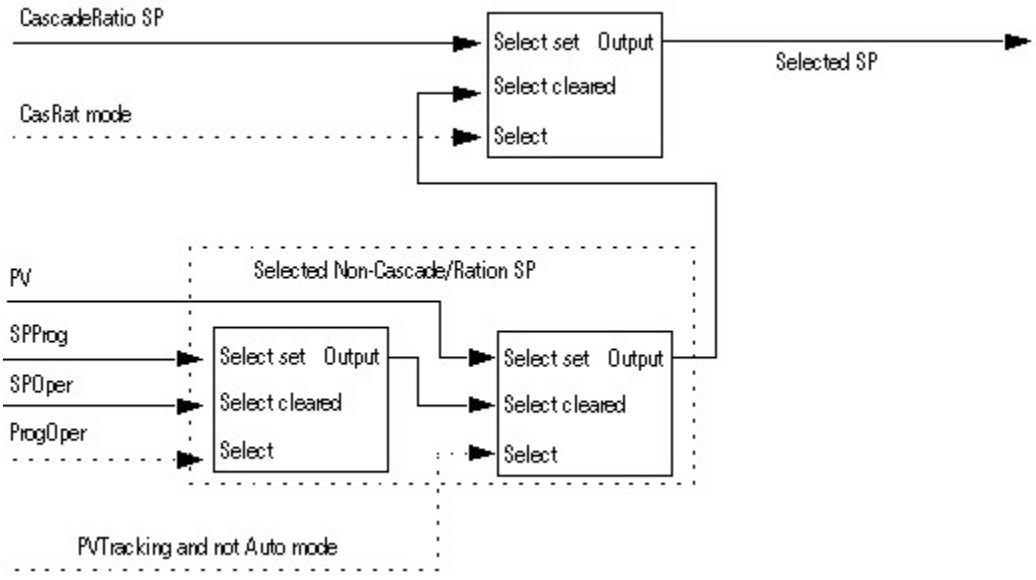
Cascade/Ratio SP

The cascade/ratio SP is based on the UseRatio and ProgOper values.



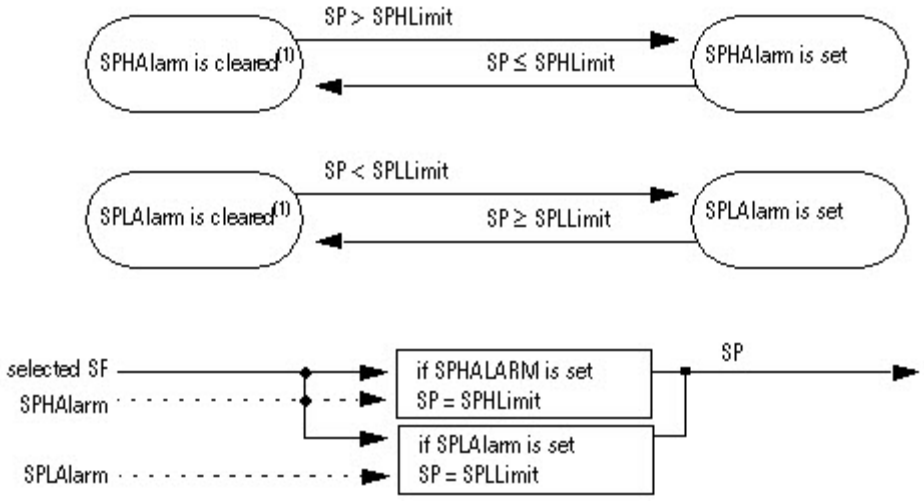
Current SP

The current SP is based on the Cascade/Ratio mode, the PVTracking value, auto mode, and the ProgOper value.



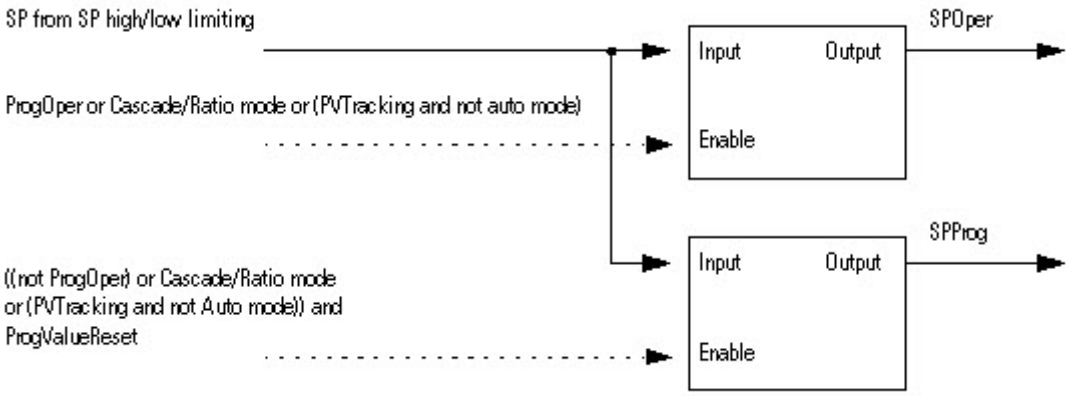
SP High/Low Limiting

The high-to-low alarming algorithm compares SP to the SPHLimit and SPLLimit alarm limits. SPHLimit cannot be greater than PVEUMax and SPLLimit cannot be less than PVEUMin.



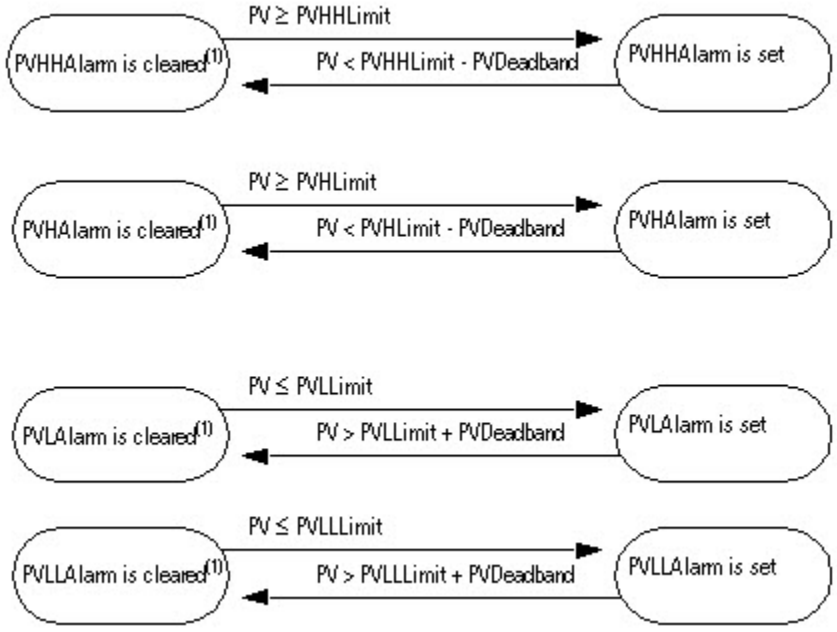
Updating the SPOper and SPProg Values

The PIDE instruction makes SPOper = SP or SPProg = SP to obtain bumpless control switching between Program and Operator control or when switching from Cascade/Ratio mode.



PV High/Low Alarming

The high-high to low-low alarming algorithm compares PV to the PV alarm limits and the PV alarm limits plus or minus the PV alarm deadband



(1) During instruction first scan, the instruction clears all the PV alarm outputs. The instruction also clears the PV alarm outputs and disables the alarming algorithm when PVFaulted is set.

PV Rate-of-Change Alarming

PV rate-of-change (ROC) alarming compares the change in the value of PV over the PVROCPeriod against the PV positive and negative rate-of-change limits. The PVROCPeriod provides a type of deadband for the rate-of-change alarm. For example, if you use a ROC alarm limit of 2°F/second with a period of execution of 100 ms, and an analog input module with a resolution of 1°F, then every time the input value changes, a ROC alarm is generated because the instruction sees a rate of 10°F/second. However, by entering a PVROCPeriod of at least 1 sec, the ROC alarm is only generated if the rate truly exceeds the 2°F/second limit.

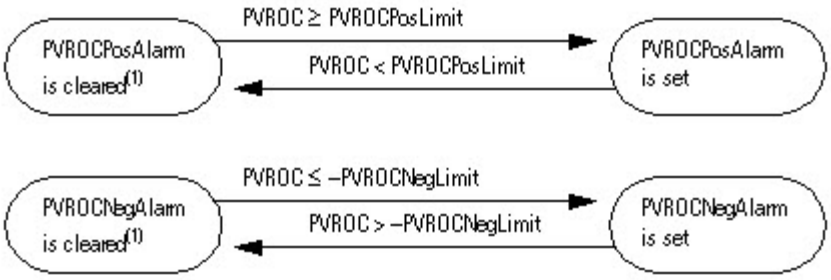
The ROC calculation is only performed when the PVROCPeriod has expired. The rate-of-change is calculated as:

ElapsedROCPeriod = ElapsedROCPeriod + ElapsedTimeSinceLastExecution

If ElapsedROCPeriod ≥ PVROCPeriod then:

This value:	Is:
PVROC	$\frac{PV_n - PVROC_{n-1}}{PVROCPeriod}$
$PVROC_{N-1}$	$PVROC_{N-1} = PV$
ElapsedROCPeriod	ElapsedROCPeriod = 0

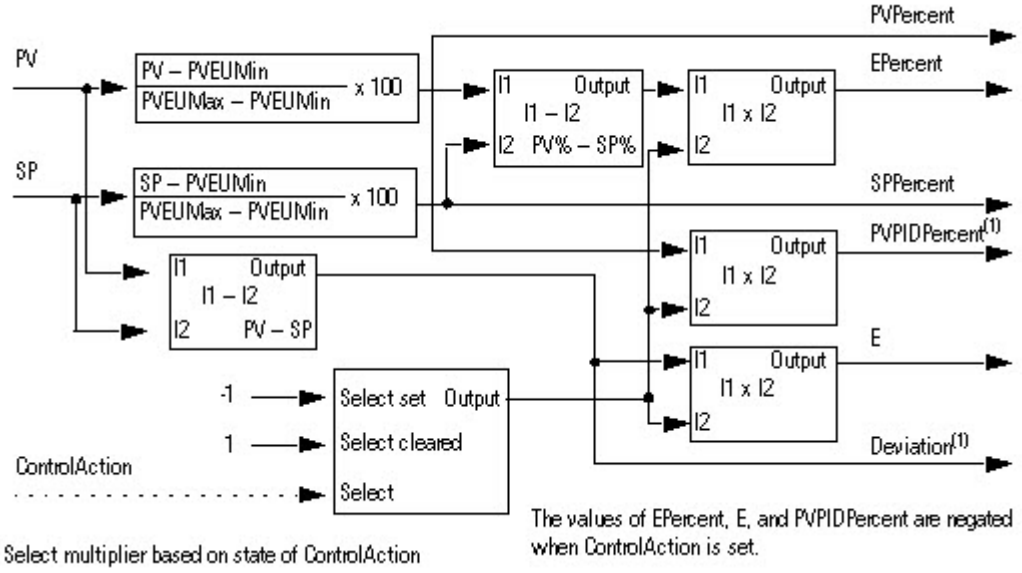
Once PVROC has been calculated, the PV ROC alarms are determined as follows:



(1) During instruction first scan, the instruction clears the PV ROC alarm outputs. The instruction also clears the PVROC alarm outputs and disables the PV ROC alarming algorithm when PVFaulted is set.

Converting the PV and SP Values to Percent

The instruction converts PV and SP to a percent and calculates the error before performing the PID control algorithm. The error is the difference between the PV and SP values. When ControlAction is set, the values of EPercent, E, and PVPIDPercent are negated before being used by the PID algorithm.

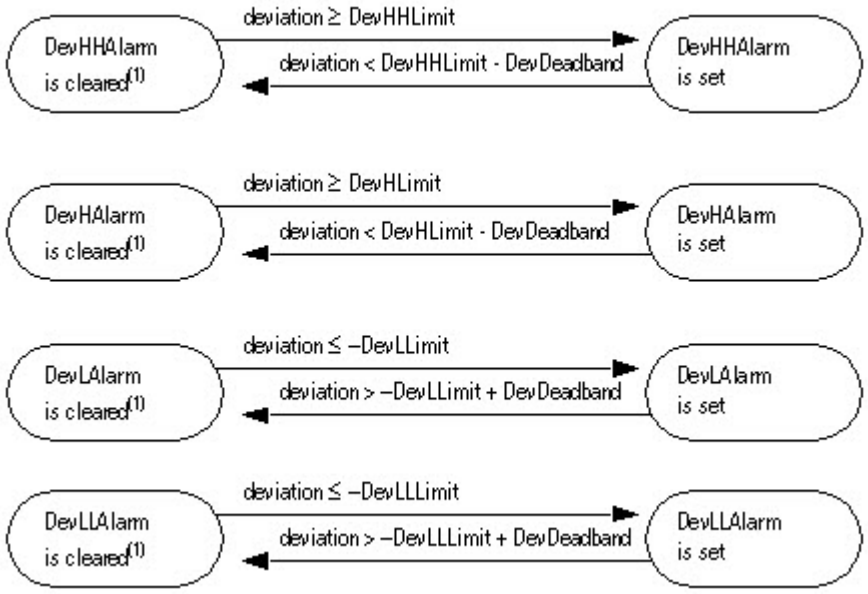


(1) PVPIDPercent and Deviation are internal parameters used by the PID control algorithm.

Deviation High/Low Alarming

Deviation is the difference in value between the process variable (PV) and setpoint (SP). Deviation alarming alerts the operator to a discrepancy between the process variable and the setpoint value.

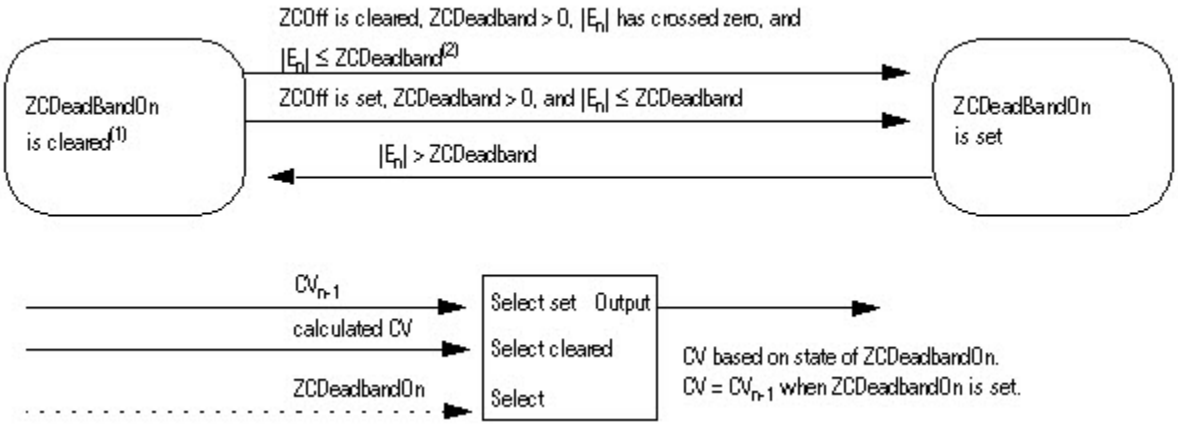
The high-high to low-low alarming algorithm compares the deviation to deviation alarm limits and the deviation alarm limits plus or minus the deadband.



(1) During instruction first scan, the instruction clears the deviation alarm outputs. The instruction also clears the deviation alarm outputs and disables the alarming algorithm when PVFaulted or PVSpanInv is set.

Zero Crossing Deadband Control

You can limit CV such that its value does not change when error remains within the range specified by ZCDeadband (| E | ≤ ZCDeadband).



(1) When ZCOff is cleared, ZCDeadband > 0, error has crossed zero for the first time, (i.e. En >= 0 and En-1 < 0 or when En <= 0 and En-1 > 0), and | En | <= ZCDeadband, the instruction sets ZCDeadbandOn.

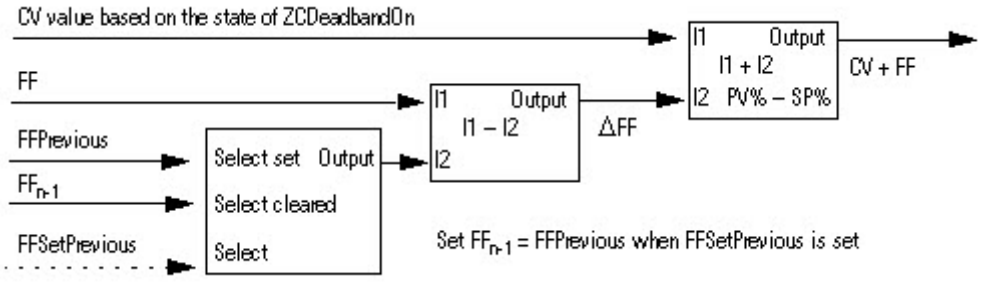
(2) On the transition to Auto or Cascade/Ratio mode, the instruction sets En-1 = En.

The instruction disables the zero crossing algorithm and clears ZCDeadband under these conditions:

- during instruction first scan
- ZCDeadband ≤ 0
- Auto or Cascade/Ratio is not the current mode
- PVFaulted is set
- PVSpanInv is set

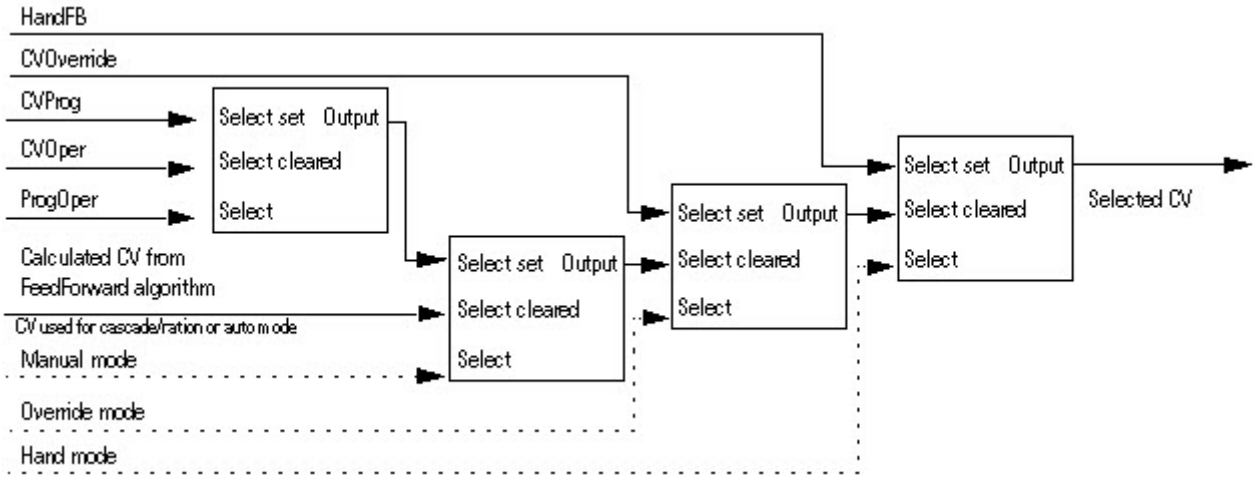
Feedforward Control

Compute CV by summing CV from the zero crossing algorithm with ΔFF. The value of ΔFF = FF – FFn-1. When FFSetPrevious is set, FFn-1=FFPrevious. This lets you preset FFn-1 to a specified value before the instruction calculates the value of ΔFF.



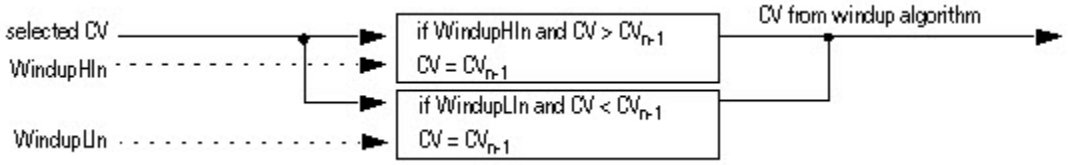
Selecting the Control Variable

Once the PID algorithm has been executed, select the CV based on program or operator control and the current PID mode.



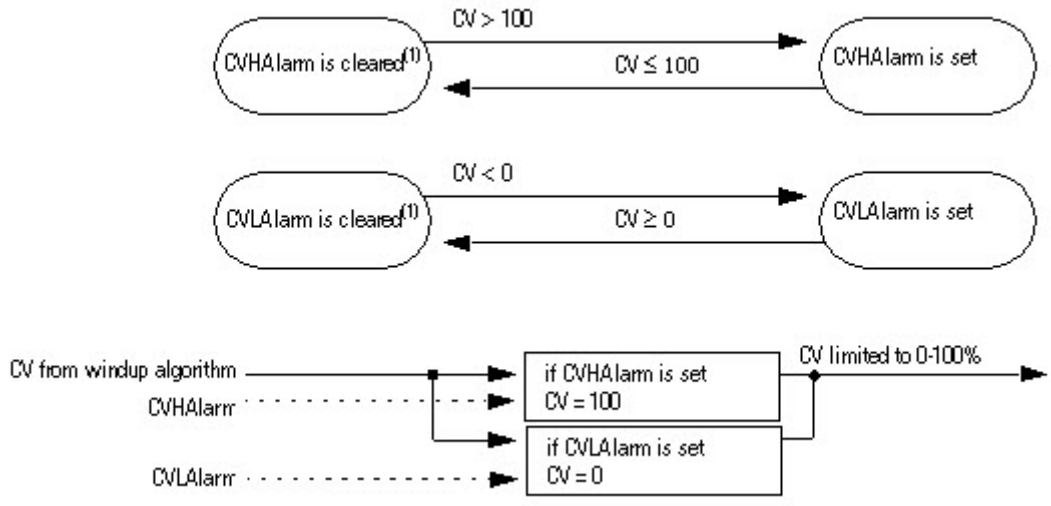
CV Windup Limiting

Limit the CV such that its value cannot increase when WindupHIn is set or decrease when WindupLIn is set. These inputs are typically the WindupHOut or WindupLOut outputs from a secondary loop. The WindupHIn and WindupLIn inputs are ignored if CVInitializing, CVFault, or CVEUSpanInv is set.



CV Percent Limiting

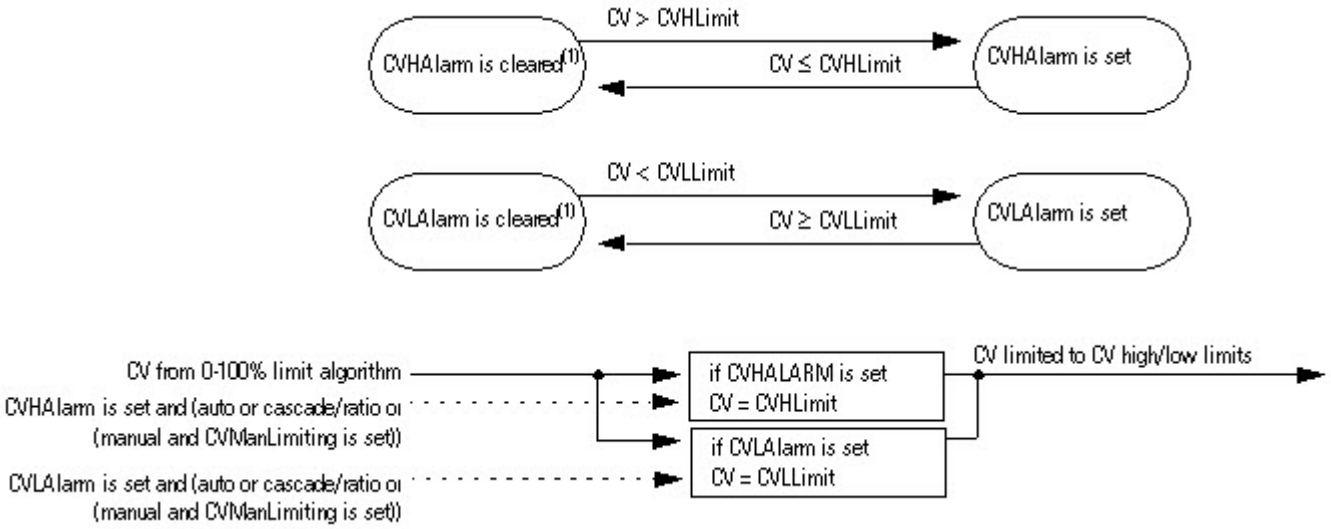
The following diagram illustrates how the instruction determines CV percent limiting.



(1) During instruction first scan, the instruction clears the alarm outputs.

CV High/Low Limiting

The instruction always performs alarming based on CVHLimit and CVLLimit. Limit CV by CVHLimit and CVLLimit when in auto or cascade/ratio mode. When in manual mode, limit CV by CVHLimit and CVLLimit when CVManLimiting is set. Otherwise limit CV by 0 and 100 percent.



(1) During instruction first scan, the instruction clears the alarm outputs.

CV Rate-of-Change Limiting

The PIDE instruction limits the rate-of-change of CV when in Auto or Cascade/Ratio mode or when in Manual mode and CVManLimiting is set. A value of zero disables CV rate-of-change limiting.

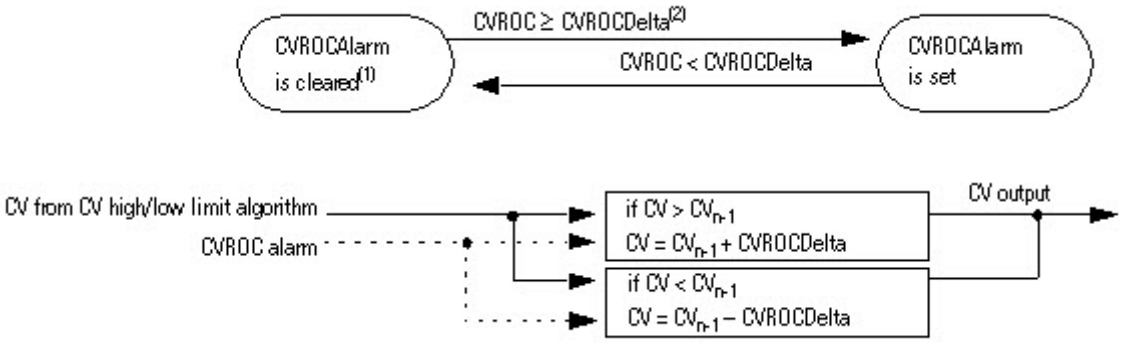
The CV rate-of-change is calculated as:

$$CVROC = |CV_n - CV_{n-1}|$$

$$CVROCDelta = CVROCLimit \times DeltaT$$

where DeltaT is in seconds.

Once CV rate-of-change has been calculated, the CV rate-of-change alarms are determined as follows:

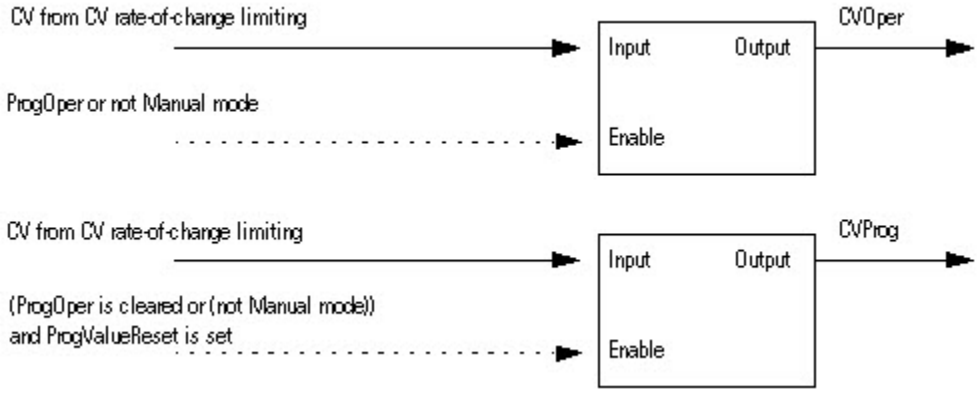


(1) During instruction first scan, the instruction clears the alarm output. The instruction also clears the alarm output and disables the CV rate-of-change algorithm when CVInitializing is set.

(2) When in Auto or Cascade/Ratio mode or when in Manual mode and CVManLimiting is set, the instruction limits the change of CV.

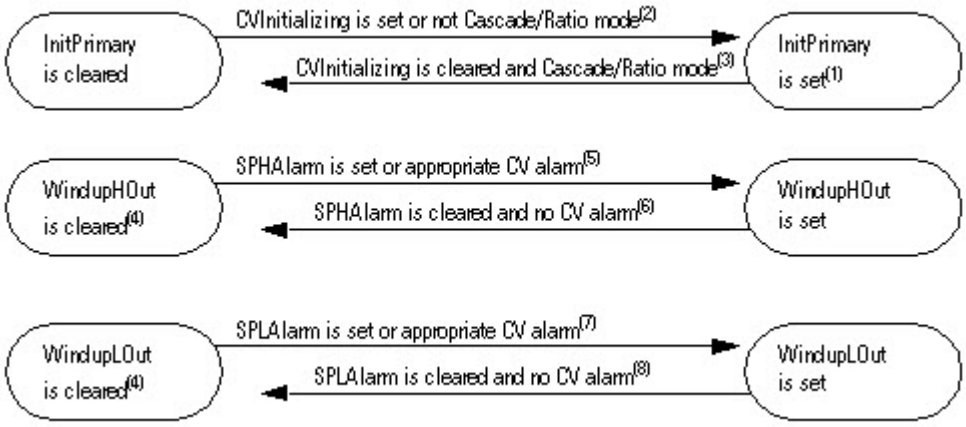
Updating the CVOper and CVProg Values

If not in the Operator Manual mode, the PIDE instruction sets CVOper = CV. This obtains bumpless mode switching from any control to the Operator Manual mode.



Primary Loop Control

Primary loop control is typically used by a primary PID loop to obtain bumpless switching and anti-reset windup when using Cascade/Ratio mode. The primary loop control includes the initialize primary loop output and the anti-reset windup outputs. The InitPrimary output is typically used by the CVInitReq input of a primary PID loop. The windup outputs are typically used by the windup inputs of a primary loop to limit the windup of its CV output.



(1) During instruction first scan, the instruction sets InitPrimary.

(2) When CVInitializing is set or when not in Cascade/Ratio mode the instruction sets InitPrimary.

(3) When CVInitializing is cleared and in Cascade/Ratio mode, the instruction clears InitPrimary.

(4) During instruction first scan, the instruction clears the windup outputs. The instruction also clears the windup outputs and disables the CV windup algorithm when CVInitializing is set or if either CVFaulted or CVEUSpanInv is set.

(5) The instruction sets WindupHOut when SPHAlarm is set, or when ControlAction is cleared and CVHAlarm is set, or when ControlAction is set and CVLAlarm is set.

The SP and CV limits operate independently. A SP high limit does not prevent CV from increasing in value. Likewise, a CV high or low limit does not prevent SP from increasing in value.

(6) The instruction clears WindupHOut when SPHAlarm is cleared, and not (ControlAction is cleared and CVHAlarm is set), and not (ControlAction is set and CVLAlarm is set).

(7) The instruction sets WindupLOut when SPLAlarm is set, or when ControlAction is cleared and CVLAlarm is set, or when ControlAction is set and CVHAlarm is set.

The SP and CV limits operate independently. A SP low limit does not prevent CV from increasing in value. likewise a CV low or high limit does not prevent SP from increasing in value.

(8) The instruction clears WindupLOut when SPLAlarm is cleared and not (ControlAction is cleared and CVLAlarm is set) and not (ControlAction is set and CVHAlarm is set).

Processing Faults

The following table describes how the instruction handles execution faults:

Fault Condition	Action
CVFaulted is true or CVEUSpanInv is true	Instruction is not initialized, CVInitializing is cleared to false Compute PV and SP percent, calculate error, update internal parameters for EPercent and PVPIDPercent PID control algorithm is not executed Disable the Auto and Cascade/Ratio modes. If Override or Hand is not the current mode, set to Manual mode. Set CV to value determined by Program or Operator control and mode (Manual, Override, or Hand).
PVFaulted is true	Disable the Auto and Cascade/Ratio modes. If Override or Hand is not the current mode, set to Manual mode PV high-low, PV rate-of-change, and deviation high-low alarm outputs are cleared to false PID control algorithm is not executed Set CV to value by determined by Program or Operator control and mode (Manual, Override, or Hand).
PVSpanInv is true or SPLimitsInv is true	Disable the Auto and Cascade/Ratio modes. If Override or Hand is not the current mode, set to Manual mode Do not compute PV and SP percent PID control algorithm is not executed Set CV to value by determined by Program or Operator control and mode (Manual, Override, or Hand).
RatioLimitsInv is true and CasRat is true and UseRatio is true	If not already in Hand or Override, set to Manual model Disable the Cascade/Ratio mode Set CV to value determined by Program or Operator control and mode (Manual, Override, or Hand).
TimingModeInv is true or RTSTimeStampInv is true or DeltaTInv is true	If not already in Hand or Override, set to Manual mode

See also

Function Block Attributes on page 1043

Function Block Faceplate Controls on page 1095

Common Attributes on page 1083

Structured Text Syntax on page 1057

Position Proportional (POSP)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

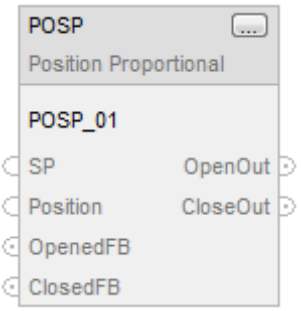
The POSP instruction opens or closes a device by pulsing open or close contacts at a user defined cycle time with a pulse width proportional to the difference between the desired and actual positions.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

POSP(POSP_tag)

Operands

Function Block

Operand	Type	Format	Description
POSP tag	POSITION_PROP	Structure	POSP structure

Structured Text

Operand	Type	Format	Description
block tag	POSITION_PROP	Structure	POSP structure

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

POSITION_PROP Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.

Input Parameter	Data Type	Description
SP	REAL	Setpoint. This is the desired value for the position. This value must use the same engineering units as Position. Valid = any float Default = 0.0
Position	REAL	Position feedback. This analog input comes from the position feedback from the device. Valid = any float Default = 0.0
OpenedFB	BOOL	Opened feedback. This input signals when the device is fully opened. When true, the open output is not allowed to turn on. Default is false.
ClosedFB	BOOL	Closed feedback. This input signals when the device is fully closed. When true, the close output is not allowed to turn on. Default is false.
PositionEUMax	REAL	Maximum scaled value of Position and SP. Valid = any float Default = 100.0
PositionEUMin	REAL	Minimum scaled value of Position and SP. Valid = any float Default = 0.0
CycleTime	REAL	Period of the output pulse in seconds. A value of zero clears both OpenOut and CloseOut. If this value is invalid, the instruction assumes a value of zero and sets the appropriate bit in Status. Valid = any positive float Default = 0.0
OpenRate	REAL	Open rate of the device in %/second. A value of zero clears OpenOut. If this value is invalid, the instruction assumes a value of zero and sets the appropriate bit in Status. Valid = any positive float Default = 0.0
CloseRate	REAL	Close rate of the device in %/second. A value of zero clears CloseOut. If this value is invalid, the instruction assumes a value of zero and sets the appropriate bit in Status. Valid = any positive float Default = 0.0
MaxOnTime	REAL	Maximum time in seconds that an open or close pulse can be on. If OpenTime or CloseTime is calculated to be larger than this value, they are limited to this value. If this value is invalid, the instruction assumes a value of CycleTime and sets the appropriate bit in Status. Valid = 0.0 to CycleTime Default = CycleTime

Input Parameter	Data Type	Description
MinOnTime	REAL	Minimum time in seconds that an open or close pulse can be on. If OpenTime or CloseTime is calculated to be less than this value, they are set to zero. If this value is invalid, the instruction assumes a value of zero and sets the appropriate bit in Status. Valid = 0.0 to MaxOnTime Default = 0.0
Deadtime	REAL	Additional pulse time in seconds to overcome friction in the device. Deadtime is added to the OpenTime or CloseTime when the device changes direction or is stopped. If this value is invalid, the instruction sets the appropriate bit in Status and uses a value of Deadtime = 0.0. Valid = 0.0 to MaxOnTime Default = 0.0

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if PositionPercent overflows.
OpenOut	BOOL	This output is pulsed to open the device.
CloseOut	BOOL	This output is pulsed to close the device.
PositionPercent	REAL	Position feedback is expressed as percent of the Position span.
SPPercent	REAL	Setpoint is expressed as percent of the Position span.
OpenTime	REAL	Pulse time in seconds of OpenOutput for the current cycle.
CloseTime	REAL	Pulse time in seconds of CloseOutput for the current cycle.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
CycleTimeInv (Status.1)	BOOL	Invalid CycleTime value. The instruction uses zero.
OpenRateInv (Status.2)	BOOL	Invalid OpenRate value. The instruction uses zero.
CloseRateInv (Status.3)	BOOL	Invalid CloseRate value. The instruction uses zero.
MaxOnTimeInv (Status.4)	BOOL	Invalid MaxOnTime value. The instruction uses the CycleTime value.
MinOnTimeInv (Status.5)	BOOL	Invalid MinOnTime value. The instruction uses zero.
DeadtimeInv (Status.6)	BOOL	Invalid Deadtime value. The instruction uses zero.
PositionPctInv (Status.7)	BOOL	The calculated PositionPercent value is out of range.
SPPercentInv (Status.8)	BOOL	The calculated SPPercent value is out of range.

Output Parameter	Data Type	Description
PositionSpanInv (Status.9)	BOOL	PositionEUMax = PositionEUMin.

Description

The POSP instruction usually receives the desired position setpoint from a PID instruction output.

Scaling the Position and Setpoint Values

The PositionPercent and SPPercent outputs are updated each time the instruction is executed. If either of these values is out of range (less than 0% or greater than 100%), the appropriate bit in Status is set, but the values are not limited. The instruction uses these formulas to calculate whether the values are in range:

$$PositionPercent = \frac{Position - PositionEUMin}{PositionEUMax - PositionEUMin} \times 100$$

$$SPPercent = \frac{SP - PositionEUMin}{PositionEUMax - PositionEUMin} \times 100$$

How the POSP Instruction Uses the Internal Cycle Timer

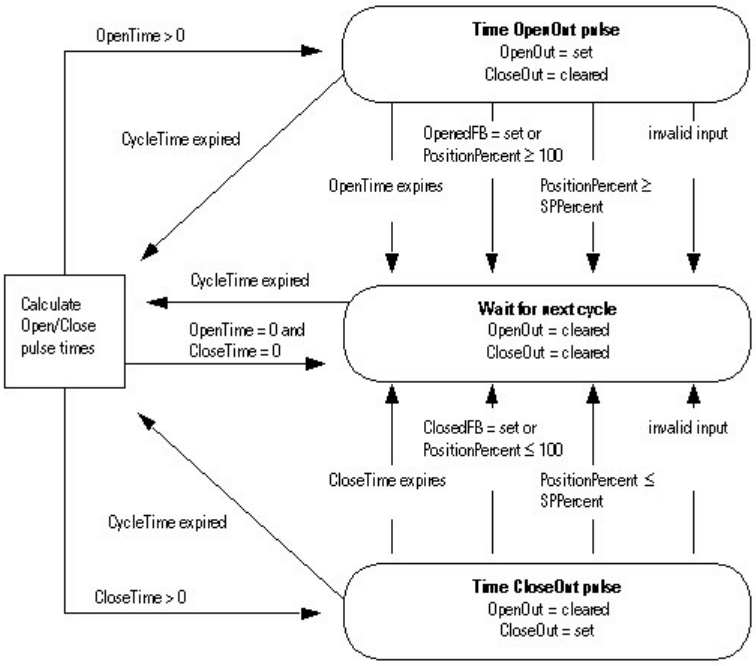
The instruction uses CycleTime to determine how often to recalculate the duration of Open and Close output pulses. An internal timer is maintained and updated by DeltaT. DeltaT is the elapsed time since the instruction last executed. Whenever the internal timer equals or exceeds the programmed CycleTime (cycle time expires) the Open and Close outputs are recalculated.

You can change the CycleTime at any time.

If CycleTime = 0, the internal timer is cleared to 0, OpenOut is cleared to false and CloseOut is cleared to false.

Producing Output Pulses

The following diagram shows the three primary states of the POSP instruction.



Calculating Open and Close Pulse Times

OpenOut is pulsed whenever SP > Position feedback. When this occurs, the instruction sets CloseTime = 0 and the duration for which OpenOut is to be turned on is calculated as:

$$OpenTime = \frac{SPPercent - PositionPercent}{OpenRate}$$

If OpenTimen-1 < CycleTime, then add Deadtime to OpenTime.

If OpenTime > MaxOnTime, then limit to MaxOnTime.

If OpenTime < MinOnTime, then set OpenTime = 0.

If any of the following conditions exist, OpenOut is not pulsed and OpenTime = 0.

OpenFB is true or PositionPercent ≥ 100

CycleTime = 0

OpenRate = 0

SPPercent is invalid

The CloseOut is pulsed whenever SP < Position feedback. When this occurs, the instruction sets OpenTime = 0 and the duration for which CloseOut is to be turned on is calculated as:

$$CloseTime = \frac{PositionPercent - SPPercent}{CloseRate}$$

If CloseTimen-1 < CycleTime, then add Deadtime to CloseTime.

If CloseTime > MaxOnTime, then limit to MaxOnTime.

If CloseTime < MinOnTime, then set CloseTime to 0.

If any of the following conditions exist, CloseOut will not be pulsed and CloseTime will be cleared to 0.0.

ClosedFB is true or PositionPercent ≤ 0

CycleTime = 0

CloseRate = 0

SPPercent is invalid

OpenOut and CloseOut will not be pulsed if SPPercent equals PositionPercent. Both OpenTime and CloseTime will be cleared to false.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	OpenTime and CloseTime are cleared to 0.0.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

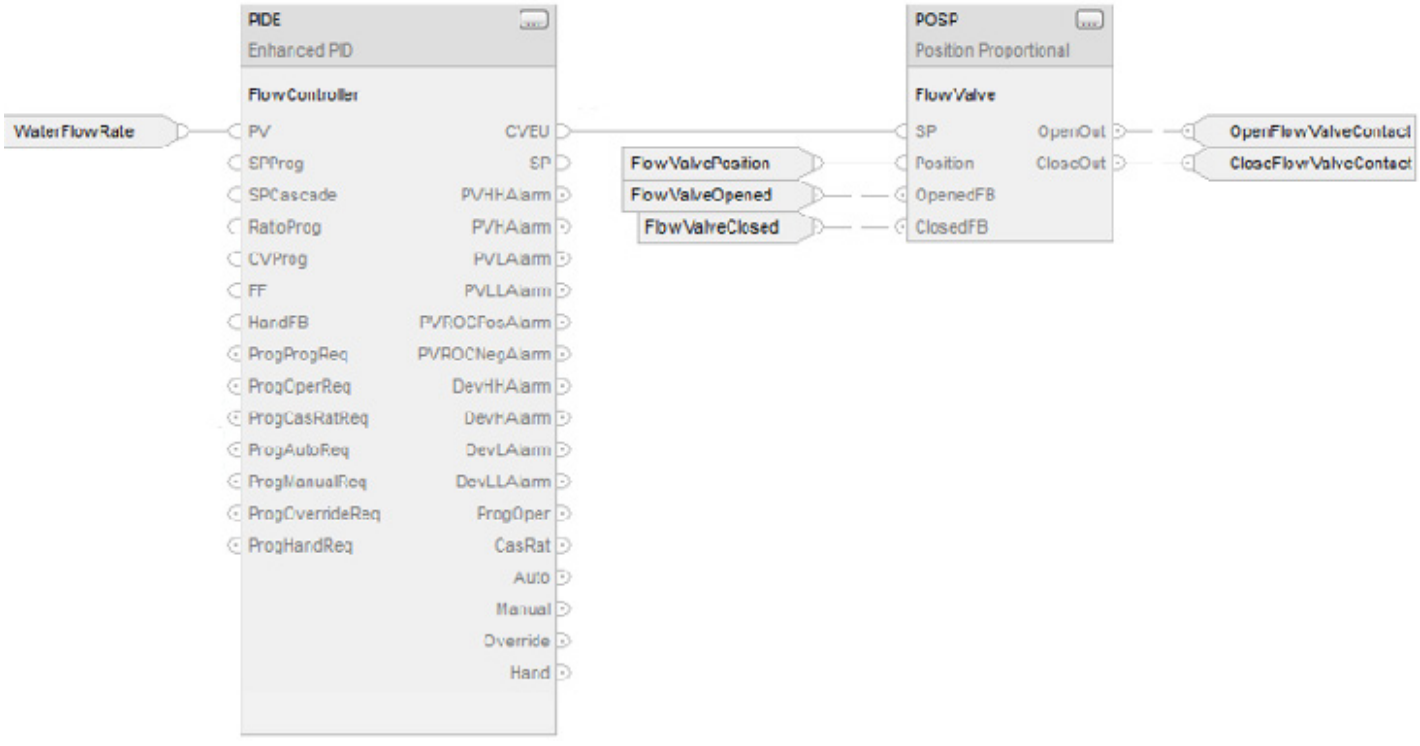
Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Examples

Example 1

In this example, the POSP instruction opens or closes a motor-operated valve based on the CVEU output of the PIDE instruction. The actual valve position is wired into the Position input and optional limit switches, which show if the valve is fully opened or closed, are wired into the OpenedFB and ClosedFB inputs. The OpenOut and CloseOut outputs are wired to the open and close contacts on the motor-operated valve.

Function Block



Structured Text

```
FlowController.PV := WaterFlowRate;
PIDE(FlowController);

FlowValve.SP := FlowController.CVEU;
FlowValve.Position := FlowValvePosition;
FlowValve.OpenedFB := FlowValveOpened;
FlowValve.ClosedFB := FlowValveClosed;
POSP(FlowValve);

OpenFlowValveContact := FlowValve.OpenOut;
CloseFlowValveContact := FlowValve.CloseOut;
```

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Ramp/Soak (RMPS)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

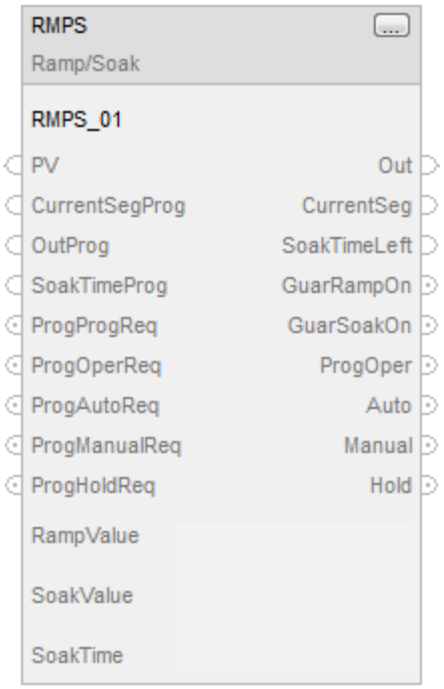
The RMPS instruction provides for a number of segments of alternating ramp and soak periods.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

RMPS(RMPS_tag,RampValue,SoakValue,SoakTime);

Operands

Function Block

Operand	Type	Format	Description
RMPS tag	RAMP_SOAK	structure	RMPS structure
RampValue	REAL	array	Ramp Value array. Enter a ramp value for each segment (0 to NumberOfSegs-1). Ramp values are entered as time in minutes or as a rate in units/minute. The TimeRate parameter reflects which method is used to specify the ramp. If a ramp value is invalid, the instruction sets the appropriate bit in Status and changes to Operator Manual or Program Hold mode. The array must be at least as large as NumberOfSegs. Valid = 0.0 to maximum positive float

Operand	Type	Format	Description
SoakValue	REAL	array	Soak Value array. Enter a soak value for each segment (0 to NumberOfSegs-1). The array must be at least as large as NumberOfSegs. Valid = any float
SoakTime	REAL	array	Soak Time array. Enter a soak time for each segment (0 to NumberOfSegs-1). Soak times are entered in minutes. If a soak value is invalid, the instruction sets the appropriate bit in Status and changes to Operator Manual or Program Hold mode. The array must be at least as large as NumberOfSegs. Valid = 0.0 to maximum positive float

Structured Text

Operand	Type	Format	Description
RMPS tag	RAMP_SOAK	structure	RMPS structure
RampValue	REAL	array	Ramp Value array. Enter a ramp value for each segment (0 to NumberOfSegs-1). Ramp values are entered as time in minutes or as a rate in units/minute. The TimeRate parameter reflects which method is used to specify the ramp. If a ramp value is invalid, the instruction sets the appropriate bit in Status and changes to Operator Manual or Program Hold mode. The array must be at least as large as NumberOfSegs. Valid = 0.0 to maximum positive float
SoakValue	REAL	array	Soak Value array. Enter a soak value for each segment (0 to NumberOfSegs-1). The array must be at least as large as NumberOfSegs. Valid = any float
SoakTime	REAL	array	Soak Time array. Enter a soak time for each segment (0 to NumberOfSegs-1). Soak times are entered in minutes. If a soak value is invalid, the instruction sets the appropriate bit in Status and changes to Operator Manual or Program Hold mode. The array must be at least as large as NumberOfSegs. Valid = 0.0 to maximum positive float

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

RMPS Structure

Specify a unique RMPS structure for each instruction.

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
PV	REAL	The scaled analog temperature signal input to the instruction. Valid = any float Default = 0.0
PVFault	BOOL	Bad health indicator of PV. If true, the input is invalid, the instruction is placed in Program Hold or Operator Manual mode, and the instruction sets the appropriate bit in Status. Default is false.
NumberOfSegs	DINT	Number of segments. Specify the number of ramp/soak segments used by the instruction. The arrays for RampValue, SoakValue, and SoakTime must be at least as large as NumberOfSegs. If this value is invalid, the instruction is placed into Operator Manual or Program Hold mode and the instruction sets the appropriate bit in Status. Valid = 1 to (minimum size of RampValue, SoakValue, or SoakTime arrays) Default = 1
ManHoldAftInit	BOOL	Manual/Hold after initialization. If true, the ramp/soak is in Operator Manual or Program Hold mode after initialization completes. Otherwise, the ramp/soak remains is in its previous mode after initialization completes. Default is false.
CyclicSingle	BOOL	Cyclic/single execution. True for cyclic action or false for single action. Cyclic action continuously repeats the ramp/soak profile. Single action performs the ramp/ soak profile once and then stops. Default is false.
TimeRate	BOOL	Time/rate ramp value configuration. True if the RampValue parameters are entered as a time in minutes to reach the soak temperature. False if the RampValue parameters are entered as a rate in units/minute. Default is false.
GuarRamp	BOOL	Guaranteed ramp. If true and the instruction is in Auto, ramping is temporarily suspended if the PV differs from the Output by more than RampDeadband. Default is cleared.
RampDeadband	REAL	Guaranteed ramp deadband. Specify the amount in engineering units that PV is allowed to differ from the output when GuarRamp is on. If this value is invalid, the instruction sets RampDeadband = 0.0 and the instruction sets the appropriate bit in Status. Valid = any float $\geq$ 0.0 Default = 0.0
GuarSoak	BOOL	Guaranteed soak. If true and the instruction is in auto, the soak timer is cleared if the PV differs from the Output by more than SoakDeadband. Default is false.

Input Parameter	Data Type	Description
SoakDeadband	REAL	Guaranteed soak deadband. Specify the amount in engineering units that the PV is allowed to differ from the output when GuarSoak is on. If this value is invalid, the instruction sets SoakDeadband = 0.0 and the instruction sets the appropriate bit in Status. Valid = any float ≥ 0.0 Default = 0.0
CurrentSegProg	DINT	Current segment program. The user program writes a requested value for the CurrentSeg into this input. This value is used if the ramp/soak is in Program Manual mode. If this value is invalid, the instruction sets the appropriate bit in Status. Valid = 0 to NumberOfSegs-1 Default = 0
OutProg	REAL	Output program. The user program writes a requested value for the Out into this input. This value is used as the Out when the ramp/soak is in Program Manual mode. Valid = any float Default = 0.0
SoakTimeProg	REAL	Soak time program. The user program writes a requested value for the SoakTimeLeft into this input. This value is used if the ramp/soak is in Program Manual mode. If this value is invalid, the instruction sets the appropriate bit in Status. Valid = 0.0 to maximum positive float Default = 0.0
CurrentSegOper	DINT	Current segment operator. The operator interface writes a requested value for the CurrentSeg into this input. This value is used if the ramp/soak is in Operator Manual mode. If this value is invalid, the instruction sets the appropriate bit in Status. Valid = 0 to NumberOfSegs-1 Default = 0
OutOper	REAL	Output operator. The operator interface writes a requested value for the Out into this input. This value is used as the Out when the ramp/soak is in Operator Manual mode. Valid = any float Default = 0.0
SoakTimeOper	REAL	Soak time operator. The operator interface writes a requested value for the SoakTimeLeft into this input. This value is used if the ramp/soak is in Operator Manual mode. If this value is invalid, the instruction sets the appropriate bit in Status. Valid = 0.0 to maximum positive float Default = 0.0
ProgProgReq	BOOL	Program program request. Set to true by the user program to request Program control. Ignored if ProgOperReq is true. Holding this true and ProgOperReq false locks the instruction in Program control. Default is false.
ProgOperReq	BOOL	Program operator request. Set to true by the user program to request Operator control. Holding this true locks the instruction in Operator control. Default is false.

Input Parameter	Data Type	Description
ProgAutoReq	BOOL	Program auto mode request. Set to true by the user program to request the ramp/soak to enter Auto mode. Ignored if the loop is in Operator control, if ProgManualReq is true, or if ProgHoldReq is true. Default is false.
ProgManualReq	BOOL	Program manual mode request. Set to true by the user program to request the ramp/soak to enter Manual mode. Ignored if the ramp/soak is in Operator control or if ProgHoldReq is true. Default is false.
ProgHoldReq	BOOL	Program hold mode request. Set to true by the user program to request to stop the ramp/ soak without changing the Out, CurrentSeg, or SoakTimeLeft. Also useful when a PID loop getting its setpoint from the ramp/soak leaves cascade. An operator can accomplish the same thing by placing the ramp/soak into Operator Manual mode. Default is false.
OperProgReq	BOOL	Operator program request. Set to true by the operator interface to request Program control. Ignored if ProgOperReq is true. The instruction clears this input to false. Default is false.
OperOperReq	BOOL	Operator operator request. Set to true by the operator interface to request Operator control. Ignored if ProgProgReq is true and ProgOperReq is false. The instruction clears this input to false. Default is false.
OperAutoReq	BOOL	Operator auto mode request. Set to true by the operator interface to request the ramp/soak to enter Auto mode. Ignored if the loop is in Program control or if OperManualReq is true. The instruction clears this input to false. Default is false.
OperManualReq	BOOL	Operator manual mode request. Set to true by the operator interface to request the ramp/soak to enter Manual mode. Ignored if the loop is in Program control. The instruction clears this input to false. Default is false.
Initialize	BOOL	Initialize program and operator values. When true and in manual, the instruction sets CurrentSegProg = 0, CurrentSegOper = 0, SoakTimeProg = SoakTime[0], and SoakTimeOper = SoakTime[0]. Initialize is ignored when in Auto or Hold mode. The instruction clears this parameter to false. Default is false.
ProgValueReset	BOOL	Reset program control values. When true, the instruction clears ProgProgReq, ProgOperReq, ProgAutoReq, ProgHoldReq, and ProgManualReq to false. Default is false.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if Out overflows.
Out	REAL	The output of the ramp/soak instruction.
CurrentSeg	DINT	Current segment number. Displays the current segment number in the ramp/soak cycle. Segments start numbering at 0.
SoakTimeLeft	REAL	Soak time left. Displays the soak time remaining for the current soak.
GuarRampOn	BOOL	Guaranteed ramp status. Set to true if the Guaranteed Ramp feature is in use and the ramp is temporarily suspended because the PV differs from the output by more than the RampDeadband.
GuarSoakOn	BOOL	Guaranteed soak status. Set to true if the Guaranteed Soak feature is in use and the soak timer is cleared because the PV differs from the output by more than the SoakDeadband.
ProgOper	BOOL	Program/Operator control indicator. True when in Program control. False when in Operator control.
Auto	BOOL	Auto mode. True when the ramp/soak is in Program Auto or Operator Auto mode.
Manual	BOOL	Manual mode. True when the ramp/soak is in Program Manual or Operator Manual mode.
Hold	BOOL	Hold mode. True when the ramp/soak is in program Hold mode.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
PVFaulted (Status.1)	BOOL	PVHealth is bad.
NumberOfSegsInv (Status.2)	BOOL	The NumberOfSegs value is invalid value or is not compatible with an array size.
RampDeadbandInv (Status.3)	BOOL	Invalid RampDeadband value.
SoakDeadbandInv (Status.4)	BOOL	Invalid SoakDeadband value.
CurrSegProgInv (Status.5)	BOOL	Invalid CurrSegProg value.
SoakTimeProgInv (Status.6)	BOOL	Invalid SoakTimeProg value.
CurrSegOperInv (Status.7)	BOOL	Invalid CurrSegOper value.
SoakTimeOperInv (Status.8)	BOOL	Invalid SoakTimeOper value.
RampValueInv (Status.9)	BOOL	Invalid RampValue value.
SoakTimeInv (Status.10)	BOOL	Invalid SoakTime value

Description

The RMPS instruction is typically used to provide a temperature profile in a batch heating process. The output of this instruction is typically the input to the setpoint of a PID loop.

Whenever the value computed for the output is invalid, NAN, or ± INF, the instruction sets Out = the invalid value and sets the math overflow status flag. The internal parameters are not updated. In each subsequent scan, the output is computed using the internal parameters from the last scan when the output was valid.

Monitoring the RMPS Instruction

There is an operator faceplate available for the RMPS instruction.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	Clear CurrentSeg to 0. Mode is set to operator manual mode. SoakTimeProg and SoakTimeOper are set to SoakTime[0] if SoakTime[0] is valid.
Instruction first scan	All the operator request inputs are cleared to false. If ProgValueReset is true, all the program request inputs are cleared to false. The operator control mode is set to manual mode if the current mode is hold.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.

Condition/State	Action Taken
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Initial Mode Applied on Instruction First Scan

The following table shows the ending control based on the program request inputs.

Control at Start of First Scan	ProgOperReq	ProgProgReq	ProgValueReset	FirstRun	Control at End of First Scan
Operator Control	false	true	false	na	Program Control
	na	false	na	na	Operator Control
Program Control	true	Na	false	false	Operator Control
	na	Na	true	true	Operator Control
	false	false	false	true	Operator Control
	false	true	false	na	Program Control
	na	Na	true	false	Program Control
	false	false	false	false	Program Control

The following table shows the ending control based on the Manual, Auto, and Hold mode requests.

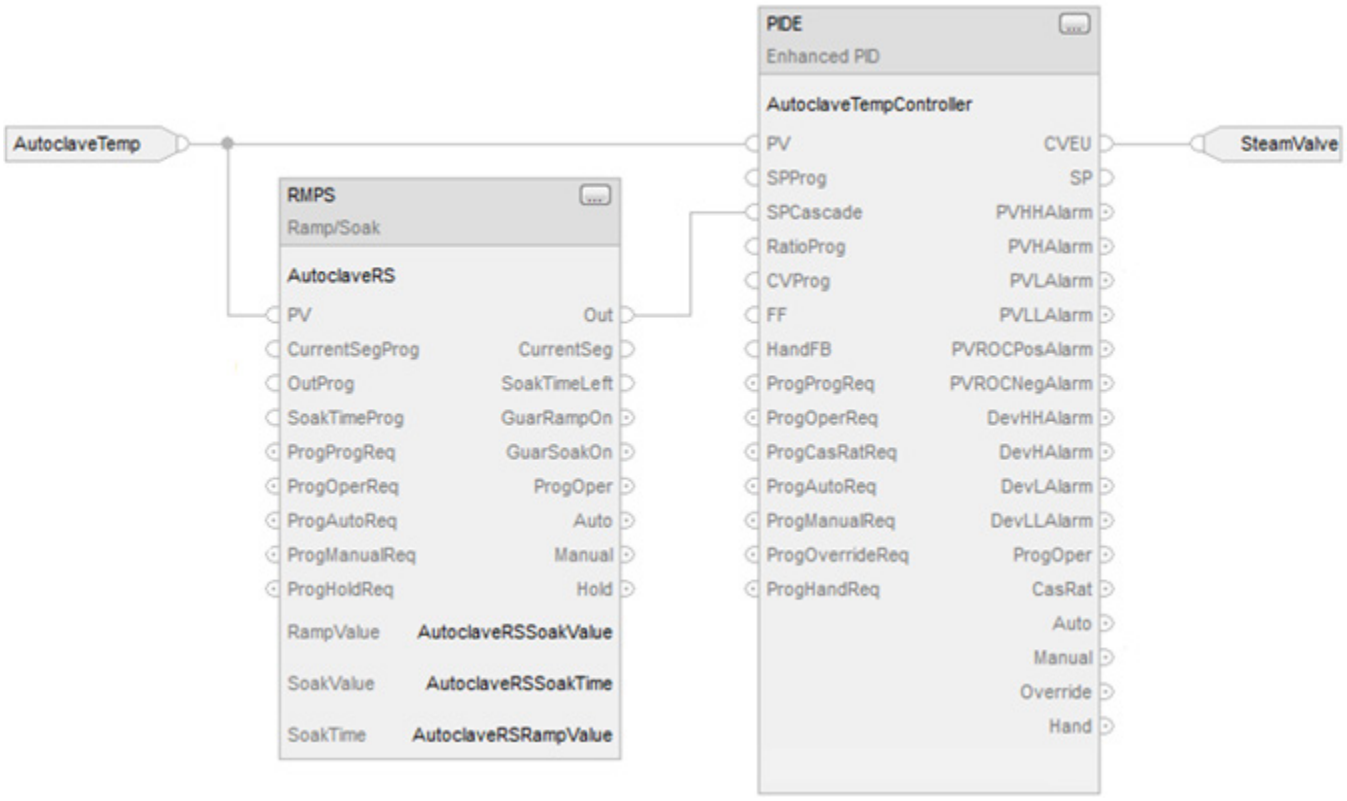
Control at Start of First Scan	Oper Auto Req	Oper Man Req	Prog Auto Req	Prog Man Req	Prog Hold Req	Man Hold AftInit	Prog Value Reset	First Run	Control at End of First Scan
Operator Control	na	na	na	na	na	false	na	false	Operator Current mode
	na	na	na	na	na	na	na		Operator Manual mode
	na	na	na	na	na	true	na	na	
Program Control	na	na	false	false	false	false	na	false	Program Current mode
	na	na	na	na	na	false	true	false	
	na	na	true	false	false	false	false	na	Program Auto mode
	na	na	na	true	false	false	false	na	Program Manual mode
	na	na	na	na	true	false	false	na	Program Hold mode
	na	na	na	na	na	true	na	na	

Example

In this example, the RMPS instruction drives the setpoint of a PIDE instruction. When the PIDE instruction is in Cascade/Ratio mode, the output of the RMPS instruction is used as the setpoint. The PV to the PIDE instruction can be optionally fed into the PV input of the RMPS instruction if you want to use guaranteed ramping and/or guaranteed soaking.

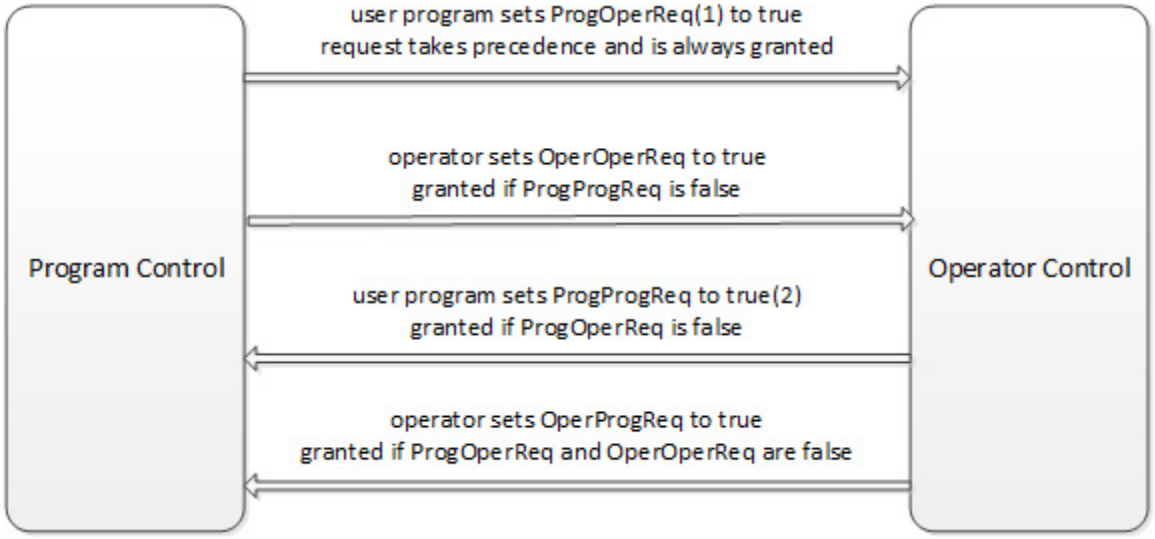
In this example, the AutoclaveRSSoakValue, AutoclaveRSSoakTime, and AutoclaveRSRampValue arrays are REAL arrays with 10 elements to allow up to a 10 segment RMPS profile.

Function Block



Switching Between Program Control and Operator Control

The RMPS instruction can be controlled by either a user program or through an operator interface. Control can be changed at any time.



(1) You can lock the instruction in Operator control by leaving ProgOperReq to true.

(2) You can lock the instruction in Program control by leaving ProgProgReq to true while ProgOperReq is false.

When transitioning from Operator control to Program control while the ProgAutoReq, ProgManualReq, and ProgHoldReq inputs are false, the mode is determined as follows:

If the instruction was in Operator Auto mode, then the transition is to Program Auto mode.

If the instruction was in Operator Manual mode, then the transition is to Program Manual mode.

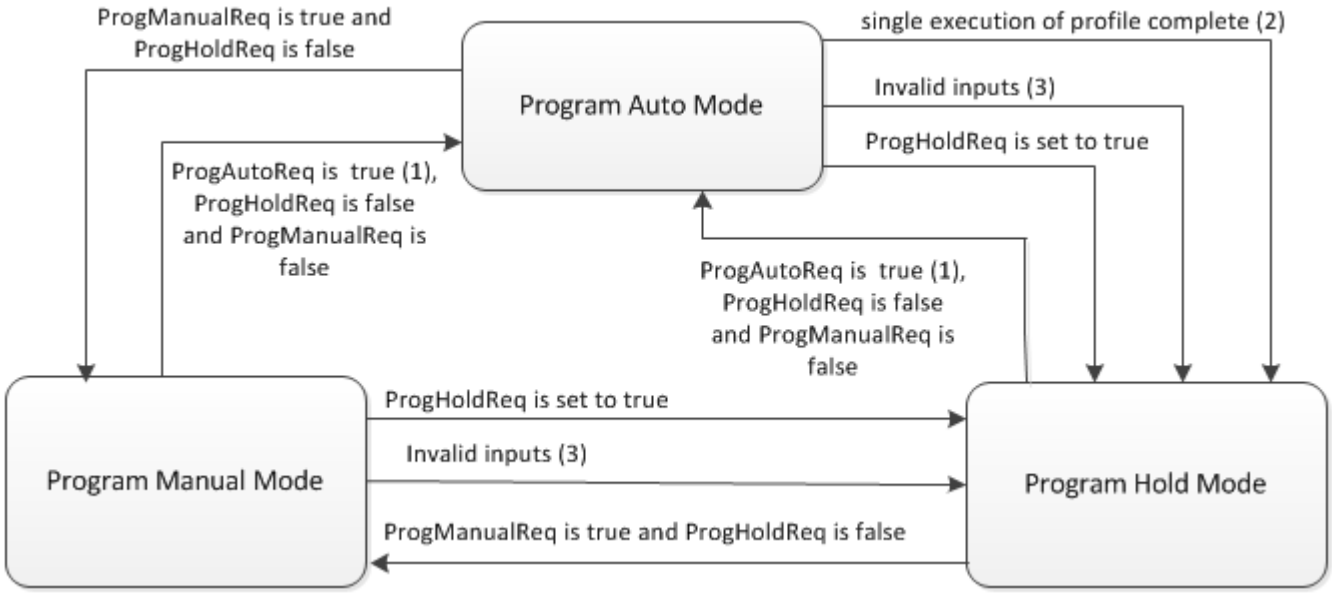
When transitioning from Program control to Operator control while the OperAutoReq and OperManualReq inputs are false, the mode is determined as follows:

If the instruction was in Program Auto mode, then the transition is to Operator Auto mode.

If the instruction was in Program Manual or Program Hold mode, then the transition is to Operator Manual mode.

Program Control

The following diagram shows how the RMPS instruction operates when in Program control.



(1) In single (non-cyclic) execution, you must toggle ProgAutoReq from false to true if one execution of the ramp/soak profile is complete and you want another execution of the ramp/soak profile.

(2) When the instruction is configured for single execution, and the Auto mode Ramp-Soak profile completes, the instruction transitions to Hold mode.

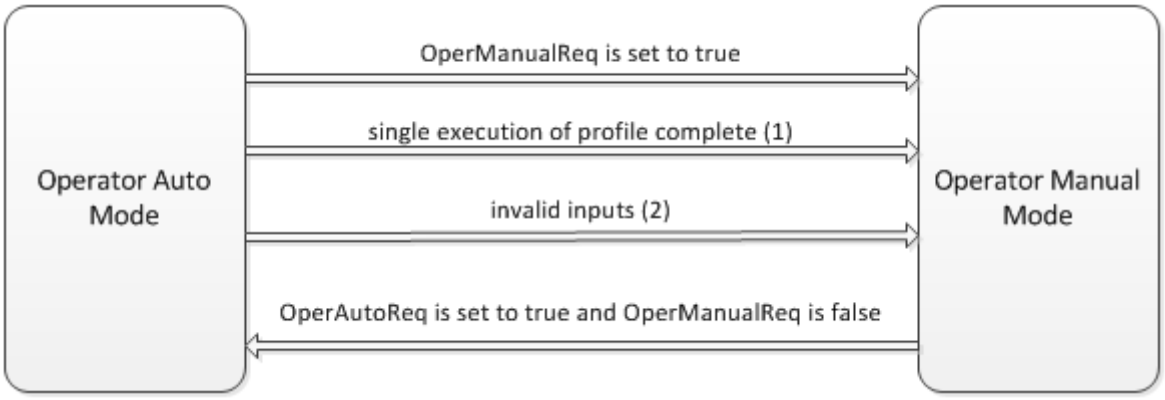
(3) The instruction is placed in Hold mode if PVFaulted is true or any of the following inputs are invalid: NumberOfSegs, CurrentSegProg, or SoakTimeProg.

The following table describes the possible Program modes.

Mode	Description
Program Auto Mode	While in Auto mode, the instruction sequentially executes the ramp/ soak profile.
Program Manual Mode	While in Manual mode the user program directly controls the instruction's Out. The CurrentSegProg, SoakTimeProg, and OutProg inputs are transferred to the CurrentSeg, SoakTimeLeft, and Out outputs. When the instruction is placed in auto mode, the ramp/soak function resumes with the values last input from the user program. CurrentSegProg and SoakTimeProg are not transferred if they are invalid. To facilitate a "bumpless" transition into Manual mode, the CurrentSegProg, SoakTimeProg, and OutProg inputs are continuously updated to the current values of CurrentSeg, SoakTimeLeft, and Out when ProgValueReset is set and the instruction is not in Program Manual mode.
Program Hold Mode	While in Hold mode, the instruction's outputs are maintained at their current values. If in this mode when ProgOperReq is set to change to Operator control, the instruction changes to Operator Manual mode.

Operator Control

The following diagram shows how the RMPS instruction operates when in Operator control.



(1) When the instruction is configured for Single Execution, and the Auto mode ramp/soak profile completes, the instruction transitions to manual mode.

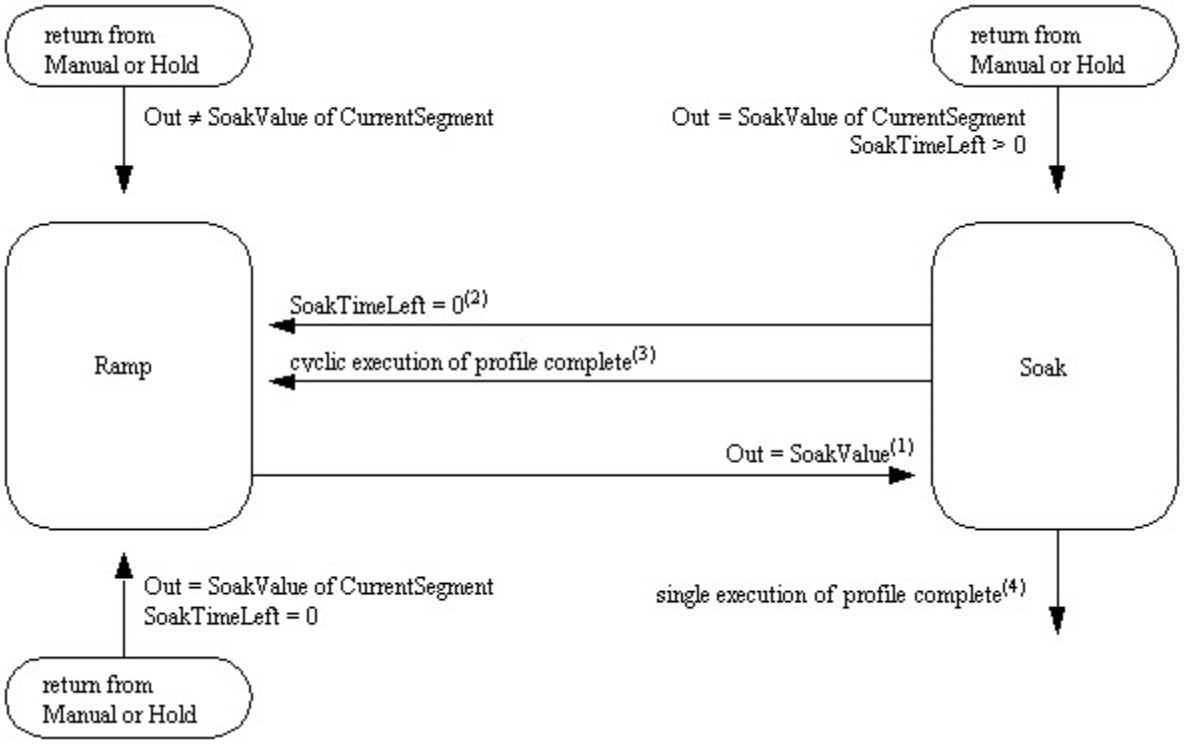
(2) The instruction is placed in Manual mode if PVFaulted is true or any of the following inputs are invalid: NumberOfSegs, CurrentSegOper, or SoakTimeOper.

The following table describes the possible Operator modes.

Mode	Description
Operator Auto Mode	While in Auto mode, the instruction sequentially executes the ramp/ soak profile.
Operator Manual Mode	While in Manual mode the operator directly controls the instruction's Out. The CurrentSegOper, SoakTimeOper, and OutOper inputs are transferred to the CurrentSeg, SoakTimeLeft, and Out outputs. When the instruction is placed in Auto mode, the ramp/soak function resumes with the values last input from the operator. CurrentSegOper and SoakTimeOper are not transferred if they are invalid. To facilitate a "bumpless" transition into Manual mode, the CurrentSegOper, SoakTimeOper, and OutOper inputs are continuously updated to the current values of CurrentSeg, SoakTimeLeft, and Out whenever the instruction is not in Operator Manual mode.

Executing the Ramp/Soak Profile

The following diagram illustrates how the RMPS instruction executes the ramp/soak profile.



(1) The Ramp is complete when Out = SoakValue. If, during ramp execution, Out > SoakValue, Out is limited to SoakValue.

(2) Soaking is complete when Out is held for the amount of time specified in the current segment's SoakTime. If the segment executed was not the last segment, CurrentSeg increments by one.

(3) Soaking has completed for the last programmed segment and the instruction is configured for cyclic execution. The instruction sets CurrentSeg = 0.

(4) Soaking has completed for the last programmed segment and the instruction is configured for single execution.

(5) When returning to Auto mode, the instruction determines if ramping or soaking resumes. What to do next depends on the values of Out, SoakTimeLeft, and the SoakValue of the current segment. If Out = SoakValue for the current segment, and SoakTimeLeft = 0, then the current segment has completed and the next segment starts.

Ramping

The ramp cycle ramps Out from the previous segment's SoakValue to the current segment's SoakValue. The time in which the ramp is traversed is defined by the RampValue parameters.

Ramping is positive if Out < target SoakValue of the current segment. If the ramp equation calculates a new Out which exceeds the target SoakValue, the Out is set to the target SoakValue.

Ramping is negative if Out > the target SoakValue of the current segment. If the ramp equation calculates a new Out which is less than the target SoakValue, the Out is set to the target SoakValue.

Each segment has a ramp value. You have the option of programming the ramp in units of time or rate. All segments must be programmed in the same units. The following table describes the ramping options:

Parameter	Description
Time-based ramping	TimeRate is set to true for time-based ramping (in minutes) The rate of change for the current segment is calculated and either added or subtracted to Out until Out reaches the current segment's soak value. In the following equation, DeltaT is the elapsed time in minutes since the instruction last executed. $Out = Out \pm \frac{(SoakValue_{CurrentSeg} - RampStart)}{RampValue_{CurrentSeg}} \times \Delta t$ Where RampStart is the value of Out at the start of the Current Segment.
Rate-based ramping	TimeRate is cleared to false for rate-based ramping (in units/minute) The programmed rate of change is either added or subtracted to Out until Out reaches the current segment's soak value. In the following equation, DeltaT is the elapsed time in minutes since the instruction last executed. $Out = Out \pm RampValue_{CurrentSeg} \times \Delta t$

Guaranteed Ramping

Set the input GuarRamp to true to enable guaranteed ramping. When enabled, the instruction monitors the difference between Out and PV. If the difference is outside of the programmed RampDeadband, the output is left unchanged until the difference between PV and Out are within the deadband. The output GuarRampOn is set to true whenever Out is held due to guaranteed ramping being in effect.

Soaking

Soaking is the amount of time the block output is to remain unchanged until the next ramp-soak segment is started. The soak cycle holds the output at the SoakValue for a programmed amount of time before proceeding to the next segment. The amount of time the output is to soak is programmed in the SoakTime parameters.

Each segment has a SoakValue and SoakTime. Soaking begins when Out is ramped to the current segment's SoakValue. SoakTimeLeft represents the time in minutes remaining for the output to soak. During ramping, SoakTimeLeft is set to the current segment's SoakTime. Once ramping is

complete, SoakTimeLeft is decreased to reflect the time in minutes remaining for the current segment. SoakTimeLeft = 0 when SoakTime expires.

Guaranteed Soaking

Set the input GuarSoak to true to enable guaranteed soaking. When enabled, the instruction monitors the difference between Out and PV. If the difference is outside of the SoakDeadband, timing of the soak cycle is suspended and the internal soak timer is cleared. When the difference between Out and PV returns to within the deadband, timing resumes. The output GuarSoakOn is set to true when timing is held due to guaranteed soaking being in effect.

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Scale (SCL)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

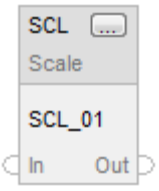
The SCL instruction converts an unscaled input value to a floating point value in engineering units.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

SCL(SCL_tag)

Operands

Function Block

Operand	Type	Format	Description
SCL tag	SCALE	Structure	SCL structure

Structured Text

Operand	Type	Format	Description
SCL tag	SCALE	Structure	SCL structure

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

SCALE Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
In	REAL	The analog signal input. Valid = any float Default = 0.0
InRawMax	REAL	The maximum value attainable by the input to the instruction. If InRawMax ≤ InRawMin, the instruction sets the appropriate bit in Status and stops updating the output. Valid = InRawMax > InRawMin Default = 0.0
InRawMin	REAL	The minimum value attainable by the input to the instruction. If InRawMin ≥ InRawMax, the instruction sets the appropriate bit in Status and stops updating the output. Valid = InRawMin < InRawMax Default = 0.0
InEUMax	REAL	The scaled value of the input corresponding to InRawMax. Valid = any real value Default = 0.0
InEUMin	REAL	The scaled value of the input corresponding to InRawMin. Valid = any real value Default = 0.0

Input Parameter	Data Type	Description
Limiting	BOOL	Limiting selector. If true, Out is limited to between InEUMin and InEUMax. Default is false.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if Out overflows.
Out	REAL	The output that represents scaled value of the analog input. Valid = any real value Default = InEUMin
MaxAlarm	BOOL	The above maximum input alarm indicator. This value is set to true when In> InRawMax.
MinAlarm	BOOL	The below minimum input alarm indicator. This value is set to true when In < InRawMin.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
InRawRangeInv (Status.1)	BOOL	InRawMin ≥ InRawMax.

Description

Use the SCL instruction with analog input modules that do not support scaling to a full resolution floating point value.

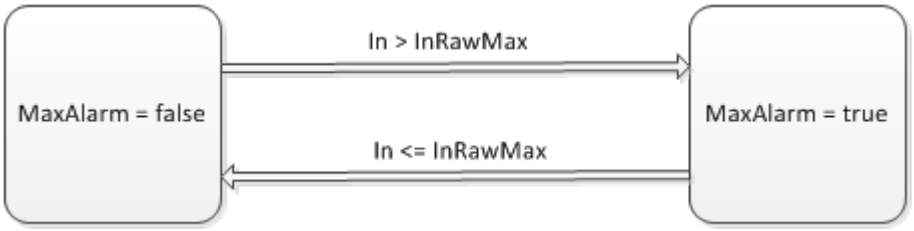
For example, the 1771-IFE module is a 12-bit analog input module that supports scaling only in integer values. If you use a 1771-IFE module to read a flow of 0-100 gallons per minute (gpm), you typically do not scale the module from 0-100 because that limits the resolution of the module. Instead, use the SCL instruction and configure the module to return an unscaled (0-4095) value, which the SCL instruction converts to 0-100 gpm (floating point) without a loss of resolution. This scaled value could then be used as an input to other instructions.

The SCL instruction uses this algorithm to convert unscaled input into a scaled value:

$$Out = (In - InRawMin) \times \left(\frac{InEUMax - InEUMin}{InRawMax - InRawMin} \right) + InEUMin$$

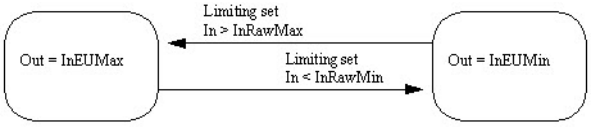
Alarming

Once the instruction calculates Out, the MaxAlarm and MinAlarm are determined as follows:



Limiting

Limiting is performed on Out when Limiting is set. The instruction sets Out = InEUMax when In > InRawMax. The instruction sets Out = InEUMin when In < InRawMin.



Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true The instruction executes.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	EnableIn and EnableOut bits are cleared to false.

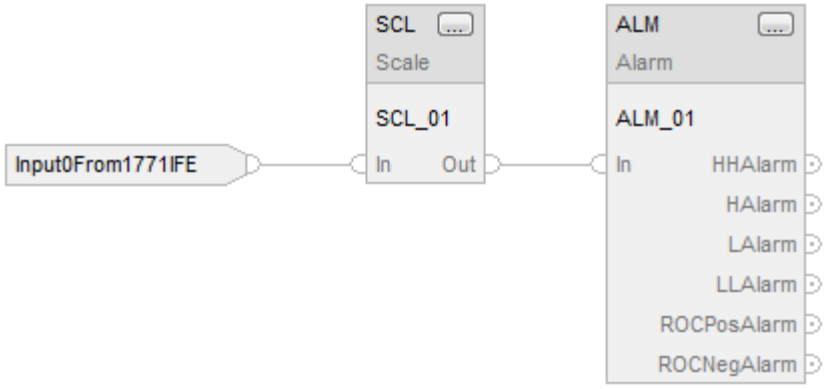
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

The SCL instruction is typically used with analog input modules that do not support on-board scaling to floating point engineering units. In this example, the SCL instruction scales an analog input from a 1771-IFE module. The instruction places the result in Out, which is used by an ALM instruction.

Function Block



Structured Text

```
SCL_01.In := Input0From1771IFE;
SCL(SCL_01);

ALM_01.In := SCL_01.Out;
ALM(ALM_01);
```

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Split Range Time Proportional (SRTP)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380,

CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

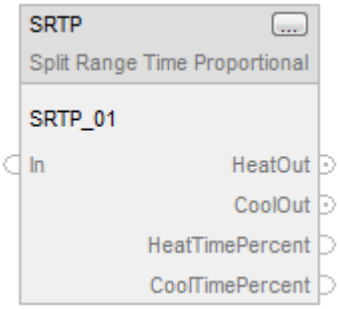
The SRTP instruction takes the 0-100% output of a PID loop and drives heating and cooling digital output contacts with a periodic pulse. This instruction controls applications such as barrel temperature control on extrusion machines.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

SRTP(SRTP_tag)

Operands

Function Block

Operand	Type	Format	Description
SRTP tag	SPLIT_RANGE	Structure	SRTP structure

Structured Text

Operand	Type	Format	Description
SRTP tag	SPLIT_RANGE	Structure	SRTP structure

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

SPLIT_RANGE Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
In	REAL	The analog signal input asking for heating or cooling. This input typically comes from the CVEU of a PID loop. Valid = any float
CycleTime	REAL	The period of the output pulses in seconds. A value of zero turns off both heat and cool outputs. If this value is invalid, the instruction assumes a value of zero and sets the appropriate bit in Status. Valid = any positive float Default = 0.0
MaxHeatIn	REAL	Maximum heat input. This value specifies the percentage of the In which will cause maximum heating. This is typically 100% for a heat/cool loop. Valid = any float Default = 100.0
MinHeatIn	REAL	Minimum heat input. Specify the percent of In that represents the start of the heating range and causes minimum heating. This is typically 50% for a heat/cool loop. Valid = any float Default = 50.0
MaxCoolIn	REAL	Maximum cool input. Specify the percent of In that causes maximum cooling. This is typically 0% for a heat/cool loop. Valid = any float Default = 0.0
MinCoolIn	REAL	Minimum cool input. Specify the percent of In that causes minimum cooling. This is typically 50% for a heat/cool loop. Valid = any float Default = 50.0
MaxHeatTime	REAL	Maximum heat time in seconds. Specify the maximum time in seconds that a heating pulse can be on. If the instruction calculates HeatTime to be greater than this value, HeatTime is limited to MaxHeatTime. If MaxHeatTime is invalid, the instruction assumes a value of CycleTime and sets the appropriate bit in Status. Valid = 0.0 to CycleTime Default = CycleTime

Input Parameter	Data Type	Description
MinHeatTime	REAL	Minimum heat time in seconds. Specify the minimum time in seconds that a heating pulse can be on. If the instruction calculates HeatTime to be less than this value, HeatTime is set to zero. If MinHeatTime is invalid, the instruction assumes a value of zero and sets the appropriate bit in Status. Valid = 0.0 to MaxHeatTime Default = 0.0
MaxCoolTime	REAL	Maximum cool time in seconds. Specify the maximum time in seconds that a cooling pulse can be on. If the instruction calculates CoolTime to be larger than this value, CoolTime is limited to MaxCoolTime. If MaxCoolTime is invalid, the instruction assumes a value of CycleTime and sets the appropriate bit in Status. Valid = 0.0 to CycleTime Default = CycleTime
MinCoolTime	REAL	Minimum cool time in seconds. Specify the minimum time in seconds that a cooling pulse can be on. If the instruction calculates CoolTime to be less than this value, CoolTime is set to zero. If MinCoolTime is invalid, the instructions assumes a value of zero and sets the appropriate bit in Status. Valid = 0.0 to MaxCoolTime Default = 0.0

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if HeatTimePercent or CoolTimePercent overflows.
HeatOut	BOOL	Heating output pulse. The instruction pulses this output for the heating contact.
CoolOut	BOOL	Cooling output pulse. The instruction pulses this output for the cooling contact.
HeatTimePercent	REAL	Heating output pulse time in percent. This value is the calculated percent of the current cycle that the HeatingOutput will be on. This allows you to use the instruction with an analog output for heating if required.
CoolTimePercent	REAL	Cooling output pulse time in percent. This value is the calculated percent of the current cycle that the CoolingOutput will be on. This allows you to use the instruction with an analog output for cooling if required.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.

Output Parameter	Data Type	Description
CycleTimeInv (Status.1)	BOOL	Invalid CycleTime value. The instruction uses zero.
MaxHeatTimeInv (Status.2)	BOOL	Invalid MaxHeatTime value. The instruction uses the CycleTime value.
MinHeatTimeInv (Status.3)	BOOL	Invalid MinHeatTime value. The instruction uses zero.
MaxCoolTimeInv (Status.4)	BOOL	Invalid MaxCoolTime value. The instruction uses the CycleTime value.
MinCoolTimeInv (Status.5)	BOOL	Invalid MinCoolTime value. The instruction uses zero.
HeatSpanInv (Status.6)	BOOL	MaxHeatIn = MinHeatIn.
CoolSpanInv (Status.7)	BOOL	MaxCoolIn = MinCoolIn.

Description

The length of the SRTP pulse is proportional to the PID output. The instruction parameters accommodate heating and cooling applications.

Using the Internal Cycle Timer

The instruction maintains a free running cycle timer that cycles from zero to the programmed CycleTime. The internal timer is updated by DeltaT. DeltaT is the elapsed time since the instruction last executed. This timer determines if the outputs need to be turned on.

You can change CycleTime at any time. If CycleTime = 0, the internal timer is cleared and HeatOut and CoolOut are cleared to false.

Calculating Heat and Cool Times

Heat and cool times are calculated every time the instruction is executed.

HeatTime is the amount of time within CycleTime that the heat output is to be turned on.

$$HeatTime = \frac{In - MinHeatIn}{MaxHeatIn - MinHeatIn} \times CycleTime$$

If HeatTime < MinHeatTime, set HeatTime = 0.

If HeatTime> MaxHeatTime, limit HeatTime = MaxHeatTime.

HeatTimePercent is the percentage of CycleTime that the HeatOut pulse is true.

$$HeatTimePercent = \frac{HeatTime}{CycleTime} \times 100$$

CoolTime is the amount of time within CycleTime that the cool output is to be turned on.

$$CoolTime = \frac{In - MinCoolIn}{MaxCoolIn - MinCoolIn} \times CycleTime$$

If CoolTime < MinCoolTime, set CoolTime = 0.

If CoolTime > MaxCoolTime, limit CoolTime = MaxCoolTime.

CoolTimePercent is the percentage of CycleTime that the CoolOut pulse is true.

$$CoolTimePercent = \frac{CoolTime}{CycleTime} \times 100$$

The instruction controls heat and cool outputs using these rules:

- Set HeatOut to true if HeatTime ≥ the internal cycle time accumulator. Clear HeatOut to false when the internal cycle timer > HeatTime.
- Set CoolOut to true if CoolTime ≥ the internal cycle time accumulator. Clear CoolOut to false if the internal cycle timer > CoolTime.
- Clear HeatOut and CoolOut to false if CycleTime = 0.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A

Condition/State	Action Taken
Instruction first scan	HeatOut and CoolOut are cleared to false. HeatTimePercent and CoolTimePercent are cleared to 0.0.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

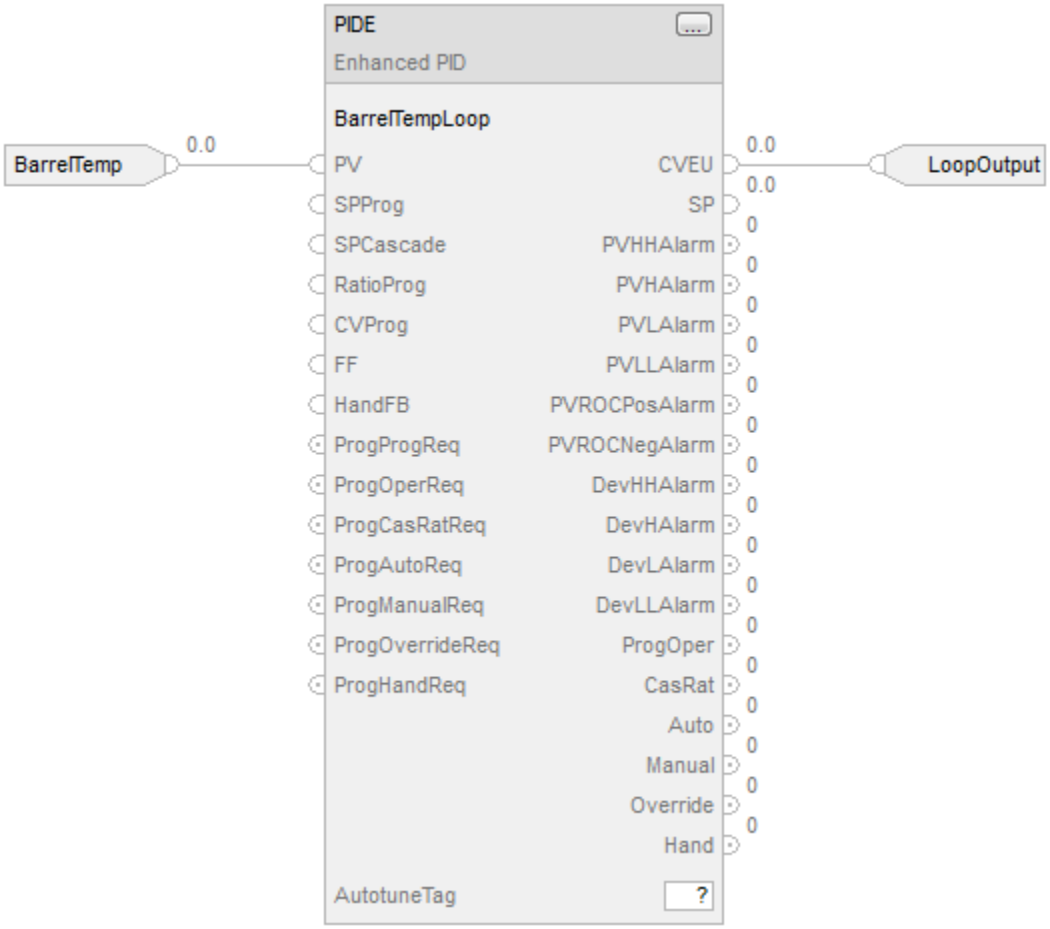
Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

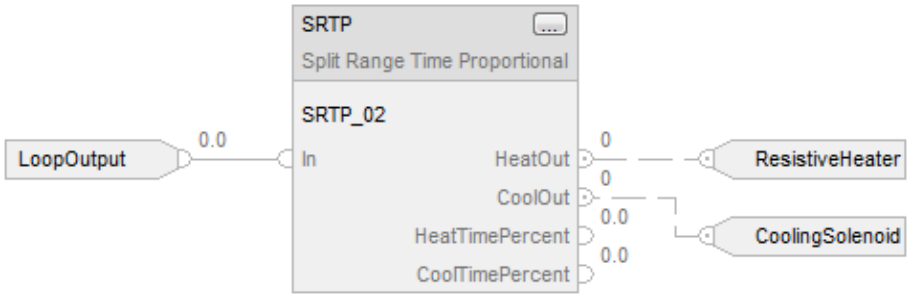
In this example, a PIDE instruction executes in a slow, lower priority task because it is a slow, temperature loop. The output of the PIDE instruction is a controller-scoped tag because it becomes the input to an SRTP instruction. The SRTP instruction executes in a faster, higher priority task so that the pulse outputs are more accurate

Function Block

Place the PIDE instruction in a slow, lower priority task



Place the SRTP instruction in a faster, higher priority task.



Structured Text

Place the PIDE instruction in a slow, lower priority task.

```
BarrelTempLoop.PV := BarrelTemp;

    PIDE(BarrelTempLoop);

LoopOutput := BarrelTempLoop.CVEU;
```

Place the SRTP instruction in a faster, higher priority task.

```
SRTP_02.In := LoopOutput;
  SRTP(SRTP_02);
ResistiveHeater := SRTP_02.HeatOut;
CoolingSolenoid := SRTP_02.CoolOut;
```

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Totalizer (TOT)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

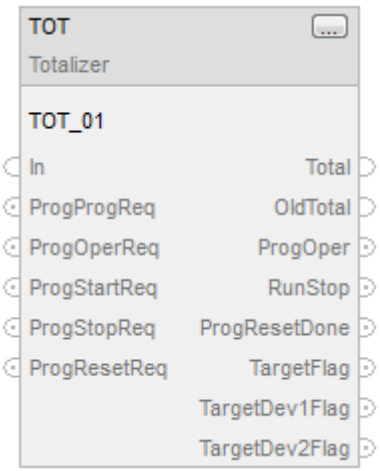
The TOT instruction provides a time-scaled accumulation of an analog input value.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

TOT(TOT_tag)

Operands

Function Block

Operand	Type	Format	Description
TOT tag	TOTALIZER	Structure	TOT structure

Structured Text

Operand	Type	Format	Description
TOT tag	TOTALIZER	Structure	TOT structure

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

TOTALIZER Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0
InFault	BOOL	Bad health indicator of In. If true, it indicates the input signal has an error, the instruction sets the appropriate bit in Status, the control algorithm is not executed, and Total is not updated. Default is false.
TimeBase	DINT	The timebase input. The time base of the totalization based on the In engineering units. 0 = Seconds 1 = Minutes 2 = Hours 3 = Days For example, use TimeBase = minutes if In has units of gal/min. If this value is invalid, the instruction sets the appropriate bit in Status and does not update the Total. For more information about timing modes, see Function Block Attributes. Valid = 0 to 3 Default = 0

Input Parameter	Data Type	Description
Gain	REAL	The multiplier of the incremental totalized value. The user can use the Gain to convert the units of totalization. For example, use the Gain to convert gal/min to a total in barrels. Valid = any float Default = 1.0
ResetValue	REAL	The reset value input. The reset value of Total when OperResetReq or ProgResetReq transitions from false to true. Valid = any float Default = 0.0
Target	REAL	The target value for the totalized In. Valid = any float Default = 0.0
TargetDev1	REAL	The large deviation pre-target value of the Total compared to the Target. This value is expressed as a deviation from the Target. Valid = any float Default = 0.0
TargetDev2	REAL	The small deviation pre-target value of the Total compared to the Target. This value is expressed as a deviation from the Target. Valid = any float Default = 0.0
LowInCutoff	REAL	The instruction low input cutoff input. When the In is at or below the LowInCutoff value, totalization ceases. Valid = any float Default = 0.0
ProgProgReq	BOOL	Program program request. Set to true to request Program control. Ignored if ProgOperReq is true. Holding this true and ProgOperReq false locks the instruction in Program control. Default is false.
ProgOperReq	BOOL	Program operator request. Set to true to request Operator control. Holding this true locks the instruction in Operator control. Default is false.
ProgStartReq	BOOL	The program start request input. Set to true to request totalization to start. Default is false.
ProgStopReq	BOOL	The program stop request input. Set to true to request totalization to stop. Default is false.
ProgResetReq	BOOL	The program reset request input. Set to true to request the Total to reset to the ResetValue. Default is false.

Input Parameter	Data Type	Description
OperProgReq	BOOL	Operator program request. Set to true by the operator interface to request Program control. The instruction clears this input to false. Default is false.
OperOperReq	BOOL	Operator operator request. Set to true by the operator interface to request Operator control. The instruction clears this input to false. Default is false.
OperStartReq	BOOL	The operator start request input. Set to true by the operator interface to request totalization to start. The instruction clears this input to false. Default is false.
OperStopReq	BOOL	The operator stop request input. Set to true by the operator interface to request totalization to stop. The instruction clears this input to false. Default is false.
OperResetReq	BOOL	The operator reset request input. Set to true by the operator interface to request totalization to reset. The instruction clears this input to false. Default is false.
ProgValueReset	BOOL	Reset program control values. When true, clear all the program request inputs to false at each execution of the instruction. Default is false.
TimingMode	DINT	Selects timing execution mode. 0 = Period mode 1 = Oversample mode 2 = Real time sampling mode For more information about timing modes, see Function Block Attributes. Valid = 0 to 2 Default = 0
OversampleDT	REAL	Execution time for oversample mode. Valid = 0 to 4194.303 seconds Default = 0
RTSTime	DINT	Module update period for real time sampling mode Valid = 1 to 32,767ms Default = 1
RTSTimeStamp	DINT	Module time stamp value for real time sampling mode. Valid = 0 to 32,767ms Default = 0

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if Total overflows.

Output Parameter	Data Type	Description
Total	REAL	The totalized value if In.
OldTotal	REAL	The value of the total before a reset occurred. You can monitor this value to read the exact total just before the last reset.
ProgOper	BOOL	Program/operator control indicator. True when in Program control. False when in Operator control.
RunStop	BOOL	The indicator of the operational state of the totalizer. True when the TOT instruction is running. False when the TOT instruction is stopped.
ProgResetDone	BOOL	The indicator that the TOT instruction has completed a program reset request. Set to true when the instruction resets as a result of ProgResetReq. You can monitor this to determine that a reset successfully completed. Cleared to false when ProgResetReq is false.
TargetFlag	BOOL	The flag for Total. Set to true when Total ≥ Target.
TargetDev1Flag	BOOL	The flag for TargetDev1. Set to true when Total ≥ Target - TargetDev1.
TargetDev2Flag	BOOL	The flag for TargetDev2. Set to true when Total ≥ Target - TargetDev2.
LowInCutoffFlag	BOOL	The instruction low input cutoff flag output. Set to true when In £ LowInCutoff.
DeltaT	REAL	Elapsed time between updates. This is the elapsed time in seconds used by the control algorithm to calculate the process output.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
InFaulted (Status.1)	BOOL	In value faulted.
TimeBaseInv (Status.2)	BOOL	Invalid TimeBase value.
TimingModeInv (Status.27)	BOOL	Invalid TimingMode value.
RTSMissed (Status.28)	BOOL	Only used in real time sampling mode. Set to true when ABS(DeltaT – RTSTime) > 1 millisecond.
RTSTimeInv (Status.29)	BOOL	Invalid RTSTime value.
RTSTimeStampInv (Status.30)	BOOL	Invalid RTSTimeStamp value.
DeltaTInv (Status.31)	BOOL	Invalid DeltaT value. This can occur if OversampleDT is invalid in oversample timing mode.

Description

This instruction typically totals the amount of a material added over time, based on a flow signal.

The TOT instruction supports:

- Time base selectable as seconds, minutes, hours, or days.
- You can specify a target value and up to two pre-target values. Pre-target values are typically used to switch to a slower feed rate. Digital flags announce the reaching of the target or pre-target values.
- A low flow input cutoff that you can use to eliminate negative totalization due to slight flow meter calibration inaccuracies when the flow is shut off.
- Operator or program capability to start/stop/reset.
- A user defined reset value.
- Trapezoidal-rule numerical integration to improve accuracy.
- The internal totalization is done with double precision math to improve accuracy.

Monitoring the TOT Instruction

There is an operator faceplate available for the TOT instruction.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.

Condition/State	Action Taken
Instruction first run	Total is set to ResetValue. OldTotal is cleared to 0.0. ProgOper is cleared to false.
Instruction first scan	All operator request inputs are cleared to false. If ProgValueReset is true then all program request inputs are cleared to false.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Check for Low Input Cutoff

If (In <= LowInCutoff), the instruction sets LowInCutoffFlag to true and makes In(n-1) = 0.0.

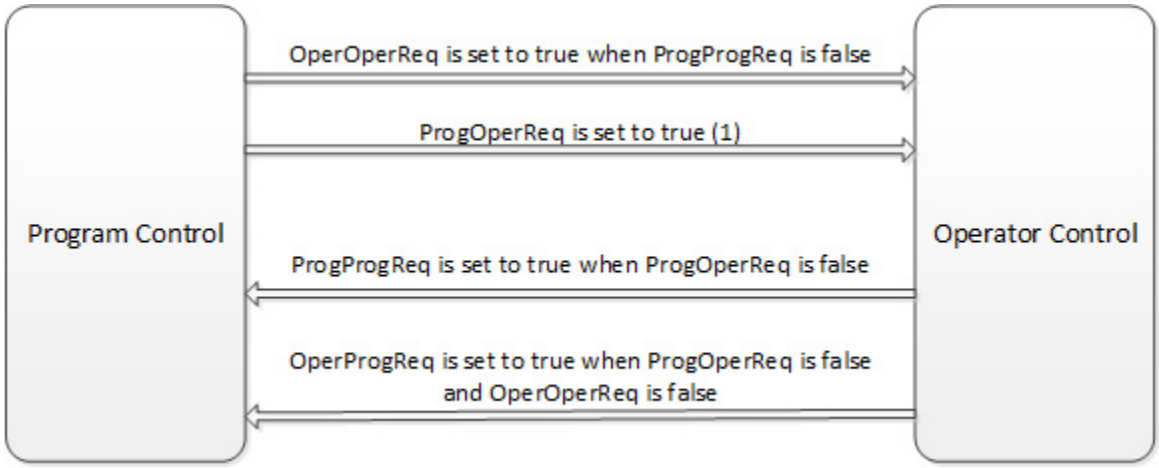
Otherwise, the instruction clears LowInCutoffFlag to false.

When the LowInCutoffFlag is true, the operation mode is determined but totalization ceases.

When LowInCutoffFlag is false, totalization continues that scan.

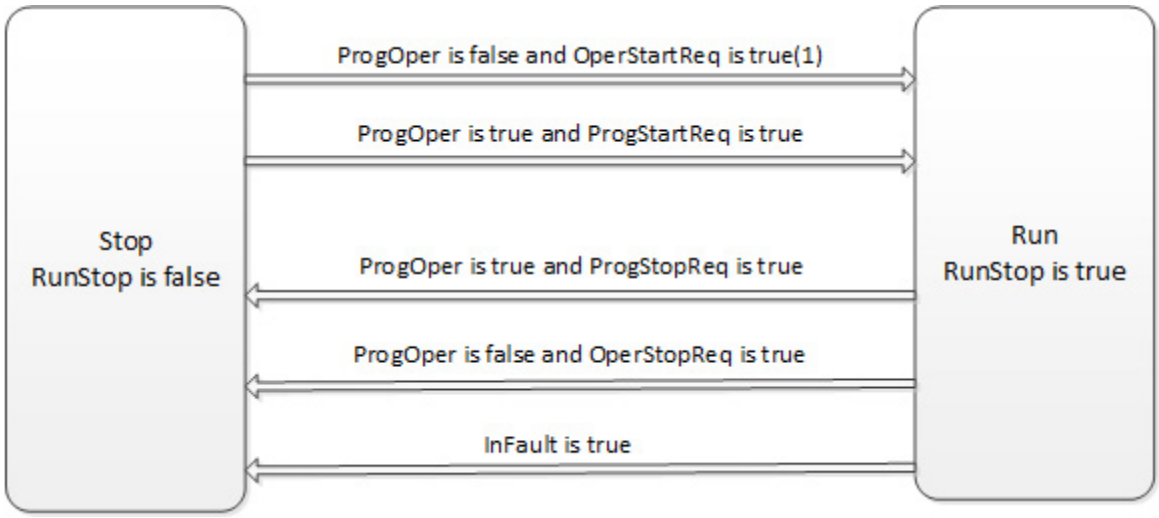
Operating Modes

The following diagram shows how the TOT instruction changes between Program control and Operator control.



(1) The instruction remains in operator control mode when ProgOperReq is true.

The following diagram shows how the TOT instruction changes between Run and Stop modes.



(1) The stop requests take precedence over start requests

(2) The first scan in run after a stop, the totalization is not evaluated, but in $_{n-1}$ is updated.

During the next scan, totalization resumes.

All operator request inputs are cleared to false at the end of each scan. If ProgValueReset is true, all program request inputs are cleared to false at the end of each scan.

Resetting the TOT Instruction

When ProgResetReq transitions to true while ProgOper is true, the following happens:

- OldTotal = Total
- Total = ResetValue
- ProgResetDone is set to true

If ProgResetReq is false and ProgResetDone is true then ProgResetDone is cleared to false

When OperResetReq transitions to true while ProgOper is false, the following happens:

- OldTotal = Total
- Total = ResetValue

Calculating the Totalization

When RunStop is true and LowInCutoffFlag is false, the following equation performs the totalization calculation.

$$Total_n = Total_{n-1} + Gain \times \frac{DeltaT}{2 \times TimeBase} \times (In_n + In_{n-1})$$

where TimeBase is:

Value	Condition
1	TimeBase = 0 (seconds)
60	TimeBase = 1 (minutes)
3600	TimeBase = 2 (hours)
86400	TimeBase = 3 (days)

Determining if Target Values Have Been Reached

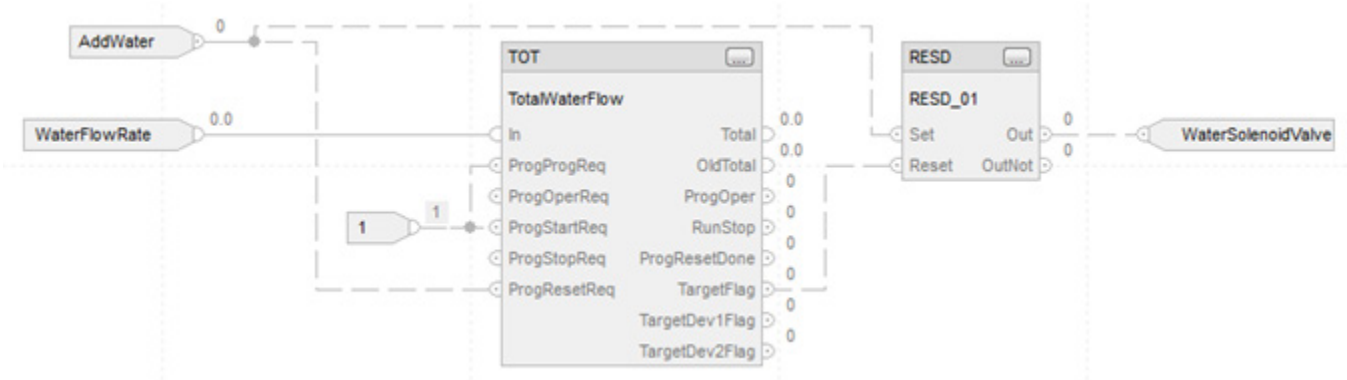
Once the totalization has been calculated, these rules determine whether the target or pre-target values have been reached:

- TargetFlag is true when Total ≥Target
- TargetDev1Flag is true when Total ≥ (Target - TargetDev1)
- TargetDev2Flag is true when Total ≥ (Target - TargetDev2)

Example

In this example, the TOT instruction meters a target quantity of water into a tank and shuts off the flow once the proper amount of water has been added. When the AddWater pushbutton is pressed, the TOT instruction resets and starts totalizing the amount of water flowing into the tank. Once the Target value is reached, the TOT instruction sets the TargetFlag output, which causes the solenoid valve to close. For this example, the TOT instruction was "locked" into Program Run by setting the ProgProgReq and ProgStartReq inputs. This is done for this example because the operator never needs to directly control the TOT instruction.

Function Block



Structured Text

```
TotalWaterFlow.In := WaterFlowRate;
TotalWaterFlow.ProgProgReq := 1;
TotalWaterFlow.ProgStartReq := 1;
TotalWaterFlow.ProgResetReq := AddWater;
TOT(TotalWaterFlow);

RESD_01.Set := AddWater;
RESD_01.Reset := TotalWaterFlow.TargetFlag;
RESD(RESD_01);

WaterSolenoidValve := RESD_01.Out;
```

See also

Function Block Attributes on page 1043

Common Attributes on page 1083

Structured Text Syntax on page 1057

Coordinated Control (CC)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The CC instruction controls a single process variable by manipulating as many as three different control variables. As an option, any of the three outputs can be used as an input to create feed forward action in the controller. The CC calculates the control variables (CV1, CV2, and CV3) in the auto mode based on the PV - SP deviation, internal models, and tuning.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block

Structured Text

CC(CC_tag);

Operands

Structured Text

Operands	Type	Format	Description
CC tag	COORDINATED_ CONTROL	structure	CC Structure

See Structured Text Syntax for more information on the syntax of expressions within structured text.

IMPORTANT Whenever an APC block detects a change in Delta Time (DeltaT), a ModelInit will be performed. For this reason the blocks should only be run in one of the TimingModes in which DeltaT will be constant.

- TimingMode = 0 (Periodic) while executing these function blocks in a Periodic Task
- TimingMode = 1 (Oversample)

In either case, if the Periodic Task time is dynamically changed, or the OversampleDT is dynamically changed, the block will perform a ModelInit.

The following TimingMode setting are not recommended due to jitter in DeltaT:

- TimingMode = 0 (Periodic) while executing these function blocks in a Continuous or Event Task
- TimingMode = 2 (RealTimeSample)

Function Block

Operands	Type	Format	Description
CC tag	COORDINATED CONTROL	structure	CC Structure

Structure

Input Parameters	Data Type	Description	Valid and Default Values
EnableIn	BOOL	Enable Input. If False, the function block will not execute and outputs are not updated.	Default=TRUE
PV	REAL	Scaled process variable input. This value is typically read from an analog input module.	Valid = any float Default = 0.0
PVFault	BOOL	PV bad health indicator. If PV is read from an analog input, then PVFault will normally be controlled by the analog input fault status. If PVFault is TRUE, it indicates an error on the input module, set bit in Status.	FALSE = Good Health Default = FALSE
PVUEMax	REAL	Maximum scaled value for PV. The value of PV and SP that corresponds to 100% span of the Process Variable. If PVEUMax ≤ PVEUMin, set bit in Status.	Valid = PVEUMin < PVEUMax ≤ maximum positive float Default = 100.0
PVUEMin	REAL	Minimum scaled value for PV. The value of PV and SP that corresponds to 0% span of the Process Variable. If PVEUMax ≤ PVEUMin, set bit in Status.	Valid = maximum negative float ≤ PVEUMin < PVEUMax Default = 0.0
SPProg	REAL	SP Program value, scaled in PV units. SP is set to this value when the instruction is in Program control.	Valid = SPLLimit to SPHLimit Default = 0.0
SPOper	REAL	SP Operator value, scaled in PV units. SP set to this value when in Operator control. If value of SPProg or SPOper < SPLLimit or > SPHLimit, set bit in Status and limit value used for SP.	Valid = SPLLimit to SPHLimit Default = 0.0

Input Parameters	Data Type	Description	Valid and Default Values
SPHLimit	REAL	SP high limit value, scaled in PV units. If SPHLimit < SPLLimit or SPHLimit > PVEUMax, set bit in Status.	Valid = SPLLimit to PVEUMax Default = 100.0
SPLLimit	REAL	SP low limit value, scaled in PV units. If SPLLimit < PVEUMin, or SPHLimit < SPLLimit, set bit in Status and limit SP by using the value of SPLLimit.	Valid = PVEUMin to SPHLimit Default = 0.0
CV1Fault	BOOL	Control variable 1 bad health indicator. If CV1EU controls an analog output, then CV1Fault will normally come from the analog output's fault status. If CV1Fault is TRUE, it indicates an error on the output module, set bit in Status.	Default = FALSE FALSE = Good Health
CV2Fault	BOOL	Control variable 2 bad health indicator. If CV2EU controls an analog output, then CV2Fault will normally come from the analog output's fault status. If CV2Fault is TRUE, it indicates an error on the output module, set bit in Status.	Default = FALSE FALSE = Good Health
CV3Fault	BOOL	Control variable 3 bad health indicator. If CV3EU controls an analog output, then CV3Fault will normally come from the analog output's fault status. If CV3Fault is TRUE, it indicates an error on the output module, set bit in Status.	Default = FALSE FALSE = Good Health
CV1InitReq	BOOL	CV1 initialization request. While TRUE, set CV1EU to the value of CVInitValue. This signal will normally be controlled by the In Hold status on the analog output module controlled by CV1EU or from the InitPrimary output of a secondary loop.	Default = FALSE
CV2InitReq	BOOL	CV2 initialization request. While TRUE, set CV2EU to the value of CVInitValue. This signal will normally be controlled by the In Hold status on the analog output module controlled by CV2EU or from the InitPrimary output of a secondary loop.	Default = FALSE

Input Parameters	Data Type	Description	Valid and Default Values
CV3InitReq	BOOL	CV3 initialization request. While TRUE, set CV3EU to the value of CVInitValue. This signal will normally be controlled by the In Hold status on the analog output module controlled by CV32EU or from the InitPrimary output of a secondary loop.	Default = FALSE
CV1InitValue	REAL	CV1EU initialization value, scaled in CV1EU units. When CV1Initializing is TRUE set CV1EU equal to CV1InitValue and CV1 to the corresponding percentage value. CV1InitValue will normally come from the feedback of the analog output controlled by CV1EU or from the setpoint of a secondary loop. The instruction initialization is disabled when CVFaulted or CVEUSpanInv are TRUE.	Valid = any float Default = 0.0
CV2InitValue	REAL	CV2EU initialization value, scaled in CV2EU units. When CV2Initializing is TRUE set CV2EU equal to CV2InitValue and CV2 to the corresponding percentage value. CV2InitValue will normally come from the feedback of the analog output controlled by CV2EU or from the setpoint of a secondary loop. The instruction initialization is disabled when CVFaulted or CVEUSpanInv are TRUE.	Valid = any float Default = 0.0
CV3InitValue	REAL	CV3EU initialization value, scaled in CV3EU units. When CV3Initializing is TRUE set CV3EU equal to CV3InitValue and CV3 to the corresponding percentage value. CV3InitValue will normally come from the feedback of the analog output controlled by CV3EU or from the setpoint of a secondary loop. The instruction initialization is disabled when CVFaulted or CVEUSpanInv are TRUE.	Valid = any float Default = 0.0
CV1Prog	REAL	CV1 Program-Manual value. CV1 is set to this value when in Program control and Manual mode.	Valid = 0.0 through 100.0 Default = 0.0
CV2Prog	REAL	CV2 Program-Manual value. CV2 is set to this value when in Program control and Manual mode.	Valid = 0.0...100.0 Default = 0.0

Input Parameters	Data Type	Description	Valid and Default Values
CV3Prog	REAL	CV3 Program-Manual value. CV3 is set to this value when in Program control and Manual mode.	Valid = 0.0...100.0 Default = 0.0
CV1Oper	REAL	CV1 Operator-Manual value. CV1 is set to this value when in Operator control and Manual mode. If not Operator-Manual mode, set CV1Oper to the value of CV1 at the end of each function block execution. If value of CV1Oper < 0 or > 100, or < CV1LLimit or > CV1HLimit when CVManLimiting is TRUE, set unique Status bit and limit value used for CV.	Valid = 0.0...100.0 Default = 0.0
CV2Oper	REAL	CV2 Operator-Manual value. CV2 is set to this value when in Operator control and Manual mode. If not Operator-Manual mode, set CV2Oper to the value of CV2 at the end of each function block execution. If value of CV2Oper < 0 or > 100, or < CV2LLimit or > CV2HLimit when CVManLimiting is TRUE, set unique Status bit and limit value used for CV.	Valid = 0.0...100.0 Default = 0.0
CV3Oper	REAL	CV3 Operator-Manual value. CV3 is set to this value when in Operator control and Manual mode. If not Operator-Manual mode, set CV3Oper to the value of CV3 at the end of each function block execution. If value of CV3Oper < 0 or > 100, or < CV3LLimit or > CV3HLimit when CVManLimiting is TRUE, set unique Status bit and limit value used for CV.	Valid = 0.0...100.0 Default = 0.0
CV1OverrideValue	REAL	CV1 Override value. CV1 set to this value when in Override mode. This value should correspond to a safe state output of the loop. If value of CV1OverrideValue < 0 or >100, set unique Status bit and limit value used for CV.	Valid = 0.0...100.0 Default = 0.0

Input Parameters	Data Type	Description	Valid and Default Values
CV2OverrideValue	REAL	CV2 Override value. CV2 set to this value when in Override mode. This value should correspond to a safe state output of the loop. If value of CV2OverrideValue < 0 or >100, set unique Status bit and limit value used for CV.	Valid = 0.0...100.0 Default = 0.0
CV3OverrideValue	REAL	CV3 Override value. CV3 set to this value when in Override mode. This value should correspond to a safe state output of the loop. If value of CV3OverrideValue < 0 or >100, set unique Status bit and limit value used for CV.	Valid = 0.0...100.0 Default = 0.0
CV1TrackValue	REAL	CV1 track value. When CVTrackReq is enabled and the CC function block is in Manual mode, the CV1TrackValue will be ignored, and the CC internal model will update its historical data with the CV1Oper or CV1Prog value. When CVTrackReq is enabled and the CC function block is in Auto, the internal model will update its historical data based on the value of CV1TrackValue. The CV1 in this case will be allowed to move as if the CC function block was still controlling the process. This is useful in multiloop selection schemes where you want the CC function block to follow the output of a different controlling algorithm, where you would connect the output of the controlling algorithm into the CV1TrackValue.	Valid = 0.0...100.0 Default = 0.0

Input Parameters	Data Type	Description	Valid and Default Values
CV2TrackValue	REAL	CV2 track value. When CVTrackReq is enabled and the CC function block is in Manual mode, the CV2TrackValue will be ignored, and the CC internal model will update its historical data with the CV2Oper or CV2Prog value. When CVTrackReq is enabled and the CC function block is in Auto, the internal model will update its historical data based on the value of CV2TrackValue. The CV2 in this case will be allowed to move as if the CC function block was still controlling the process. This is useful in multiloop selection schemes where you want the CC function block to follow the output of a different controlling algorithm, where you would connect the output of the controlling algorithm into the CV2TrackValue.	Valid = 0.0...100.0 Default = 0.0
CV3TrackValue	REAL	CV3 track value. When CVTrackReq is enabled and the CC function block is in Manual mode, the CV3TrackValue will be ignored, and the CC internal model will update its historical data with the CV3Oper or CV3Prog value. When CVTrackReq is enabled and the CC function block is in Auto, the internal model will update its historical data based on the value of CV3TrackValue. The CV3 in this case will be allowed to move as if the CC function block was still controlling the process. This is useful in multiloop selection schemes where you want the CC function block to follow the output of a different controlling algorithm, where you would connect the output of the controlling algorithm into the CV3TrackValue.	Valid = 0.0...100.0 Default = 0.0

Input Parameters	Data Type	Description	Valid and Default Values
CVManLimiting	BOOL	Limit CV(n), where (n) can be 1, 2, or 3, in Manual mode request. If Manual mode and CVManLimiting is TRUE, CV will be limited by the CV(n)HLimit and CV(n)LLimit values.	Default = FALSE
CV1EUMax	REAL	Maximum value for CV1EU. The value of CV1EU that corresponds to 100% CV1. If CV1EUMax = CV1EUMin, set bit in Status.	Valid = any float Default = 100.0
CV2EUMax	REAL	Maximum value for CV2EU. The value of CV2EU that corresponds to 100% CV2. If CV2EUMax = CV2EUMin, set bit in Status.	Valid = any float Default = 100.0
CV3EUMax	REAL	Maximum value for CV3EU. The value of CV3EU that corresponds to 100% CV3. If CV3EUMax = CV3EUMin, set bit in Status.	Valid = any float Default = 100.0
CV1EUMin	REAL	Minimum value of CV1EU. The value of CV1EU that corresponds to 0% CV1. If CV1EUMax = CV1EUMin, set bit in Status.	Valid = any float Default = 0.0
CV2EUMin	REAL	Minimum value of CV2EU. The value of CV2EU that corresponds to 0% CV2. If CV2EUMax = CV2EUMin, set bit in Status.	Valid = any float Default = 0.0
CV3EUMin	REAL	Minimum value of CV3EU. The value of CV3EU that corresponds to 0% CV3. If CV3EUMax = CV3EUMin, set bit in Status.	Valid = any float Default = 0.0
CV1HLimit	REAL	CV1 high limit value. This is used to set the CV1HAlarm output. It is also used for limiting CV1 when in Auto mode or in Manual if CVManLimiting is TRUE. If CV1HLimit > 100, if CV1HLimit < CV1LLimit, set bit in Status. If CV1HLimit < CV1LLimit, limit CV1 by using the value of CV1LLimit.	Valid = CV1LLimit < CV1HLimit ≤ 100.0 Default = 100.0
CV2HLimit	REAL	CV2 high limit value. This is used to set the CV2HAlarm output. It is also used for limiting CV2 when in Auto mode or in Manual if CVManLimiting is TRUE. If CV2HLimit > 100, if CV2HLimit < CV2LLimit, set bit in Status. If CV2HLimit < CV2LLimit, limit CV2 by using the value of CV2LLimit.	Valid = CV2LLimit < CV2HLimit ≤ 100.0 Default = 100.0

Input Parameters	Data Type	Description	Valid and Default Values
CV3HLimit	REAL	CV3 high limit value. This is used to set the CV3HAlarm output. It is also used for limiting CV3 when in Auto mode or in Manual if CVManLimiting is TRUE. If CV3HLimit > 100, if CV3HLimit < CV3LLimit, set bit in Status. If CV3HLimit < CV3LLimit, limit CV3 by using the value of CV3LLimit.	Valid = CV3LLimit < CV3HLimit ≤ 100.0 Default = 100.0
CV1LLimit	REAL	CV1 low limit value. This is used to set the CV1LAlarm output. It is also used for limiting CV1 when in Auto mode or in Manual mode if CVManLimiting is TRUE. If CV1LLimit < 0 set bit in Status. If CV1HLimit < CV1LLimit, limit CV by using the value of CV1LLimit.	Valid = 0.0 ≤ CV1LLimit < CV1HLimit Default = 0.0
CV2LLimit	REAL	CV2 low limit value. This is used to set the CV2LAlarm output. It is also used for limiting CV2 when in Auto mode or in Manual mode if CVManLimiting is TRUE. If CV2LLimit < 0 set bit in Status. If CV2HLimit < CV2LLimit, limit CV by using the value of CV2LLimit.	Valid = 0.0 ≤ CV2LLimit < CV1HLimit Default = 0.0
CV3LLimit	REAL	CV3 low limit value. This is used to set the CV3LAlarm output. It is also used for limiting CV3 when in Auto mode or in Manual mode if CVManLimiting is TRUE. If CV3LLimit < 0 set bit in Status. If CV3HLimit < CV3LLimit, limit CV by using the value of CVLLimit.	Valid = 0.0 ≤ CV3LLimit < CV1HLimit Default = 0.0
CV1ROCPosLimit	REAL	CV1 rate of change limit, in percent per second. Rate of change limiting is only used when in Auto mode or in Manual mode if CVManLimiting is TRUE. A value of zero disables CV1 ROC limiting. If value of CV1ROCLimit < 0, set bit in Status and disable CV1 ROC limiting.	Valid = 0.0 to maximum positive float Default = 0.0
CV2ROCPosLimit	REAL	CV2 rate of change limit, in percent per second. Rate of change limiting is only used when in Auto mode or in Manual mode if CVManLimiting is TRUE. A value of zero disables CV2 ROC limiting. If value of CV2ROCLimit < 0, set bit in Status and disable CV2 ROC limiting.	Valid = 0.0 to maximum positive float Default = 0.0

Input Parameters	Data Type	Description	Valid and Default Values
CV3ROCPosLimit	REAL	CV3 rate of change limit, in percent per second. Rate of change limiting is only used when in Auto mode or in Manual mode if CVManLimiting is TRUE. A value of zero disables CV3 ROC limiting. If value of CV3ROCLimit < 0, set bit in Status and disable CV3 ROC limiting.	Valid = 0.0 to maximum positive float Default = 0.0
CV1ROCNegLimit	REAL	CV1 rate of change limit, in percent per second. Rate of change limiting is only used when in Auto mode or in Manual mode if CVManLimiting is TRUE. A value of zero disables CV1 ROC limiting. If value of CV1ROCLimit < 0, set bit in Status and disable CV1 ROC limiting.	Valid = 0.0 to maximum positive float Default = 0.0
CV2ROCNegLimit	REAL	CV2 rate of change limit, in percent per second. Rate of change limiting is only used when in Auto mode or in Manual mode if CVManLimiting is TRUE. A value of zero disables CV2 ROC limiting. If value of CV2ROCLimit < 0, set bit in Status and disable CV2 ROC limiting.	Valid = 0.0 to maximum positive float Default = 0.0
CV3ROCNegLimit	REAL	CV3 rate of change limit, in percent per second. Rate of change limiting is only used when in Auto mode or in Manual mode if CVManLimiting is TRUE. A value of zero disables CV3 ROC limiting. If value of CV3ROCLimit < 0, set bit in Status and disable CV3 ROC limiting.	Valid = 0.0 to maximum positive float Default = 0.0
CV1HandFB	REAL	CV1 HandFeedback value. CV1 set to this value when in Hand mode and CV1HandFBFault is FALSE (good health). This value would typically come from the output of a field mounted hand/auto station and would be used to generate a bumpless transfer out of Hand mode. If value of CV1HandFB < 0 or > 100, set unique Status bit and limit value used for CV1.	Valid = 0.0...100.0 Default = 0.0

Input Parameters	Data Type	Description	Valid and Default Values
CV2HandFB	REAL	CV2 HandFeedback value. CV2 set to this value when in Hand mode and CV2HandFBFault is FALSE (good health). This value would typically come from the output of a field mounted hand/auto station and would be used to generate a bumpless transfer out of Hand mode. If value of CV2HandFB < 0 or > 100, set unique Status bit and limit value used for CV2.	Valid = 0.0...100.0 Default = 0.0
CV3HandFB	REAL	CV3 HandFeedback value. CV3 set to this value when in Hand mode and CV3HandFBFault is FALSE (good health). This value would typically come from the output of a field mounted hand/auto station and would be used to generate a bumpless transfer out of Hand mode. If value of CV3HandFB < 0 or > 100, set unique Status bit and limit value used for CV3.	Valid = 0.0...100.0 Default = 0.0
CV1HandFBFault	BOOL	CV1HandFB value bad health indicator. If the CV1HandFB value is read from an analog input, then CV1HandFBFault will normally be controlled by the status of the analog input channel. If CV1HandFBFault is TRUE, it indicates an error on the input module, set bit in Status.	FALSE = Good Health Default = FALSE
CV2HandFBFault	BOOL	CV2HandFB value bad health indicator. If the CV2HandFB value is read from an analog input, then CV2HandFBFault will normally be controlled by the status of the analog input channel. If CV2HandFBFault is TRUE, it indicates an error on the input module, set bit in Status.	FALSE = Good Health Default = FALSE
CV3HANDFBFault	BOOL	CV3HandFB value bad health indicator. If the CV3HandFB value is read from an analog input, then CV3HandFBFault will normally be controlled by the status of the analog input channel. If CV3HandFBFault is TRUE, it indicates an error on the input module, set bit in Status.	FALSE = Good Health Default = FALSE
CV1Target	REAL	Target value for CV1.	Valid = 0.0...100.0 Default = 0.0

Input Parameters	Data Type	Description	Valid and Default Values
CV2Target	REAL	Target value for CV2.	Valid = 0.0...100.0 Default = 0.0
CV3Target	REAL	Target value for CV3.	Valid = 0.0...100.0 Default = 0.0
CV1WindupHIn	BOOL	CV1Windup high request. When TRUE, CV1 will not be allowed to increase in value. This signal will typically be the CV1WindupHOut output from a secondary loop.	Default = FALSE
CV2WindupHIn	BOOL	CV2Windup high request. When TRUE, CV2 will not be allowed to increase in value. This signal will typically be the CV2WindupHOut output from a secondary loop.	Default = FALSE
CV3WindupHIn	BOOL	CV3Windup high request. When TRUE, CV3 will not be allowed to increase in value. This signal will typically be the CV3WindupHOut output from a secondary loop.	Default = FALSE
CV1WindupLIn	BOOL	CV1 Windup low request. When TRUE, CV1 will not be allowed to decrease in value. This signal will typically be the CV1WindupLOut output from a secondary loop.	Default = FALSE
CV2WindupLIn	BOOL	CV2 Windup low request. When TRUE, CV2 will not be allowed to decrease in value. This signal will typically be the CV2WindupLOut output from a secondary loop.	Default = FALSE
CV3WindupLIn	BOOL	CV3 Windup low request. When TRUE, CV3 will not be allowed to decrease in value. This signal will typically be the CV3WindupLOut output from a secondary loop.	Default = FALSE
GainEUSpan	BOOL	CVxModelGain units can be represented as EU or % of span. Set to interpret ModelGain as EU, reset to interpret ModelGain as % of Span.	Default = 0
CV1ProcessGainSign	BOOL	Used only for Autotuning. Sign of the process gain (Delta PV/Delta CV1). Set to indicate a negative process gain (increase in output causes a decrease in PV). Reset to indicate a positive process gain (increase in output causes an increase in PV).	Default = FALSE

Input Parameters	Data Type	Description	Valid and Default Values
CV2ProcessGainSign	BOOL	Used only for Autotuning. Sign of the process gain (Delta PV/Delta CV2). Set to indicate a negative process gain (increase in output causes a decrease in PV). Reset to indicate a positive process gain (increase in output causes an increase in PV).	Default = FALSE
CV3ProcessGainSign	BOOL	Used only for Autotuning. Sign of the process gain (Delta PV/Delta CV3). Set to indicate a negative process gain (increase in output causes a decrease in PV). Reset to indicate a positive process gain (increase in output causes an increase in PV).	Default = FALSE
ProcessType	DINT	Process type selection (1=Integrating, 0=non-integrating)	Default = 0
CV1ModelGain	REAL	The internal model gain parameter for CV1. Enter a positive or negative gain depending on process direction.	Valid = maximum negative float –> maximum positive float Default = 0.0
CV2ModelGain	REAL	The internal model gain parameter for CV2. Enter a positive or negative gain depending on process direction.	Valid = maximum negative float –> maximum positive float Default = 0.0
CV3ModelGain	REAL	The internal model gain parameter for CV3. Enter a positive or negative gain depending on process direction.	Valid = maximum negative float –> maximum positive float Default = 0.0
CV1ModelTC	REAL	The internal model time constant for CV1 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV2ModelTC	REAL	The internal model time constant for CV2 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV3ModelTC	REAL	The internal model time constant for CV3 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV1ModelDT	REAL	The internal model deadtime for CV1 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV2ModelDT	REAL	The internal model deadtime for CV2 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV3ModelDT	REAL	The internal model deadtime for CV3 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0

Input Parameters	Data Type	Description	Valid and Default Values
CV1RespTC	REAL	The tuning parameter that determines the speed of the control variable action for CV1 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV2RespTC	REAL	The tuning parameter that determines the speed of the control variable action for CV2 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV3RespTC	REAL	The tuning parameter that determines the speed of the control variable action for CV3 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
Act1stCV	DINT	The first CV to act to compensate for PV-SP deviation. 1=CV1, 2=CV2, 3=CV3	Valid = 1-3 Default = 1
Act2ndCV	DINT	The second CV to act to compensate for PV-SP deviation. 1=CV1, 2=CV2, 3=CV3	Valid = 1-3 Default = 2
Act3rdCV	DINT	The third CV to act to compensate for PV-SP deviation. 1=CV1, 2=CV2, 3=CV3	Valid = 1-3 Default = 3
Target1stCV	DINT	The CV with top priority to be driven to its target value. 1=CV1, 2=CV2, 3=CV3	Valid = 1-3 Default = 1
Target2ndCV	DINT	The CV with second highest priority to be driven to its target value. 1=CV1, 2=CV2, 3=CV3	Valid = 1-3 Default = 2
Target3rdCV	DINT	The CV with third highest priority to be driven to its target value. 1=CV1, 2=CV2, 3=CV3	Valid = 1-3 Default = 3
PVTracking	BOOL	SP track PV request. Set TRUE to enable SP to track PV. Ignored when in Auto modes. SP will only track PV when all three outputs are in manual. As soon as any output returns to Auto, PVTracking stops.	Default = FALSE
CVTrackReq	BOOL	CV Track request. Set true to enable CV Tracking when autotune is OFF. Ignored in Hand and Override mode.	Default = FALSE
ManualAfterInit	BOOL	Manual mode after initialization request. When TRUE, the appropriate CV(n), where (n) can be 1, 2, or 3, will be placed in Manual mode when CV(n)Initializing is set TRUE unless the current mode is Override or Hand. When ManualAfterInit is FALSE, the CV(n) mode will not be changed.	Default = FALSE

Input Parameters	Data Type	Description	Valid and Default Values
ProgProgReq	BOOL	Program Program Request. Set TRUE by the user program to request Program control. Ignored if ProgOperReq is TRUE. Holding this TRUE and ProgOperReq FALSE can be used to lock the function block into program control. When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
ProgOperReq	BOOL	Program Operator Request. Set TRUE by the user program to request Operator control. Holding this TRUE can be used to lock the function block into operator control. If value of HandFB < 0 or > 100, set unique Status bit and limit value used for CV.	Default = FALSE
ProgCV1AutoReq	BOOL	Program-Auto mode request for CV1. Set TRUE by the user program to request Auto mode. If value of CV1HandFB < 0 or > 100, set unique Status bit and limit value used for CV1.	Default = FALSE
ProgCV2AutoReq	BOOL	Program-Auto mode request for CV2. Set TRUE by the user program to request Auto mode. If value of CV2HandFB < 0 or > 100, set unique Status bit and limit value used for CV2.	Default = FALSE
ProgCV3AutoReq	BOOL	Program-Auto mode request for CV3. Set TRUE by the user program to request Auto mode. If value of CV3HandFB < 0 or > 100, set unique Status bit and limit value used for CV3.	Default = FALSE
ProgCV1ManualReq	BOOL	Program-Manual mode request for CV1. Set TRUE by the user program to request Manual mode. If value of CV1HandFB < 0 or > 100, set unique Status bit and limit value used for CV1.	Default = FALSE

Input Parameters	Data Type	Description	Valid and Default Values
ProgCV2ManualReq	BOOL	Program-Manual mode request for CV2. Set TRUE by the user program to request Manual mode. If value of CV2HandFB < 0 or > 100, set unique Status bit and limit value used for CV2.	Default = FALSE
ProgCV3ManualReq	BOOL	Program-Manual mode request for CV3. Set TRUE by the user program to request Manual mode. If value of CV3HandFB < 0 or > 100, set unique Status bit and limit value used for CV3.	Default = FALSE
ProgCV1OverrideReq	BOOL	Program-Override mode request for CV1. Set TRUE by the user program to request Override mode. If value of CV1HandFB < 0 or > 100, set unique Status bit and limit value used for CV1.	Default = FALSE
ProgCV2OverrideReq	BOOL	Program-Override mode request for CV2. Set TRUE by the user program to request Override mode. If value of CV2HandFB < 0 or > 100, set unique Status bit and limit value used for CV2.	Default = FALSE
ProgCV3OverrideReq	BOOL	Program-Override mode request for CV3. Set TRUE by the user program to request Override mode. If value of CV3HandFB < 0 or > 100, set unique Status bit and limit value used for CV3.	Default = FALSE
ProgCV1HandReq	BOOL	Program-Hand mode request for CV1. Set TRUE by the user program to request Hand mode. This value will usually be read as a digital input from a hand/auto station.	Default = FALSE
ProgCV2HandReq	BOOL	Program-Hand mode request for CV2. Set TRUE by the user program to request Hand mode. This value will usually be read as a digital input from a hand/auto station.	Default = FALSE
ProgCV3HandReq	BOOL	Program-Hand mode request for CV3. Set TRUE by the user program to request Hand mode. This value will usually be read as a digital input from a hand/auto station.	Default = FALSE

Input Parameters	Data Type	Description	Valid and Default Values
OperProgReq	BOOL	Operator Program Request. Set TRUE by the operator interface to request Program control. The function block resets this parameter to FALSE.	Default = FALSE
OperOperReq	BOOL	Operator Operator Request. Set TRUE by the operator interface to request Operator control. The function block will reset this parameter to FALSE.	Default = FALSE
OperCV1AutoReq	BOOL	Operator-Auto mode request for CV1. Set TRUE by the operator interface to request Auto mode. The function block will reset this parameter to FALSE.	Default = FALSE
OperCV2AutoReq	BOOL	Operator-Auto mode request for CV2. Set TRUE by the operator interface to request Auto mode. The function block will reset this parameter to FALSE.	Default = FALSE
OperCV3AutoReq	BOOL	Operator-Auto mode request for CV3. Set TRUE by the operator interface to request Auto mode. The function block will reset this parameter to FALSE.	Default = FALSE
OperCV1ManualReq	BOOL	Operator-Manual mode request for CV1. Set TRUE by the operator interface to request Manual mode. The function block resets this parameter to FALSE.	Default = FALSE
OperCV2ManualReq	BOOL	Operator-Manual mode request for CV2. Set TRUE by the operator interface to request Manual mode. The function block resets this parameter to FALSE.	Default = FALSE
OperCV3ManualReq	BOOL	Operator-Manual mode request for CV3. Set TRUE by the operator interface to request Manual mode. The function block resets this parameter to FALSE.	Default = FALSE

Input Parameters	Data Type	Description	Valid and Default Values
ProgValueReset	BOOL	Reset Program control values. When TRUE, the Prog_xxx_Req inputs are reset to FALSE. When TRUE and Program control, set SPProg equal to SP and CVxProg equal to CVx. When ProgValueReset is TRUE, the instruction sets this input to FALSE.	Default = FALSE
TimingMode	DINT	Selects Time Base Execution mode. Value/Description 0 = Periodic mode 1 = Oversample mode 2 = Real time sampling mode For more information about timing modes, see Function Block Attributes.	Valid = 0...2 Default = 0
OverSampleDT	REAL	Execution time for Oversample mode.	Valid = 0...4194.303 seconds Default = 0
RTSTime	DINT	Module update period for Real Time Sampling mode.	Valid = 1...32,767 1 count = 1 ms
RTSTimeStamp	DINT	Module time stamp value for Real Time Sampling mode.	Valid = 0...32,767 (wraps from 32,767...0) 1 count = 1 ms
PVTuneLimit	REAL	PV tuning limit scaled in the PV units. When Autotune is running and predicted PV exceeds this limit, the tuning will be aborted.	Valid = any float Default=0
AtuneTimeLimit	REAL	Maximum time for autotune to complete following the CV step change. When autotune exceeds this time, tuning will be aborted.	Valid range: any float > 0. Default = 60 minutes
NoiseLevel	DINT	An estimate of the noise level expected on the PV to compensate for it during tuning. The selections are: 0=low, 1=medium, 2=high	Range: 0...2 Default=1
CV1StepSize	REAL	CV1 step size in percent for the tuning step test. Step size is directly added to CV1 subject to high/low limiting.	Range: -100% ... 100% Default=10%
CV2StepSize	REAL	CV2 step size in percent for the tuning step test. Step size is directly added to CV2 subject to high/low limiting.	Range: -100% ... 100% Default=10%
CV3StepSize	REAL	CV3 step size in percent for the tuning step test. Step size is directly added to CV3 subject to high/low limiting.	Range: -100% ... 100% Default=10%

Input Parameters	Data Type	Description	Valid and Default Values
CV1ResponseSpeed	DINT	Desired speed of closed loop response for CV1. Slow response: ResponseSpeed=0 Medium response: ResponseSpeed=1 Fast response: ResponseSpeed=2. If ResponseSpeed is less than 0, Slow response is used. If ResponseSpeed is greater than 2, Fast response is used.	Range: 0...2 Default=1
CV2ResponseSpeed	DINT	Desired speed of closed loop response for CV2. Slow response: ResponseSpeed=0 Medium response: ResponseSpeed=1 Fast response: ResponseSpeed=2. If ResponseSpeed is less than 0, Slow response is used. If ResponseSpeed is greater than 2, Fast response is used.	Range: 0...2 Default=1
CV3ResponseSpeed	DINT	Desired speed of closed loop response for CV3. Slow response: ResponseSpeed=0 Medium response: ResponseSpeed=1 Fast response: ResponseSpeed=2. If ResponseSpeed is less than 0, Slow response is used. If ResponseSpeed is greater than 2, Fast response is used.	Range: 0...2 Default=1
CV1ModelInit	BOOL	Internal model initialization switch for CV1. Refer to topic CC Function Block Model Initialization in CC Function Block Tuning.	Default = FALSE
CV2ModelInit	BOOL	Internal model initialization switch for CV2. Refer to topic CC Function Block Model Initialization in CC Function Block Tuning.	Default = FALSE
CV3ModelInit	BOOL	Internal model initialization switch for CV3. Refer to topic CC Function Block Model Initialization in CC Function Block Tuning.	Default = FALSE
Factor	REAL	Non-integrating model approximation factor. Only used for integrating process types.	Default = 100

Input Parameters	Data Type	Description	Valid and Default Values
AtuneCV1Start	BOOL	Start Autotune request for CV1. Set True to initiate auto tuning of the CV1 output. Ignored when CV1 is not in Manual mode. The function block resets the input to FALSE.	Default = FALSE
AtuneCV2Start	BOOL	Start Autotune request for CV2. Set True to initiate auto tuning of the CV2 output. Ignored when CV2 is not in Manual mode. The function block resets the input to FALSE.	Default = FALSE
AtuneCV3Start	BOOL	Start Autotune request for CV3. Set True to initiate auto tuning of the CV3 output. Ignored when CV3 is not in Manual mode. The function block resets the input to FALSE.	Default = FALSE
AtuneCV1UseModel	BOOL	Use Autotune model request for CV1. Set True to replace the current model parameters with the calculated Autotune model parameters. The function block resets the input parameter to FALSE.	Default = FALSE
AtuneCV2UseModel	BOOL	Use Autotune model request for CV2. Set True to replace the current model parameters with the calculated Autotune model parameters. The function block resets the input parameter to FALSE.	Default = FALSE
AtuneCV3UseModel	BOOL	Use Autotune model request for CV3. Set True to replace the current model parameters with the calculated Autotune model parameters. The function block resets the input parameter to FALSE.	Default = FALSE
AtuneCV1Abort	BOOL	Abort Autotune request for CV1. Set True to abort the auto tuning of CV1 output. The function block resets input parameter to FALSE.	Default = FALSE
AtuneCV2Abort	BOOL	Abort Autotune request for CV2. Set True to abort the auto tuning of CV2 output. The function block resets input parameter to FALSE.	Default = FALSE
AtuneCV3Abort	BOOL	Abort Autotune request for CV3. Set True to abort the auto tuning of CV3 output. The function block resets input parameter to FALSE.	Default = FALSE

Output Parameters	Data Type	Description	Valid and Default Values
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if CV1EU, CV2EU or CV3EU overflows.	
CV1EU	REAL	Scaled control variable output for CV1. Scaled by using CV1EUMax and CV1EUMin, where CV1EUMax corresponds to 100% and CV1EUMin corresponds to 0%. This output is typically used to control an analog output module or a secondary loop. CV1EU = (CV1 * CV1EUSpan / 100) + CV1EUMin CV1EU span calculation: CV1EUSpan = (CV1EUMax – CV1EUMin)	
CV2EU	REAL	Scaled control variable output for CV2. Scaled by using CV2EUMax and CV2EUMin, where CV2EUMax corresponds to 100% and CV2EUMin corresponds to 0%. This output is typically used to control an analog output module or a secondary loop. CV2EU = (CV2 * CV2EUSpan / 100) + CV2EUMin CV2EU span calculation: CV2EUSpan = (CV2EUMax – CV2EUMin)	
CV3EU	REAL	Scaled control variable output for CV3. Scaled by using CV3EUMax and CV3EUMin, where CV3EUMax corresponds to 100% and CV3EUMin corresponds to 0%. This output is typically used to control an analog output module or a secondary loop. CV3EU = (CV3 * CV3EUSpan / 100) + CV3EUMin CV3EU span calculation: CV3EUSpan = (CV3EUMax – CV3EUMin)	
CV1	REAL	Control variable 1 output. This value will always be expressed as 0...100%. CV1 is limited by CV1HLimit and CV1LLimit when in Auto mode or in Manual mode if CVManLimiting is TRUE; otherwise limited by 0 and 100%.	

Output Parameters	Data Type	Description	Valid and Default Values
CV2	REAL	Control variable 2 output. This value will always be expressed as 0...100%. CV2 is limited by CV2HLimit and CV2LLimit when in Auto mode or in Manual mode if CVManLimiting is TRUE; otherwise limited by 0 and 100%.	
CV3	REAL	Control variable 3 output. This value will always be expressed as 0...100%. CV3 is limited by CV3HLimit and CV3LLimit when in Auto mode or in Manual mode if CVManLimiting is TRUE; otherwise limited by 0 and 100%.	
DeltaCV1	REAL	Difference between the Current CV1 and the previous CV1 (Current CV1 - previous CV1).	
DeltaCV2	REAL	Difference between the Current CV2 and the previous CV2 (Current CV2 - previous CV2).	
DeltaCV3	REAL	Difference between the Current CV3 and the previous CV3 (Current CV3 - previous CV3).	
CV1Initializing	BOOL	Initialization mode indicator for CV1. Set TRUE when CV1InitReq or function blockFirstScan are TRUE, or on a TRUE to FALSE transition of CV1Fault (bad to good). CV1Initializing is set FALSE after the function block has been initialized and CV1InitReq is no longer TRUE.	
CV2nitializing	BOOL	Initialization mode indicator for CV2. Set TRUE when CV2InitReq, function blockFirstScan or OLCFirstRun, are TRUE, or on a TRUE to FALSE transition of CV2Fault (bad to good). CV2Initializing is set FALSE after the function block has been initialized and CV2InitReq is no longer TRUE.	
CV3initializing	BOOL	Set TRUE when CV3InitReq, function blockFirstScan or OLCFirstRun, are TRUE, or on a TRUE to FALSE transition of CV3Fault (bad to good). CV3Initializing is set FALSE after the function block has been initialized and CV3InitReq is no longer TRUE.	

Output Parameters	Data Type	Description	Valid and Default Values
CV1HAlarm	BOOL	CV1 high alarm indicator. TRUE when the calculated value for CV1 > 100 or CV1HLimit.	
CV12HAlarm	BOOL	CV2 high alarm indicator. TRUE when the calculated value for CV2 > 100 or CV2HLimit.	
CV3HAlarm	BOOL	CV3 high alarm indicator. TRUE when the calculated value for CV3 > 100 or CV3HLimit.	
CV1LAlarm	BOOL	CV1 low alarm indicator. TRUE when the calculated value for CV1 < 0 or CV1LLimit.	
CV2LAlarm	BOOL	CV2 low alarm indicator. TRUE when the calculated value for CV2 < 0 or CV2LLimit.	
CV3LAlarm	BOOL	CV3 low alarm indicator. TRUE when the calculated value for CV3 < 0 or CV3LLimit.	
CV1ROCPosAlarm	BOOL	CV1 rate of change alarm indicator. TRUE when the calculated rate of change for CV1 exceeds CV1ROCPosLimit.	
CV2ROCPosAlarm	BOOL	CV2 rate of change alarm indicator. TRUE when the calculated rate of change for CV2 exceeds CV2ROCPosLimit.	
CV3ROCPosAlarm	BOOL	CV3 rate of change alarm indicator. TRUE when the calculated rate of change for CV3 exceeds CV3ROCPosLimit.	
CV1ROCNegAlarm	BOOL	CV1 rate of change alarm indicator. TRUE when the calculated rate of change for CV1 exceeds CV1ROCNegLimit.	
CV2ROCNegAlarm	BOOL	CV2 rate of change alarm indicator. TRUE when the calculated rate of change for CV2 exceeds CV2ROCNegLimit.	
CV3ROCNegAlarm	BOOL	CV3 rate of change alarm indicator. TRUE when the calculated rate of change for CV3 exceeds CV3ROCNegLimit.	
SP	REAL	Current setpoint value. The value of SP is used to control CV when in the Auto or the PV Tracking mode, scaled in PV units.	
SPPercent	REAL	The value of SP expressed in percent of span of PV. SPPercent = ((SP – PVEUMin) * 100) / PVSpan. PV Span calculation: PVSpan = (PVEUMax – PVEUMin)	

Output Parameters	Data Type	Description	Valid and Default Values
SPHAlarm	BOOL	SP high alarm indicator. TRUE when the SP ≥ SPHLimit.	
SPLAlarm	BOOL	SP low alarm indicator. TRUE when the SP ≤ SPLLimit.	
PVPercent	REAL	PV expressed in percent of span. PVPercent = ((PV – PVEUMin) * 100) / PVSpan PV Span calculation: PVSpan = (PVEUMax – PVEUMin)	
E	REAL	Process error. Difference between SP and PV, scaled in PV units.	
EPercent	REAL	The error expressed as a percent of span.	
CV1WindupHOut	BOOL	CV1 Windup high indicator. TRUE when either a SP high or CV1 high/low limit has been reached. This signal will typically be used by the WindupHIn input to limit the windup of the CV1 output on a primary loop.	
CV2WindupHOut	BOOL	CV2 Windup high indicator. TRUE when either a SP high or CV2 high/low limit has been reached. This signal will typically be used by the WindupHIn input to limit the windup of the CV2 output on a primary loop.	
CV3WindupHOut	BOOL	CV3 Windup high indicator. TRUE when either a SP high or CV3 high/low limit has been reached. This signal will typically be used by the WindupHIn input to limit the windup of the CV3 output on a primary loop.	
CV1WindupLOut	BOOL	CV1 Windup low indicator. TRUE when either a SP or CV1 high/low limit has been reached. This signal will typically be used by the WindupLIn input to limit the windup of the CV1 output on a primary loop.	
CV2WindupLOut	BOOL	CV2 Windup low indicator. TRUE when either a SP or CV2 high/low limit has been reached. This signal will typically be used by the WindupLIn input to limit the windup of the CV2 output on a primary loop.	

Output Parameters	Data Type	Description	Valid and Default Values
CV3WindupLOut	BOOL	CV3 Windup low indicator. TRUE when either a SP or CV3 high/low limit has been reached. This signal will typically be used by the WindupLIn input to limit the windup of the CV3 output on a primary loop.	
ProgOper	BOOL	Program/Operator control indicator. TRUE when in Program control. FALSE when in Operator control.	
CV1Auto	BOOL	Auto mode indicator for CV1. TRUE when CV1 in the Auto mode.	
CV2Auto	BOOL	Auto mode indicator for CV2. TRUE when CV2 in the Auto mode.	
CV2Auto	BOOL	Auto mode indicator for CV3. TRUE when CV3 in the Auto mode.	
CV1Manual	BOOL	Manual mode indicator CV1. TRUE when CV1 in the Manual mode.	
CV2Manual	BOOL	Manual mode indicator CV2. TRUE when CV2 in the Manual mode.	
CV3Manual	BOOL	Manual mode indicator CV3. TRUE when CV3 in the Manual mode.	
CV1Override	BOOL	Override mode indicator for CV1. TRUE when CV1 in the Override mode.	
CV2Override	BOOL	Override mode indicator for CV2. TRUE when CV2 in the Override mode.	
CV3Override	BOOL	Override mode indicator for CV3. TRUE when CV3 in the Override mode.	
CV1Hand	BOOL	Hand mode indicator for CV1. TRUE when CV1 in the Hand mode.	
CV2Hand	BOOL	Hand mode indicator for CV2. TRUE when CV2 in the Hand mode.	
CV3Hand	BOOL	Hand mode indicator for CV3. TRUE when CV3 in the Hand mode.	
DeltaT	REAL	Elapsed time between updates in seconds.	
CV1StepSizeUsed	REAL	Actual CV1 step size used for tuning.	
CV2StepSizeUsed	REAL	Actual CV2 step size used for tuning.	
CV3StepSizeUsed	REAL	Actual CV3 step size used for tuning.	

Output Parameters	Data Type	Description	Valid and Default Values
CV1GainTuned	REAL	The calculated value of the internal model gain for CV1 after tuning is completed.	
CV2GainTuned	REAL	The calculated value of the internal model gain for CV2 after tuning is completed.	
CV3GainTuned	REAL	The calculated value of the internal model gain for CV3 after tuning is completed.	
CV1TCTuned	REAL	The calculated value of the internal model time constant for CV1 after tuning is completed.	
CV2TCTuned	REAL	The calculated value of the internal model time constant for CV2 after tuning is completed.	
CV3TCTuned	REAL	The calculated value of the internal model time constant for CV3 after tuning is completed.	
CV1DTTuned	REAL	The calculated value of the internal model deadtime for CV1 after tuning is completed.	
CV2DTTuned	REAL	The calculated value of the internal model deadtime for CV2 after tuning is completed.	
CV3DTTuned	REAL	The calculated value of the internal model deadtime for CV3 after tuning is completed.	
CV1RespTCTunedS	REAL	The calculated value of the control variable time constant in slow response speed for CV1 after tuning is completed.	
CV2RespTCTunedS	REAL	The calculated value of the control variable time constant in slow response speed for CV2 after tuning is completed.	
CV3RespTCTunedS	REAL	The calculated value of the control variable time constant in slow response speed for CV3 after tuning is completed.	
CV1RespTCTunedM	REAL	The calculated value of the control variable time constant in medium response speed for CV1 after tuning is completed.	
CV2RespTCTunedM	REAL	The calculated value of the control variable time constant in medium response speed for CV2 after tuning is completed.	
CV3RespTCTunedM	REAL	The calculated value of the control variable time constant in medium response speed for CV3 after tuning is completed.	

Output Parameters	Data Type	Description	Valid and Default Values
CV1RespTCTunedF	REAL	The calculated value of the control variable time constant in fast response speed for CV1 after tuning is completed.	
CV2RespTCTunedF	REAL	The calculated value of the control variable time constant in fast response speed for CV2 after tuning is completed.	
CV3RespTCTunedF	REAL	The calculated value of the control variable time constant in fast response speed for CV3 after tuning is completed.	
AtuneCV1On	BOOL	Set True when auto tuning for CV1 has been initiated.	
AtuneCV2On	BOOL	Set True when auto tuning for CV2 has been initiated.	
AtuneCV3On	BOOL	Set True when auto tuning for CV3 has been initiated.	
AtuneCV1Done	BOOL	Set True when auto tuning for CV1 has completed successfully.	
AtuneCV2Done	BOOL	Set True when auto tuning for CV2 has completed successfully.	
AtuneCV3Done	BOOL	Set True when auto tuning for CV3 has completed successfully.	
AtuneCV1Aborted	BOOL	Set True when auto tuning for CV1 has been aborted by user or due to errors that occurred during the auto tuning operation.	
AtuneCV2Aborted	BOOL	Set True when auto tuning for CV2 has been aborted by user or due to errors that occurred during the auto tuning operation.	
AtuneCV3Aborted	BOOL	Set True when auto tuning for CV3 has been aborted by user or due to errors that occurred during the auto tuning operation.	
AtuneCV1Status	DINT	Indicates the tuning status for CV1.	
AtuneCV2Status	DINT	Indicates the tuning status for CV2.	
AtuneCV3Status	DINT	Indicates the tuning status for CV3.	
AtuneCV1Fault	BOOL	CV1 Autotune has generated any of the following faults.	
AtuneCV2Fault	BOOL	CV2 Autotune has generated any of the following faults.	
AtuneCV3Fault	BOOL	CV3 Autotune has generated any of the following faults.	
AtuneCV1PVOutOfLimit	BOOL	Either PV or the deadtime-step ahead prediction of PV exceeds PVTuneLimit during CV1 Autotuning. When True, CV1 Autotuning is aborted.	

Output Parameters	Data Type	Description	Valid and Default Values
AtuneCV2PVOutOfLimit	BOOL	Either PV or the deadtime-step ahead prediction of PV exceeds PVTuneLimit during CV2 Autotuning. When True, CV2 Autotuning is aborted.	
AtuneCV3PVOutOfLimit	BOOL	Either PV or the deadtime-step ahead prediction of PV exceeds PVTuneLimit during CV3 Autotuning. When True, CV3 Autotuning is aborted.	
AtuneCV1ModeInv	BOOL	The CC mode was not Manual at start of Autotuning or the CC mode was changed from Manual during CV1 Autotuning. When True, CV1 Autotuning is not started or is aborted.	
AtuneCV2ModeInv	BOOL	The CC mode was not Manual at start of Autotuning or the CC mode was changed from Manual during CV2 Autotuning. When True, CV2 Autotuning is not started or is aborted.	
AtuneCV3ModeInv	BOOL	The CC mode was not Manual at start of Autotuning or the CC mode was changed from Manual during CV3 Autotuning. When True, CV3 Autotuning is not started or is aborted.	
AtuneCV1WindupFault	BOOL	CV1WindupHIn or CV1WindupLIn is True at start of CV1 Autotuning or during CV1 Autotuning. When True, CV1 Autotuning is not started or is aborted.	
AtuneCV2WindupFault	BOOL	CV2WindupHIn or CV2WindupLIn is True at start of CV1 Autotuning or during CV2 Autotuning. When True, CV2 Autotuning is not started or is aborted.	
AtuneCV3WindupFault	BOOL	CV3WindupHIn or CV3WindupLIn is True at start of CV3 Autotuning or during CV3 Autotuning. When True, CV3 Autotuning is not started or is aborted.	
AtuneCV1StepSize0	BOOL	CV1StepSizeUsed = 0 at start of CV1 Autotuning. When True, CV1 Autotuning is not started.	
AtuneCV2StepSize0	BOOL	CV2StepSizeUsed = 0 at start of CV2 Autotuning. When True, CV2 Autotuning is not started.	
AtuneCV3StepSize0	BOOL	CV3StepSizeUsed = 0 at start of CV3 Autotuning. When True, CV3 Autotuning is not started.	

Output Parameters	Data Type	Description	Valid and Default Values
AtuneCV1LimitsFault	BOOL	CV1LimitsInv and CVManLimiting are True at start of CV1 Autotuning or during CV1 Autotuning. When True, CV1 Autotuning is not started or is aborted.	
AtuneCV2LimitsFault	BOOL	CV2LimitsInv and CVManLimiting are True at start of CV2 Autotuning or during CV2 Autotuning. When True, CV2 Autotuning is not started or is aborted.	
AtuneCV3LimitsFault	BOOL	CV3LimitsInv and CVManLimiting are True at start of CV3 Autotuning or during CV3 Autotuning. When True, CV3 Autotuning is not started or is aborted.	
AtuneCV1InitFault	BOOL	CV1Initializing is True at start of CV1 Autotuning or during CV1 Autotuning. When True, CV1 Autotuning is not started or is aborted.	
AtuneCV2InitFault	BOOL	CV2Initializing is True at start of CV2 Autotuning or during CV2 Autotuning. When True, CV2 Autotuning is not started or is aborted.	
AtuneCV3InitFault	BOOL	CV3Initializing is True at start of CV3 Autotuning or during CV3 Autotuning. When True, CV3 Autotuning is not started or is aborted.	
AtuneCV1EUSpanChanged	BOOL	CV1EUSpan or PVEUSpan changes during CV1 Autotuning. When True, CV1 Autotuning is aborted.	
AtuneCV2EUSpanChanged	BOOL	CV2EUSpan or PVEUSpan changes during CV2 Autotuning. When True, CV2 Autotuning is aborted.	
AtuneCV3EUSpanChanged	BOOL	CV3EUSpan or PVEUSpan changes during CV3 Autotuning. When True, CV3 Autotuning is aborted.	
AtuneCV1Changed	BOOL	CV1Oper is changed when in Operation control or CV1Prog is changed when in Program control or CV1 becomes high/low or ROC limited during CV1 Autotuning. When True, CV1 Autotuning is aborted.	

Output Parameters	Data Type	Description	Valid and Default Values
AtuneCV2Changed	BOOL	CV2Oper is changed when in Operation control or CV2Prog is changed when in Program control or CV2 becomes high/low or ROC limited during CV2 Autotuning. When True, CV2 Autotuning is aborted.	
AtuneCV3Changed	BOOL	CV3Oper is changed when in Operation control or CV3Prog is changed when in Program control or CV3 becomes high/low or ROC limited during CV3 Autotuning. When True, CV3 Autotuning is aborted.	
AtuneCV1Timeout	BOOL	Elapsed time is greater then AtuneTimeLimit since step test is started. When True, CV1 Autotuning is aborted.	
AtuneCV2Timeout	BOOL	Elapsed time is greater then AtuneTimeLimit since step test is started. When True, CV2 Autotuning is aborted.	
AtuneCV3Timeout	BOOL	Elapsed time is greater then AtuneTimeLimit since step test is started. When True, CV3 Autotuning is aborted.	
AtuneCV1PVNotSettled	BOOL	The PV is changed too much to Autotune for CV1. When True, CV1 Autotuning is aborted. Wait until PV is more stable before autotuning CV1.	
AtuneCV2PVNotSettled	BOOL	The PV is changed too much to Autotune for CV2. When True, CV2 Autotuning is aborted. Wait until PV is more stable before autotuning CV2.	
AtuneCV3PVNotSettled	BOOL	The PV is changed too much to Autotune for CV3. When True, CV3 Autotuning is aborted. Wait until PV is more stable before autotuning CV3.	
Status1	DINT	Bit mapped status of the function block.	
Status2	DINT	Additional bit mapped status for the function block.	
Status3CV1	DINT	Additional bit mapped CV1 status for the function block. A value of 0 indicates that no faults have occurred.	
Status3CV2	DINT	Additional bit mapped CV2 status for the function block. A value of 0 indicates that no faults have occurred.	

Output Parameters	Data Type	Description	Valid and Default Values
Status3CV3	DINT	Additional bit mapped CV3 status for the function block. A value of 0 indicates that no faults have occurred.	
InstructFault	BOOL	The function block has generated a fault. Indicates state of bits in Status1, Status2, and Status3CV(n), where (n) can be 1, 2, or 3. A value of 0 indicates that no faults have occurred. Any parameters that could be configured with an invalid value must have a status parameter to indicate their invalid status.	
PVFaulted	BOOL	Process variable PV health bad.	
PVSpanInv	BOOL	The span of PV inValid, PVEUMax < PVEUMin.	
SPProgInv	BOOL	SPProg < SPLLimit or > SPHLimit. Limit value used for SP.	
SPOperInv	BOOL	SPOper < SPLLimit or > SPHLimit. Limit value used for SP.	
SPLimitsInv	BOOL	Limits inValid: SPLLimit < PVEUMin, SPHLimit > PVEUMax, or SPHLimit < SPLLimit. If SPHLimit < SPLLimit, then limit value by using SPLLimit.	
SampleTimeTooSmall	BOOL	Model DeadTime / DeltaT must be less than or equal to 200.	
FactorInv	BOOL	Entered value for Factor < 0.	
TimingModeInv	BOOL	Entered TimingMode inValid. If the current mode is not Override or Hand then set to Manual mode.	
RTSMissed	BOOL	Only used when in Real Time Sampling mode. Is TRUE when ABS(DeltaT – RTSTime) > 1 millisecond.	
RTSTimeInv	BOOL	Entered RTSTime inValid.	
RTSTimeStampInv	BOOL	RTSTimeStamp inValid. If the current mode is not Override or Hand, then set to Manual mode.	
DeltaTInv	BOOL	DeltaT inValid. If the current mode is not Override or Hand then, set to Manual mode.	
CV1Faulted	BOOL	Control variable CV1 health bad.	
CV2Faulted	BOOL	Control variable CV2 health bad.	
CV3Faulted	BOOL	Control variable CV3 health bad.	
CV1HandFBFaulted	BOOL	CV1 HandFB value health bad.	
CV2HandFBFaulted	BOOL	CV2 HandFB value health bad.	
CV3HandFBFaulted	BOOL	CV3 HandFB value health bad.	

Output Parameters	Data Type	Description	Valid and Default Values
CV1ProgInv	BOOL	CV1Prog < 0 or > 100, or < CV1LLimit or > CV1HLimit when CVManLimiting is TRUE. Limit value used for CV1.	
CV2ProgInv	BOOL	CV2Prog < 0 or > 100, or < CV2LLimit or > CV2HLimit when CVManLimiting is TRUE. Limit value used for CV2.	
CV3ProgInv	BOOL	CV3Prog < 0 or > 100, or < CV3LLimit or > CV3HLimit when CVManLimiting is TRUE. Limit value used for CV3.	
CV1OperInv	BOOL	CV1Oper < 0 or > 100, or < CV1LLimit or > CV1HLimit when CVManLimiting is TRUE. Limit value used for CV1.	
CV2OperInv	BOOL	CV2Oper < 0 or > 100, or < CV2LLimit or> CV2HLimit when CVManLimiting is TRUE. Limit value used for CV2.	
CV3OperInv	BOOL	CV3Oper < 0 or > 100, or < CV3LLimit or> CV3HLimit when CVManLimiting is TRUE. Limit value used for CV3.	
CV1OverrideValueInv	BOOL	CV1OverrideValue < 0 or > 100. Limit value used for CV1.	
CV2OverrideValueInv	BOOL	CV2OverrideValue < 0 or > 100. Limit value used for CV2.	
CV3OverrideValueInv	BOOL	CV3OverrideValue < 0 or > 100. Limit value used for CV3.	
CV1TrackValueInv	BOOL	Entered CV1TrackValue < 0 or > 100. Limit value used for CV1.	
CV2TrackValueInv	BOOL	Entered CV2TrackValue < 0 or > 100. Limit value used for CV2.	
CV3TrackValueInv	BOOL	Entered CV3TrackValue < 0 or > 100. Limit value used for CV3.	
CV1EUSpanInv	BOOL	The span of CV1EU inValid, CV1EUMax equals CV1EUMin.	
CV2EUSpanInv	BOOL	The span of CV2EU inValid, CV2EUMax equals CV2EUMin.	
CV3EUSpanInv	BOOL	The span of CV3EU inValid, CV3EUMax equals CV3EUMin.	
CV1LimitsInv	BOOL	CV1LLimit < 0, CV1HLimit > 100, or CV1HLimit <= CV1LLimit. If CV1HLimit <= CV1LLimit, limit CV1 by using CV1LLimit.	
CV2LimitsInv	BOOL	CV2LLimit < 0, CV2HLimit > 100, or CV2HLimit <= CV2LLimit. If CV2HLimit <= CV2LLimit, limit CV2 by using CV2LLimit.	

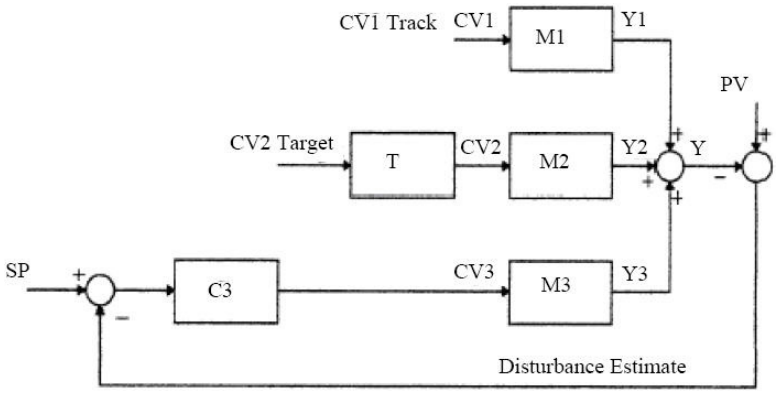
Output Parameters	Data Type	Description	Valid and Default Values
CV3LimitsInv	BOOL	CV3LLimit < 0, CV3HLimit > 100, or CV3HLimit <= CV3LLimit. If CV3HLimit <= CV3LLimit, limit CV3 by using CV3LLimit.	
CV1ROCLimitInv	BOOL	CV1ROCLimit < 0, disables CV1 ROC limiting.	
CV2ROCLimitInv	BOOL	CV2ROCLimit < 0, disables CV2 ROC limiting.	
CV3ROCLimitInv	BOOL	CV3ROCLimit < 0, disables CV3 ROC limiting.	
CV1HandFBInv	BOOL	CV1HandFB < 0 or > 100. Limit value used for CV1.	
CV2HandFBInv	BOOL	CV2HandFB < 0 or > 100. Limit value used for CV2.	
CV3HandFBInv	BOOL	CV3HandFB < 0 or > 100. Limit value used for CV3.	
CV1ModelGainInv	BOOL	CV1ModelGain is 1.#QNAN or -1.#IND (Not A Number), or ± 1.$ (Infinity ∞).	
CV2ModelGainInv	BOOL	CV2ModelGain is 1.#QNAN or -1.#IND (Not A Number), or ± 1.$ (Infinity ∞).	
CV3ModelGainInv	BOOL	CV3ModelGain is 1.#QNAN or -1.#IND (Not A Number), or ± 1.$ (Infinity ∞).	
CV1ModelTCInv	BOOL	CV1ModelTC < 0.	
CV2ModelTCInv	BOOL	CV2ModelTC < 0.	
CV3ModelTCInv	BOOL	CV3ModelTC < 0.	
CV1ModelDTInv	BOOL	CV1ModelDT < 0.	
CV2ModelDTInv	BOOL	CV2ModelDT < 0.	
CV3ModelDTInv	BOOL	CV3ModelDT < 0.	
CV1RespTCInv	BOOL	CV1RespTC < 0.	
CV2RespTCInv	BOOL	CV2RespTC < 0.	
CV3RespTCInv	BOOL	CV3RespTC < 0.	
CV1TargetInv	BOOL	CV1Target < 0. or > 100.	
CV2TargetInv	BOOL	CV2Target < 0. or > 100.	
CV3TargetInv	BOOL	CV3Target < 0. or > 100.	

Description

Coordinated Control is a flexible model-based algorithm that can be used in various configurations, for example:

- Three control variables are used to control one process variable
- Heat-cool split range control
- Feedforward control
- Zone temperature control
- Constraint control

The following illustration is an example of the Coordinated Control closed loop configuration.



In this example, CV1 is in Manual mode, CV2 is driven to its target value, and CV3 is the active control. The following table describes this example in detail.

Name	Description
CV1	Is in Manual mode
CV2	Is driven to its target value (CV2 = Target1stCV)
CV3	Is the active control (CV3 = Act1stCV)

This example could be a heat cooled system with a feed forward where:

- CV1 is feed forward;
- CV2 is cooling;
- CV3 heating.

Since CV1 is in Manual mode, CV3 target value as the lowest priority goal cannot be accomplished. PV will be maintained at the setpoint by using CV3, and at the same time CV2 will be driven to its target value (2nd priority goal).

If the operator changes the CV1 manual value, the control variable will take the change into account when calculating new CV3 and CV2.

M1	CV1 - PV First order lag with deadtime model
M2	CV2 - PV First order lag with deadtime model
M3	CV3 - PV First order lag with deadtime model
T	Target Response
C3	Model based algorithm to control PV by using CV3
Y1, Y2, Y3	Model outputs of M1, M2, M3
Y	PV prediction

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

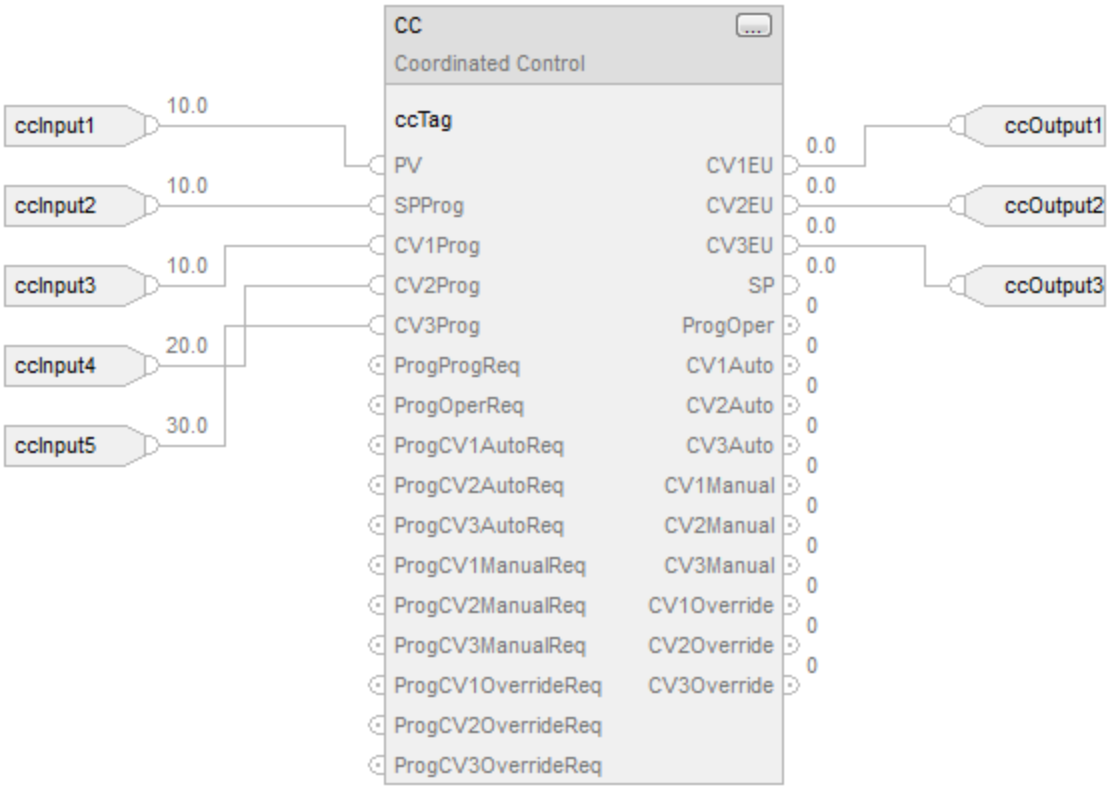
Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A.
Instruction first scan	N/A
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

Function Block



Structured Text

ccTag.PV := ccInput1;

ccTag.SPProg := ccInput2;

ccTag.CV1Prog := ccInput3;

ccTag.CV2Prog := ccInput4;

ccTag.CV3Prog := ccInput5;

CC(ccTag);

ccOutput1 := ccTag.CV1EU;

ccOutput2 := ccTag.CV2EU;

ccOutput3 := ccTag.CV3EU;

See also

Common Attributes on page 1083

CC Function Block Configuration on page 180

Select the Control Variable on page 254

Switch between Program control and Operator control on page 250

Structured Text Syntax on page 1057

Function Block Attributes on page 1043

CC Function Block Configuration

Starting with the default configuration, configure the following parameters:

Parameter	Description
PVEUMax	Maximum scaled value for PV.
PVEUMin	Minimum scaled value for PV.
SPHLimit	SP high limit value, scaled in PV units.
PPLLimit	SP low limit value, scaled in PV units.
CV1InitValue	An initial value of the control variable CV1 output.
CV2InitValue	An initial value of the control variable CV2 output.
CV3InitValue	An initial value of the control variable CV3 output.

If you have the process models available, you can intuitively tune the CC control variable by entering the following parameters:

Parameter	Description
ModelGains	Nonzero numbers (negative for direct acting control variable, positive for reverse acting control variable)
ModelTimeConstants	Always positive numbers
ModelDeadtimes	Always positive numbers
ResponseTimeConstants	Always positive numbers
Active 1st, 2nd and 3rd CV	Specify the order in which CV's will be used to compensate for PV - SP error.
Target 1st, 2nd and 3rd CV	Specify the priorities in which CV's will be driven to their respective target values.
CVTargetValues	Specify to which values should the control variable drive the individual CV's
TargetRespTC	Specify the speed of CV's to approach the target values

The function block behaves in a defined way for any combination of CV Active and Target lists and CV Auto-Manual modes. The function block attempts to accomplish these goals in the following order of priorities:

1. Control PV to SP
2. Control Target1stCV to its target value
3. Control Target2ndCV to its target value

If any CV is put in Manual mode, the CC function block gives up the goal with priority 3. If two CV's are in Manual mode, the CC function block is reduced to an IMC, (single input, single output) control variable controlling the PV to its setpoint.

In addition to this, however, the control variable reads the Manual CV values from the CV's that are in Manual mode as feedforward signals. Then, the CC

function block predicts the influence of the Manual CV values on the PV by using the appropriate internal models, and calculates the third CV that remains in Auto mode.

For integrating process types (such as level control and position control), internal nonintegrating models are used to approximate the integrating process. The Factor parameter is used to convert the identified integrating process models to nonintegrating internal models used for CV calculation. This is necessary to provide for stable function block execution.

CC Function Block Model Initialization

A model initialization occurs:

- During First Scan of the block
- When the ModelInit request parameter is set
- When DeltaT changes

You may need to manually adjust the internal model parameters or the response time constants. You can do so by changing the appropriate parameters and setting the appropriate ModelInit bit. The internal states of the control variable will be initialized, and the bit will automatically reset.

For example, modify the Model Gain for CV2 - PV model. Set the ModelInit2 parameter to TRUE to initialize the CV2 - PV internal model parameters and for the new model gain to take effect.

CC Function Block Tuning

The function block is equipped with an internal tuner (modeler). The purpose of the tuner is to identify the process model parameters and to use these parameters as internal model parameters (gain, time constant, and deadtime). The tuner also calculates an optimal response time constant. Set the tuner by configuring the following parameters for each CV - PV process.

ProcessType	Integral (level, position control) or nonintegrating (flow, pressure control)
ProcessGainSign	Set to indicate a negative process gain (increase in output causes a decrease in PV); reset to indicate a positive process gain (increase in output causes an increase in PV).
ResponseSpeed	Slow, medium, or fast, based on control objective.
NoiseLevel	An estimate of noise level on PV-low, medium, or high-such that the tuner can distinguish which PV change is a random noise and which is caused by the CV step change.
StepSize	A nonzero positive or negative number defining the magnitude of CV step change in either positive or negative direction, respectively.
PVTuneLimit	(only for integrating process type) in PV engineering units, defines how much of PV change that is caused by CV change to tolerate before aborting the tuning test due to exceeding this limit.

The tuner is started by setting the appropriate AtuneStart bit (AtuneCV1Start, for example). You can stop the tuning by setting the appropriate AtuneAbort bit.

After the tuning is completed successfully, the GainTuned, TCTuned, DTTuned, and RespTCTuned parameters are updated with the tuning results, and the AtuneStatus code is set to indicate complete.

You can copy these parameters to the ModelGain, ModelTC, and ResponseTC, respectively, by setting the AtuneUseModel bit. The control variable will

automatically initialize the internal variables and continue normal operation. It will automatically reset the AtuneUseModel bit.

See also

CC Function Block Tuning Procedure on page 182

CC Function Block Tuning Errors

If an error occurs during the tuning procedure, the tuning is aborted, and an appropriate AtuneStatus value is set. Also, a user can abort the tuning by setting the AtuneAbort parameter.

After an abort, the CV will assume its value before the step change, and the GainTuned, TCTuned, DTTuned, and RespTCTuned parameters are not updated. The AtuneStatus parameter identifies the reason for the abort.

CC Function Block Tuning Procedure

Follow these steps to configure the tuner.

1. Put all three CV parameters into Manual mode.
2. Set the AtuneStart parameter.

   The tuner starts collecting PV and CV data for noise calculation.
3. After collecting 60 samples (60*DeltaT) period, the tuner adds StepSize to the CV.

   After successfully collecting the PV data as a result of the CV step change, the CV assumes its value before the step change and the AtuneStatus, GainTuned, TCTuned, DTTuned, and RespTCTuned parameters are updated.
4. Set the AtuneUseModel parameter to copy the tuned parameters to the model parameters

   The function block then resets the AtuneUseModel parameter.
   After a successful AutoTuneDone, the Atune parameter is set to one (1).
   Tuning completed successfully.

To identify models and to calculate response time constants for all three CV-PV processes, run the tuner up to three times to obtain CV1-PV, CV2-PV, and CV3-PV models and tuning, respectively.

Internal Model Control (IMC)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The IMC instruction controls a single process variable by manipulating a single control-variable output. This function block performs an algorithm where the actual error signal is compared against that of an internal first-order lag plus deadtime model of the process. The IMC function block

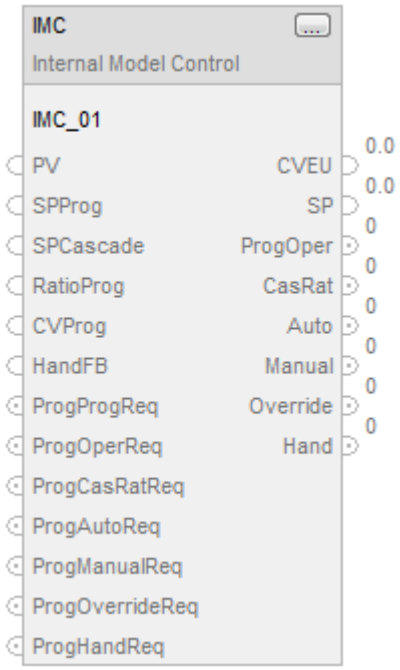
calculates the control variable output (CV) in the Auto mode based on the PV - SP deviation, internal model, and tuning.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram.

Function Block



Structured Text

IMC(IMC_tag);

Operands

Function Block

Operands:	Type:	Format	Description:
IMC tag	INTERNAL MODEL CONTROL	Structure	IMC Structure

Structured Text

Operands:	Type:	Format	Description:
IMC tag	INTERNAL MODEL CONTROL	Structure	IMC Structure

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

IMPORTANT Whenever an APC block detects a change in Delta Time (DeltaT), a ModelInit will be performed. For this reason the blocks should only be run in one of the TimingModes in which DeltaT will be constant.

- TimingMode = 0 (Periodic) while executing these function blocks in a Periodic Task
- TimingMode = 1 (Oversample)

In either case, if the Periodic Task time is dynamically changed, or the OversampleDT is dynamically changed, the block will perform a ModelInit.

The following TimingMode setting are not recommended due to jitter in DeltaT:

- TimingMode = 0 (Periodic) while executing these function blocks in a Continuous or Event Task
- TimingMode = 2 (RealTimeSample)

Structure

Input Parameters	Data Type	Description	Valid and Default Values
EnableIn	BOOL	Enable Input. If False, the function block will not execute and outputs are not updated.	Default=TRUE
PV	REAL	Scaled process variable input. This value is typically read from an analog input module.	Valid = any float Default = 0.0
PVFault	BOOL	PV bad health indicator. If PV is read from an analog input, then PVFault will normally be controlled by the analog input fault status. If PVFault is TRUE, it indicates an error on the input module, set bit in Status.	Default = FALSE FALSE = Good Health
PVUEMax	REAL	Maximum scaled value for PV. The value of PV and SP that corresponds to 100% span of the Process Variable. If PVEUMax ≤ PVEUMin, set bit in Status.	Valid = PVEUMin < PVEUMax ≤ maximum positive float Default = 100.0
PVUEMin	REAL	Minimum scaled value for PV. The value of PV and SP that corresponds to 0% span of the Process Variable. If PVEUMax ≤ PVEUMin, set bit in Status.	Valid = maximum negative float ≤ PVEUMin < PVEUMax Default = 0.0
SPProg	REAL	SP Program value, scaled in PV units. SP is set to this value when in Program control. If value of SPProg or SPOper < SPLLimit or > SPHLimit, set bit in Status and limit value used for SP.	Valid = SPLLimit to SPHLimit Default = 0.0

Input Parameters	Data Type	Description	Valid and Default Values
SPOper	REAL	SP Operator value, scaled in PV units. SP set to this value when in Operator control. If value of SPProg or SPOper < SPLLimit or > SPHLimit, set bit in Status and limit value used for SP.	Valid = SPLLimit to SPHLimit Default = 0.0
SPCascade	REAL	SP Cascade value, scaled in PV units. If CascadeRatio mode and UseRatio is FALSE, then SP is set to this value, typically this will be CVEU of a primary loop. If CascadeRatio mode and UseRatio is TRUE, then SP is set to this value times Ratio. If value of SPCascade < SPLLimit or > SPHLimit, set bit in Status and limit value used for SP.	Valid = SPLLimit to SPHLimit Default = 0.0
SPHLimit	REAL	SP high limit value, scaled in PV units. If SPHLimit < SPLLimit or SPHLimit > PVEUMax, set bit in Status.	Valid = SPLLimit to PVEUMax Default = 100.0
SPLLimit	REAL	SP low limit value, scaled in PV units. If SPLLimit < PVEUMin, or SPHLimit < SPLLimit, set bit in Status and limit SP by using the value of SPLLimit.	Valid = PVEUMin to SPHLimit Default = 0.0
UseRatio	BOOL	Allow Ratio control permissive. Set TRUE to enable ratio control when in CascadeRatio mode.	Default = FALSE
RatioProg	REAL	Ratio Program multiplier, no units (for example, scalar). Ratio and RatioOper are set to this value when in Program control. If RatioProg or RatioOper < RatioLLimit or > RatioHLimit, set bit in Status and limit value used for Ratio.	Valid = RatioLLimit to RatioHLimit Default = 1.0
RatioOper	REAL	Ratio Operator multiplier, no units (for example, scalar). Ratio is set to this value when in Operator control. If RatioProg or RatioOper < RatioLLimit or > RatioHLimit, set bit in Status and limit value used for Ratio.	Valid = RatioLLimit to RatioHLimit Default = 1.0
RatioHLimit	REAL	Ratio high limit value, no units (for example, scalar). Limits the value of Ratio obtained from RatioProg or RatioOper. If RatioLLimit < 0, set bit in Status and limit to zero. If RatioHLimit < RatioLLimit, set bit in Status and limit Ratio by using the value of RatioLLimit.	Valid = RatioLLimit to maximum positive float Default = 1.0

Input Parameters	Data Type	Description	Valid and Default Values
RatioLLimit	REAL	Ratio low limit value, no units (for example, scalar). Limits the value of Ratio obtained from RatioProg or RatioOper. If RatioLLimit < 0, set bit in Status and limit to zero. If RatioHLimit < RatioLLimit, set bit in Status and limit Ratio by using the value of RatioLLimit.	Valid = 0.0 to RatioHLimit Default = 1.0
CVFault	BOOL	Control variable bad health indicator. If CVEU controls an analog output, then CVFault will normally come from the analog output's fault status. If CVFault is TRUE, it indicates an error on the output module, set bit in Status.	Default = FALSE FALSE = Good Health
CVInitReq	BOOL	CV initialization request. While TRUE, set CVEU to the value of CVInitValue. This signal will normally be controlled by the In Hold status on the analog output module controlled by CVEU or from the InitPrimary output of a secondary IMC loop.	Default = FALSE
CVInitValue	REAL	CVEU initialization value, scaled in CVEU units. When CVInitializing is TRUE set CVEU equal to CVInitValue and CV to the corresponding percentage value. CVInitValue will normally come from the feedback of the analog output controlled by CVEU or from the setpoint of a secondary loop. The function block initialization is disabled when CVFaulted or CVEUSpanInv are TRUE (bad).	Valid = any float Default = 0.0
CVProg	REAL	CV Program-Manual value. CV is set to this value when in Program control and Manual mode. If value of CVProg or CVOper < 0 or > 100, or < CVLLimit or > CVHLimit when CVManLimiting is TRUE, set unique Status bit and limit value used for CV.	Valid = 0.0 to 100.0 Default = 0.0
CVOper	REAL	CV Operator-Manual value. CV is set to this value when in Operator control and Manual mode. If not Operator-Manual mode, set CVOper to the value of CV at the end of each function block execution. If value of CVProg or CVOper < 0 or > 100, or < CVLLimit or > CVHLimit when CVManLimiting is TRUE, set unique Status bit and limit value used for CV.	Valid = 0.0 to 100.0 Default = 0.0

Input Parameters	Data Type	Description	Valid and Default Values
CVOverrideValue	REAL	CV Override value. CV set to this value when in Override mode. This value should correspond to a safe state output of the IMC loop. If value of CVOverrideValue < 0 or >100, set unique Status bit and limit value used for CV.	Valid = 0.0 to 100.0 Default = 0.0
CVTrackValue	REAL	CV track value. When CVTrackReq is enabled and the IMC function block is in Manual, the CVTrackValue will be ignored, and the IMC internal model will update its historical data with the CVOper or CVProg value. When CVTrackReq is enabled and the IMC function block is in Auto, the internal model will update its historical data based on the value of CVTrackValue. The CV in this case will be allowed to move as if the IMC function block was still controlling the process. This is useful in multiloop selection schemes where you want the IMC function block to follow the output of a different controlling algorithm, where you would connect the output of the controlling algorithm into the CVTrackValue.	Valid = 0.0 to 100.0 Default = 0.0
CVManLimiting	BOOL	Limit CV in Manual mode request. If Manual mode and CVManLimiting is TRUE, CV will be limited by the CVHLimit and CVLLimit values.	Default = FALSE
CVEUMax	REAL	Maximum value for CVEU. The value of CVEU that corresponds to 100% CV. If CVEUMax = CVEUMin, set bit in Status.	Valid = any float Default = 100.0
CVEUMin	REAL	Minimum value of CVEU. The value of CVEU that corresponds to 0% CV. If CVEUMax = CVEUMin, set bit in Status.	Valid = any float Default = 0.0
CVHLimit	REAL	CV high limit value. This is used to set the CVHAlarm output. It is also used for limiting CV when in Auto or CascadeRatio modes or Manual mode if CVManLimiting is TRUE. If CVLLimit < 0, if CVHLimit > 100, if CVHLimit < CVLLimit, set bit in Status. If CVHLimit < CVLLimit, limit CV by using the value of CVLLimit.	Valid = CVLLimit < CVHLimit ≤ 100.0 Default = 100.0

Input Parameters	Data Type	Description	Valid and Default Values
CVLLimit	REAL	CV low limit value. This is used to set the CVLAlarm output. It is also used for limiting CV when in Auto or CascadeRatio modes or Manual mode if CVManLimiting is TRUE. If CVLLimit < 0, if CVHLimit > 100, if CVHLimit < CVLLimit, set bit in Status. If CVHLimit < CVLLimit, limit CV by using the value of CVLLimit.	Valid = 0.0 ≤ CVLLimit < CVHLimit Default = 0.0
CVROCPosLimit	REAL	CV increasing rate of change limit, in percent per second. Rate of change limiting is only used when in Auto or CascadeRatio modes or Manual mode if CVManLimiting is TRUE. A value of zero disables CV ROC limiting. If value of CVROCPOSLimit < 0, set bit in Status and disable CV ROC limiting.	Valid = 0.0 to maximum positive float Default = 0.0
CVROCNegLimit	REAL	CV decreasing rate of change limit, in percent per second. Rate of change limiting is only used when in Auto or CascadeRatio modes or Manual mode if CVManLimiting is TRUE. A value of zero disables CV ROC limiting. If value of CVROCNegLimit < 0, set bit in Status and disable CV ROC limiting.	Valid = 0.0 to maximum positive float Default = 0.0
HandFB	REAL	CV HandFeedback value. CV set to this value when in Hand mode and HandFBFault is FALSE (good health). This value would typically come from the output of a field mounted hand/auto station and would be used to generate a bumpless transfer out of Hand mode. If value of HandFB < 0 or > 100, set unique Status bit and limit value used for CV.	Valid = 0.0...100.0 Default = 0.0
HandFBFault	BOOL	HandFB value bad health indicator. If the HandFB value is read from an analog input, then HandFBFault will normally be controlled by the status of the analog input channel. If HandFBFault is TRUE, it indicates an error on the input module, set bit in Status.	Default = FALSE FALSE = Good Health
WindupHIn	BOOL	Windup high request. When TRUE, CV will not be allowed to increase in value. This signal will typically be the WindupHOut output from a secondary loop.	Default = FALSE

Input Parameters	Data Type	Description	Valid and Default Values
WindupIIn	BOOL	Windup low request. When TRUE, CV will not be allowed to decrease in value. This signal will typically be the WindupLOut output from a secondary loop.	Default = FALSE
GainEUSpan	BOOL	ModelGain units in EU or as % of span. CV ModelGain units in EU or % of span. Set to interpret ModelGain as EU, reset to interpret ModelGain as % of Span.	Default = FALSE TRUE = Gain in EU FALSE = Gain in % of span
ProcessGainSign	BOOL	Used only for Autotuning. Sign of the process gain (Delta PV/Delta CV). Set to indicate a negative process gain (increase in output causes a decrease in PV). Reset to indicate a positive process gain (increase in output causes an increase in PV).	Default = FALSE
ProcessType	DINT	Process type selection (1=Integrating, 0=non-integrating)	Default = 0
ModelGain	REAL	The internal model gain parameter. Enter a positive or negative gain depending on process direction.	Valid = maximum negative float –> maximum positive float Default = 0.0
ModelTC	REAL	The internal model time constant in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
ModelDT	REAL	The internal model deadtime in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
RespTC	REAL	The tuning parameter that determines the speed of the control variable action in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
PVTracking	BOOL	SP track PV request. Set TRUE to enable SP to track PV. Ignored when in CascadeRatio or Auto modes.	Default = FALSE
CVTrackReq	BOOL	CV Track request. Set true to enable CV Tracking when autotune is OFF. Ignored in Hand and Override mode.	Default = FALSE
AllowCasRat	BOOL	Allow CascadeRatio mode permissive. Set TRUE to allow CascadeRatio mode to be selected by using either ProgCasRatReq or OperCasRatReq.	Default = FALSE
ManualAfterInit	BOOL	Manual mode after initialization request. When TRUE, the function block will be placed in the Manual mode when CVInitializing is set TRUE unless the current mode is Override or Hand. When ManualAfterInit is FALSE, the function block's mode will not be changed.	Default = FALSE

Input Parameters	Data Type	Description	Valid and Default Values
ProgProgReq	BOOL	Program Program Request. Set TRUE by the user program to request Program control. Ignored if ProgOperReq is TRUE. Holding this TRUE and ProgOperReq FALSE can be used to lock the function block into program control. When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
ProgOperReq	BOOL	Program Operator Request. Set TRUE by the user program to request Operator control. Holding this TRUE can be used to lock the function block into operator control. When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
ProgCasRatReq	BOOL	Program-Cascade/Ratio mode request. Set TRUE by the user program to request Cascasde/Ratio mode. When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
ProgAutoReq	BOOL	Program-Auto mode request. Set TRUE by the user program to request Auto mode. When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
ProgManualReq	BOOL	Program-Manual mode request. Set TRUE by the user program to request Manual mode. When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
ProgOverrideReq	BOOL	Program-Override mode request. Set TRUE by the user program to request Override mode. When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
ProgHandReq	BOOL	Program-Hand mode request. Set TRUE by the user program to request Hand mode. This value will usually be read as a digital input from a hand/auto station. When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
OperProgReq	BOOL	Operator Program Request. Set TRUE by the operator interface to request Program control. The function block resets this parameter to FALSE.	Default = FALSE
OperOperReq		Operator Operator Request. Set TRUE by the operator interface to request Operator control. The function block will reset this parameter to FALSE.	Default = FALSE

Input Parameters	Data Type	Description	Valid and Default Values
OperCasRatReq	BOOL	Operator-CascadeRatio mode request. Set TRUE by the operator interface to request CascadeRatio mode. The function block will reset this parameter to FALSE.	Default = FALSE
OperAutoReq	BOOL	Operator-Auto mode request. Set TRUE by the operator interface to request Auto mode. The function block will reset this parameter to FALSE.	Default = FALSE
OperManualReq	BOOL	Operator-Manual mode request. Set TRUE by the operator interface to request Manual mode. The function block will reset this parameter to FALSE.	Default = FALSE
ProgValueReset	BOOL	Reset Program control values. When TRUE, the Prog_xxx_Req inputs are reset to FALSE. When TRUE and Program control, set SPProg equal to SP and CVProg equal to CV. When ProgValueReset is TRUE, the function block resets this parameter to FALSE.	Default = FALSE
TimingMode	DINT	Selects Time Base Execution mode. Value/Description 0 = Periodic mode 1 = Oversample mode 2 = Real time sampling mode For more information about timing modes, see Function Block Attributes.	Valid = 0...2 Default = 0
OverSampleDT	REAL	Execution time for Oversample mode.	Valid = 0 to max. TON_Timer elapsed time (4194.303 seconds) Default = 0
RTSTime	DINT	Module update period for Real Time Sampling mode.	Valid = 1 to 32,767 1 count = 1 ms
RTSTimeStamp	DINT	Module time stamp value for Real Time Sampling mode.	Valid = 0 to 32,767 (wraps from 32,767 to 0) 1 count = 1 ms
PVTuneLimit	REAL	PV tuning limit scaled in the PV units. When Autotune is running and predicted PV exceeds this limit, the tuning will be aborted.	Range: any float Default=0
AtuneTimeLimit	REAL	Maximum time for autotune to complete following the CV step change. When autotune exceeds this time, tuning will be aborted.	Valid range: any float > 0. Default = 60 minutes
NoiseLevel	DINT	An estimate of the noise level expected on the PV to compensate for it during tuning. The selections are: 0=low, 1=medium, 2=high	Range: 0 to 2 Default=1

Input Parameters	Data Type	Description	Valid and Default Values
CVStepSize	REAL	CV step size in percent for the tuning step test. Step size is directly added to CV subject to high/low limiting.	Range: -100% ... 100% Default=10%
ResponseSpeed	DINT	Desired speed of closed loop response. Slow response: ResponseSpeed=0 Medium response: ResponseSpeed=1 Fast response: ResponseSpeed=2. If ResponseSpeed is less than 0, Slow response is used. If ResponseSpeed is greater than 2, Fast response is used.	Range: 0 to 2 Default=1
ModelInit	BOOL	Internal model initialization switch.	Default = FALSE
Factor	REAL	Non-integrating model approximation factor. Only used for integrating process types.	Default = 100
AtuneStart	BOOL	Start Autotune request. Set True to initiate auto tuning of the function block. Ignored when IMC is not in Manual mode. The function block will reset this parameter to FALSE.	Default = FALSE
AtuneUseModel	BOOL	Use Autotune model request. Set True to replace the current model parameters with the calculated Autotune model parameters. The function block sets the input parameter to FALSE.	Default = FALSE
AtuneAbort	BOOL	Abort Autotune request. Set True to abort the auto tuning of the IMC function block. The function block sets input parameter to FALSE.	Default = FALSE

Output Parameters	Data Type	Description	Valid and Default Values
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if CVEU overflows.	
CVEU	REAL	Scaled control variable output. Scaled by using CVEUMax and CVEUMin, where CVEUMax corresponds to 100% and CVEUMin corresponds to 0%. This output is typically used to control an analog output module or a secondary loop. CVEU = (CV * CVEUSpan / 100) + CVEUMin CVEU span calculation: CVEUSpan = (CVEUMax – CVEUMin)	
CV	REAL	Control variable output. This value will always be expressed as 0...100%. CV is limited by CVHLimit and CVLLimit when in Auto or CascadeRatio mode or Manual mode if CVManLimiting is TRUE; otherwise limited by 0 and 100%.	

Output Parameters	Data Type	Description	Valid and Default Values
DeltaCV	REAL	Difference between the Current CV and the previous CV (Current CV - previous CV).	
CVInitializing	BOOL	Initialization mode indicator. Set TRUE when CVInitReq or function block FirstScan are TRUE, or on a TRUE to FALSE transition of CVFault (bad to good). CVInitializing is set FALSE after the function block has been initialized and CVInitReq is no longer TRUE.	
CVHAlarm	BOOL	CV high alarm indicator. TRUE when the calculated value for CV > 100 or CVHLimit.	
CVLAlarm	BOOL	CV low alarm indicator. TRUE when the calculated value for CV < 0 or CVLLimit.	
CVROCPosAlarm	BOOL	CV rate of change alarm indicator. TRUE when the calculated rate of change for CV exceeds CVROCPosLimit.	
CVROCNegAlarm	REAL	CV rate of change alarm indicator. TRUE when the calculated rate of change for CV exceeds CVROCNegLimit.	
SP	REAL	Current setpoint value. The value of SP is used to control CV when in the Auto, the CascadeRatio, or the PV Tracking mode, scaled in PV units.	
SPPercent	REAL	The value of SP expressed in percent of span of PV. SPPercent = ((SP – PVEUMin) * 100) / PVSpan where PVSpan = PVEUMax – PVEUMin	
SPHAlarm	BOOL	SP high alarm indicator. TRUE when the SP ≥ SPHLimit.	
SPLAlarm	BOOL	SP low alarm indicator. TRUE when the SP ≤ SPLLimit.	
PVPercent	REAL	PV expressed in percent of span. PVPercent = ((PV – PVEUMin) * 100) / PVSpan PV Span calculation: PVSpan = (PVEUMax – PVEUMin)	
E	REAL	Process error. Difference between SP and PV, scaled in PV units.	
EPercent	REAL	The error expressed as a percent of span.	
InitPrimary	BOOL	Initialize primary loop command. TRUE when not in CasRat mode or when CVInitializing is TRUE. This signal normally would be used by the CVInitReq input of a primary loop.	

Output Parameters	Data Type	Description	Valid and Default Values
WindupHOut	BOOL	Windup high indicator. TRUE when either a SP high or CV high/low limit has been reached. This signal will typically be used by the WindupHIn input to limit the windup of the CV output on a primary loop.	
WindupLOut	BOOL	Windup low indicator. TRUE when either a SP or CV high/low limit has been reached. This signal will typically be used by the WindupLIn input to limit the windup of the CV output on a primary loop.	
Ratio	REAL	Current ratio multiplier, no units.	
RatioHAlarm	BOOL	Ratio high alarm indicator. TRUE when Ratio > RatioHLimit.	
RatioLAlarm	BOOL	Ratio low alarm indicator. TRUE when Ratio < RatioLLimit.	
ProgOper	BOOL	Program/Operator control indicator. TRUE when in Program control. FALSE when in Operator control.	
CasRat	BOOL	CascadeRatio mode indicator. TRUE when in the CascadeRatio mode.	
Auto	BOOL	Auto mode indicator. TRUE when in the Auto mode.	
Manual	BOOL	Manual mode indicator. TRUE when in the Manual mode.	
Override	BOOL	Override mode indicator. TRUE when in the Override mode.	
Hand	BOOL	Hand mode indicator. TRUE when in the Hand mode.	
DeltaT	REAL	Elapsed time between updates in seconds.	
StepSizeUsed	REAL	Actual CV step size used for tuning.	
GainTuned	REAL	The calculated value of the internal model gain after tuning is completed.	
TCTuned	REAL	The calculated value of the internal model time constant after tuning is completed.	
DTTuned	REAL	The calculated value of the internal model deadtime after tuning is completed.	
RespTCTunedS	REAL	The calculated value of the control variable time constant in slow response speed after tuning is completed.	
RespTCTunedM	REAL	The calculated value of the control variable time constant in medium response speed after tuning is completed.	

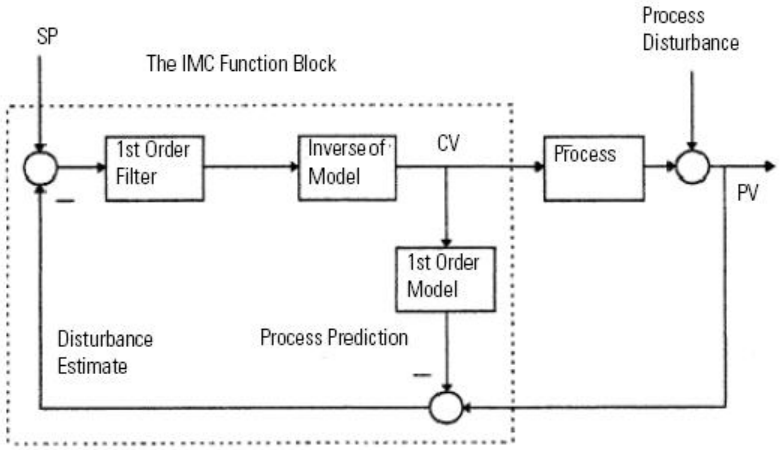
Output Parameters	Data Type	Description	Valid and Default Values
RespTCTunedF	REAL	The calculated value of the control variable time constant in fast response speed after tuning is completed.	
AtuneOn	BOOL	Set True when auto tuning has been initiated.	
AtuneDone	BOOL	Set True when auto tuning has completed successfully.	
AtuneAborted	BOOL	Set True when auto tuning has been aborted by user or due to errors that occurred during the auto tuning operation.	
AtuneStatus	DINT	Indicates the block tuning status.	
AtuneFault	BOOL	Autotune has generated any of the following faults.	Bit 0 of AtuneStatus
AtunePVOutOfLimit	BOOL	Either PV or the deadtime-step ahead prediction of PV exceeds PVTuneLimit during Autotuning. When True, Autotuning is aborted.	Bit 1 of AtuneStatus
AtuneModeInv	BOOL	The IMC mode was not Manual at start of Autotuning or the IMC mode was changed from Manual during Autotuning. When True, Autotuning is not started or is aborted.	Bit 2 of AtuneStatus
AtuneCVWindupFault	BOOL	WindupHIn or WindupLIn is True at start of Autotuning or during Autotuning. When True, Autotuning is not started or is aborted.	Bit 3 of AtuneStatus
AtuneStepSize0	BOOL	StepSizeUsed = 0 at start of Autotuning. When True, Autotuning is not started.	Bit 4 of AtuneStatus
AtuneCVLimitsFault	BOOL	CVLimitsInv and CVManLimiting are True at start of Autotuning or during Autotuning. When True, Autotuning is not started or is aborted.	Bit 5 of AtuneStatus
AtuneCVInitFault	BOOL	CVInitializing is True at start of Autotuning or during Autotuning. When True, Autotuning is not started or is aborted.	Bit 6 of AtuneStatus
AtuneEUSpanChanged	BOOL	CVEUSpan or PVEUSpan changes during Autotuning. When True, Autotuning is aborted.	Bit 7 of AtuneStatus
AtuneCVChanged	BOOL	CVOper is changed when in Operatior control or CVProg is changed when in Program control or CV becomes high/low or ROC limited during Autotuning. When True, Autotuning is aborted.	Bit 8 of AtuneStatus
AtuneTimeout	BOOL	Elapsed time is greater then AtuneTimeLimit since step test is started. When True, Autotuning is aborted.	Bit 9 of AtuneStatus

Output Parameters	Data Type	Description	Valid and Default Values
AtunePVNotSettled	BOOL	The PV is changed too much to Autotune. When True, Autotuning is aborted. Wait until PV is more stable before autotuning.	Bit 10 of AtuneStatus
Status1	DINT	Bit mapped status of the function block.	
Status2	DINT	Additional bit mapped status for the function block.	
InstructFault	BOOL	Function block has generated a fault. Indicates state of bits in Status1 and status2. A value of 0 indicates that no faults have occurred. Any parameters that could be configured with an invalid value must have a status parameter to indicate their invalid status.	Bit 0 of Status1
PVFaulted	BOOL	Process variable PV health bad.	Bit 1 of Status1
CVFaulted	BOOL	Control variable CV faulted	Bit 2 of Status1
HandFBFaulted	BOOL	HandFB value health bad	Bit 3 of Status1
PVSpanInv	BOOL	The span of PV invalid, PVEUMax < PVEUMin.	Bit 4 of Status1
SPProgInv	BOOL	SPProg < SPLLimit or > SPHLimit. Limit value used for SP.	Bit 5 of Status1
SPOperInv	BOOL	SPOper < SPLLimit or > SPHLimit. Limit value used for SP.	Bit 6 of Status1
SPCascadeInv	BOOL	SPCascade < SPLLimit or > SPHLimit. Limit value used for SP.	Bit 7 of Status1
SPLimitsInv	BOOL	Limits invalid: SPLLimit < PVEUMin, SPHLimit > PVEUMax, or SPHLimit < SPLLimit. If SPHLimit < SPLLimit, then limit value using SPLLimit.	Bit 8 of Status1
RatioLimitsInv	BOOL	Ratio high-low limits invalid, low limit < 0 or high limit < low limit.	Bit 9 of Status1
RatioProgInv	BOOL	RatioProg < RatioLLimit or > RatioHLimit. Limit value used for Ratio.	Bit 10 of Status1
RatioOperInv	BOOL	RatioOper < RatioLLimit or > RatioHLimit. Limit value used for Ratio.	Bit 11 of Status1
CVProgInv	BOOL	CVProg < 0 or > 100, or < CVLLimit or > CVHLimit when CVManLimiting is TRUE. Limit value used for CV.	Bit 12 of Status1
CVOperInv	BOOL	CVOper < 0 or > 100, or < CVLLimit or > CVHLimit when CVManLimiting is TRUE. Limit value used for CV.	Bit 13 of Status1
CVOverrideValueInv	BOOL	CVOverrideValue < 0 or > 100. Limit value used for CV.	Bit 14 of Status1
CVTrackValueInv	BOOL	CVTrackValue < 0 or > 100. Limit value used for CV.	Bit 15 of Status1
CVEUSpanInv	BOOL	The span of CVEU invalid, CVEUMax equals CVEUMin.	Bit 16 of Status1
CVLimitsInv	BOOL	CVLLimit < 0, CVHLimit > 100, or CVHLimit <= CVLLimit. If CVHLimit <= CVLLimit, limit CV by using CVLLimit.	Bit 17 of Status1

Output Parameters	Data Type	Description	Valid and Default Values
CVROCLimitInv	BOOL	CVROCLimit < 0, disables ROC limiting.	Bit 18 of Status1
HandFBInv	BOOL	HandFB < 0 or > 100. Limit value used for CV.	Bit 19 of Status1
SampleTimeTooSmall	BOOL	Model DeadTime / DeltaT must be less than or equal to 200.	Bit 20 of Status1
FactorInv	BOOL	Factor < 0.	Bit 21 of Status1
ModuleGainInv	BOOL	ModelGain is 1.#QNAN or -1.#IND (Not A Number), or ±1.$ (Infinity ∞)	Bit 22 of Status1
ModelTCInv	BOOL	ModelTC < 0.	Bit 23 of Status1
ModelDTInv	BOOL	ModelDT < 0.	Bit 24 of Status1
RespTCInv	BOOL	RespTC < 0.	Bit 25 of Status1
TimingModelInv	BOOL	TimingMode invalid. If the current mode is not Override or Hand then set to Manual mode.	Bit 27 of Status2
RTSMissed	BOOL	Only used when in Real Time Sampling mode. Is TRUE when ABS(DeltaT - RTSTime) > 1 millisecond.	Bit 28 of Status2.
RTSTimeInv	BOOL	RTSTime invalid.	Bit 29 of Status2.
RTSTimeStampInv	BOOL	RTSTimeStamp invalid. If the current mode is not Override or Hand then set to Manual mode.	Bit 30 of Status2.
DeltaTInv	BOOL	DeltaT invalid. If the current mode is not Override or Hand then set to Manual mode.	Bit 31 of Status2.

Description

The following illustration shows the IMC function block configuration.



At each execution, the IMC function block compares the actual PV measurement with PV prediction. The result is called disturbance estimate,

which is the effect of unmeasured process disturbances combined with the modeling imprecision. The disturbance estimate is used as a bias for the setpoint of the control variable. In the ideal case of no disturbances and perfect modeling, the disturbance estimate (the feedback signal) becomes equal to zero.

First Order Model	M = K/(T*s+1)*exp(-D*s)	
Inverse of Model		Inv = (T*s+1)/K
First Order Filter		F = 1/(e*s+1)

PV prediction = exp(-D*s)/(e*s+1) * (SP - Dist. estimate)

K...	Model gain
T...	Model time constant
D...	Model deadtime
e...	Response time constant
s...	Laplace variable

The function block then calculates the CV value (CVHLimit, CVLLimit, and rate of change limits are imposed) and the PV prediction.

The IMC function block can be used in place of a PID function block with the advantage over the PID control variable when controlling processes with large deadtimes.

For an integrating process type (such as level control and position control), an internal nonintegrating model is used to approximate the integrating process. The Factor parameter is used to convert the identified integrating-process model to a nonintegrating internal model that is used for CV calculation. This is necessary to provide for stable IMC execution.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

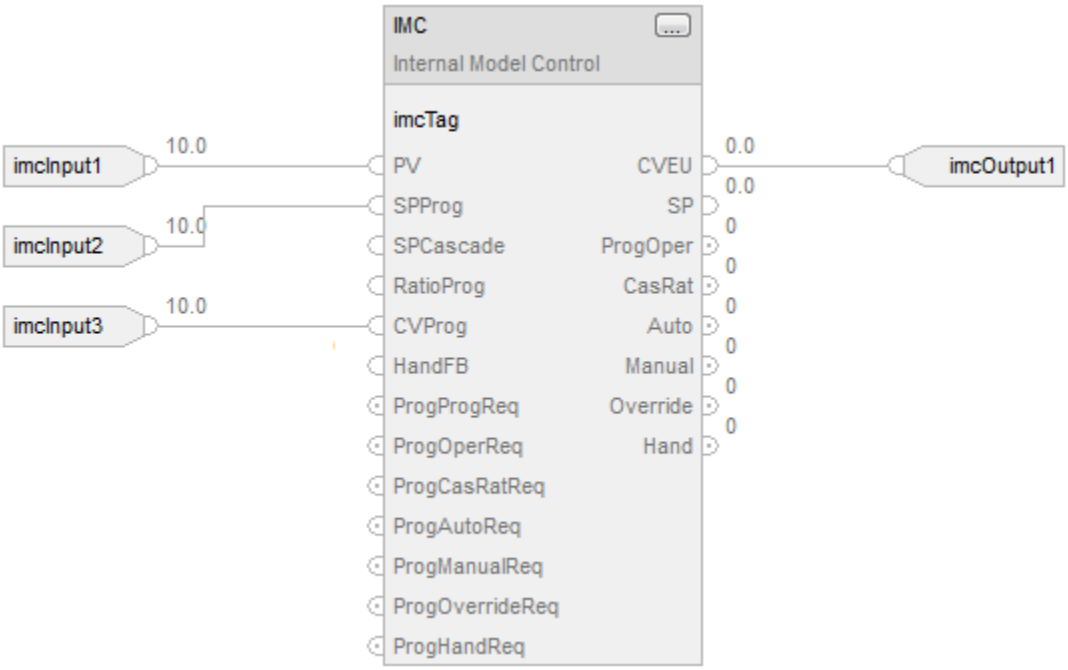
Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and .EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Examples

Function Block



Structured Text

```
imcTag.PV := imcInput1;
imcTag.SPProg := imcInput2;
imcTag.CVProg := imcInput3;
IMC(imcTag);
imcOutput1 := imcTag.CVEU;
```

See also

Processing Faults on page 254

IMC Function Block Tuning on page 201

Common Attributes on page 1083

Structured Text Syntax on page 1057

Function Block Attributes on page 1043

IMC Function Block Configuration

Follow these steps to create a basic IMC configuration.

1. Starting with the default configuration, configure the following parameters.

Parameter	Description
PVEUMax	Maximum scaled value for PV.
PVEUMin	Minimum scaled value for PV.
SPHLimit	SP high limit value, scaled in PV units.
SPLLimit	SP low limit value, scaled in PV units.
CVInitValue	An initial value of the control variable output.

2. If you have the process model available, you can intuitively tune the IMC control variable by entering the following four parameters.

Parameter	Description
Model Gain	A nonzero number (negative for direct acting control variable. positive for reverse acting control variable).
Model Time Constant	Always a positive number.
Model Deadtime	Always a positive number.
Response Time Constant	Always a positive number - used to tune the response of the IMC control variable. A smaller number gives a faster response.

   At this point, you have completed the basic configuration. You did not configure the built-in tuner. The control variable is ready to be put online in either Auto or Manual mode. For tuning, use the default settings. Refer to IMC Function Block Tuning.

3. If you do not know the process model, you need to identify the model and tune the control variable by using the built-in tuner (modeler) for the control variable to operate correctly in the Auto mode.

The control variable uses a first order lag with deadtime internal process model and a first order filter (total of four tuning parameters) to calculate the CV. The CV is calculated such that the process variable (PV) follows a first order lag trajectory when approaching the setpoint value.
Speed of response depends on the value of the response time constant. The smaller that the response time constant is, the faster the control variable response will be. The response time constant should be set such that the PV reaches the setpoint in a reasonable time based on the process dynamics. The larger that the response time constant is, the slower the control variable response will be, but the control variable also becomes more robust. Refer to IMC Function Block Tuning.
In the Manual mode, the CV is set equal to the operator-entered or program-generated CVOper or CVProg parameter.
For the Manual to Auto mode bumpless transfer and for safe operation of the control variable, the CV rate of change limiter is implemented such that the CV cannot change from its current state any faster than the rate of change limit parameter specified.

4. Set the CVROCPosLimit and CVROCNegLimit to limit the CV rate of change.

   Rate limiting is not imposed when the control variable is in Manual mode unless CVManLimiting is set.

See also

IMC Function Block Tuning on page 201

IMC Function Block Model Initialization

A Model Initialization occurs:

- during First Scan of the block
- when the ModelInit request parameter is set
- when DeltaT changes

You may need to manually adjust the internal model parameters or the response time constants. You can do so by changing the appropriate parameters and setting the appropriate ModelInit bit. The internal states of the function block will be initialized, and the bit will automatically reset.

For example, if you modify the IMC function block Model Gain for CV - PV, set the ModelInit parameter to TRUE to initialize the CV - PV internal model parameters and for the new model gain to take effect.

IMC Function Block Tuning

The function block is equipped with an internal tuner (modeler). The purpose of the tuner is to identify the process model parameters and to use these parameters as internal model parameters (gain, time constant, and deadtime). The tuner also calculates an optimal response-time constant.

Set the tuner by configuring the following parameters.

ProcessType	Integral (level, position control) or nonintegrating (flow, pressure control)
ProcessGainSign	Set to indicate a negative process gain (increase in output causes a decrease in PV); reset to indicate a positive process gain (increase in output causes an increase in PV).
ResponseSpeed	Slow, medium, or fast, based on control variable.
NoiseLevel	An estimate of noise level on PV-low, medium, or high such that the tuner can distinguish which PV change is a random noise and which is caused by the CV step change.
StepSize	A nonzero positive or negative number defining the magnitude of CV step change in either positive or negative direction, respectively.
PVTuneLimit	(Only for integrating process type) in PV engineering units, defines how much of PV change that is caused by CV change to tolerate before aborting the tuning test due to exceeding this limit.

The tuner is started by setting the AtuneStart bit. You can stop the tuning by setting the AtuneAbort bit. After the tuning is completed successfully, the GainTuned, TCTuned, DTTuned, and RespTCTuned parameters are updated with the tuning results, and the AtuneStatus code is set to indicate complete.

You can copy these parameters to the ModelGain, ModelTC, and ResponseTC, respectively, by setting the AtuneUseModel bit. The function block will automatically initialize the internal variables and continue normal operation. It will automatically reset the AtuneUseModel bit.

See also

IMC Function Block Tuning Procedure on page 202

IMC Function Block Tuning Errors on page 202

IMC Function Block Tuning Errors

If an error occurs during the tuning procedure, the tuning is aborted, and the AtuneStatus bit is set. You can abort the tuning by setting the AtuneAbort bit.

After an abort, the CV will assume its value before the step change, and the GainTuned, TCTuned, DTTuned, and RespTCTuned parameters are not updated. The AtuneStatus parameter identifies the reason for the abort.

IMC Function Block Tuning Procedure

Follow these steps to configure the tuner.

1. Put the CV into Manual mode.
2. Set the AtuneStart parameter.

   The tuner starts collecting PV and CV data for noise calculation.
3. After collecting 60 samples (60*DeltaT) period, the tuner adds StepSize to the CV.

   After successfully collecting the PV data as a result of the CV step change, the CV assumes its value before the step change and the AtuneStatus, GainTuned, TCTuned, DTTuned, and RespTCTuned parameters are updated.
4. Set the AtuneUseModel parameter to copy the tuned parameters to the model parameters.

   The function block then resets the AtuneUseModel parameter.

After a successful AutoTuneDone, the Atune parameter is set to one (1). Tuning completed successfully.

Modular Multivariable Control (MMC)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

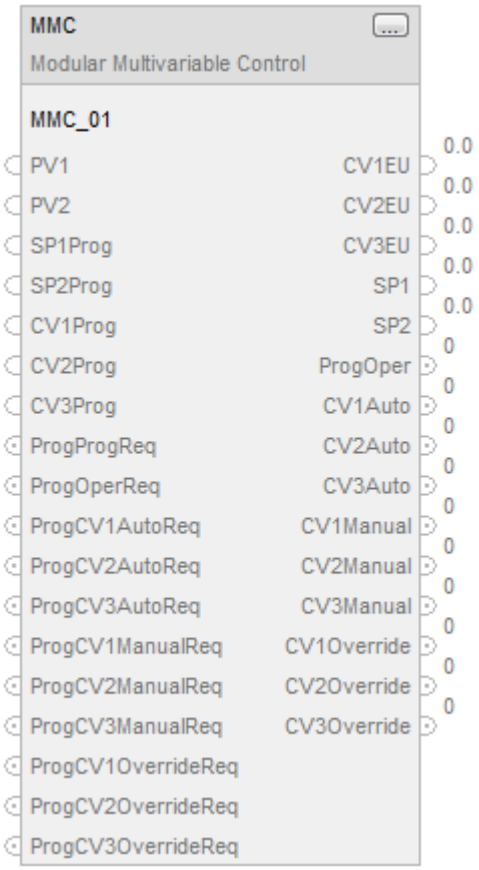
The MMC instruction controls two process variables to their setpoints using up to three control variables. The MMC instruction calculates the control variables (CV1, CV2, and CV3) in the auto mode based on the PV1 -SP1, PV2 -SP2 deviation, internal model, and tuning.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

MMC(MMC_tag);

Operands

Function Block

Operands:	Type	Format	Description
MMC tag	MODULAR MULTIVARIABLE CONTROL	structure	MMC Structure

Structured Text

Operands:	Type	Format	Description
MMC tag	MODULAR MULTIVARIABLE CONTROL	structure	MMC Structure

Structure

The following table describes the input parameters in the MMC function block.

Input Parameters	Data Type	Description	Values
EnableIn	BOOL	Enable Input. If False, the function block will not execute and outputs are not updated.	Default=TRUE
PV1	REAL	Scaled process variable input 1. This value is typically read from an analog input module.	Valid = any float Default = 0.0
PV2	REAL	Scaled process variable input 2. This value is typically read from an analog input module.	Valid = any float Default = 0.0
PV1Fault	BOOL	PV1 bad health indicator. If PV1 is read from an analog input, then PV1Fault will normally be controlled by the analog input fault status. If PVFault is TRUE, it indicates an error on the input module, set bit in Status.FALSE = Good Health	Default = FALSE
PV2Fault	BOOL	PV2 bad health indicator. If PV2 is read from an analog input, then PV2Fault will normally be controlled by the analog input fault status. If PVFault is TRUE, it indicates an error on the input module, set bit in Status.FALSE = Good Health	Default = FALSE
PV1EUMax	REAL	Maximum scaled value for PV1. The value of PV1 and SP1 that corresponds to 100% span of the Process Variable. If PV1EUMax ≤ PV1EUMin, set bit in Status.	Valid = PV1EUMin < PV1EUMax <= maximum positive float Default = 100.0

Input Parameters	Data Type	Description	Values
PV2EUMax	REAL	Maximum scaled value for PV2. The value of PV2 and SP2 that corresponds to 100% span of the Process Variable. If PV2EUMax ≤ PV2EUMin, set bit in Status.	Valid = PV2EUMin < PV2EUMax <= maximum positive float Default = 100.0
PV1UEMin	REAL	Minimum scaled value for PV1. The value of PV1 and SP1 that corresponds to 0% span of the Process Variable. If PV1EUMax ≤ PV1EUMin, set bit in Status.	Valid = maximum negative float <= PV1EUMin < PV1EUMax Default = 0.0
PV2UEMin	REAL	Minimum scaled value for PV2. The value of PV2 and SP2 that corresponds to 0% span of the Process Variable. If PV1EUMax ≤ PV1EUMin, set bit in Status.	Valid = maximum negative float <= PV2EUMin < PV2EUMax Default = 0.0
SP1Prog	REAL	SP1 Program value, scaled in PV units. SP1 is set to this value when Program control.	Valid = SP1LLimit to SP1HLimit Default = 0.0
SP2Prog	REAL	SP2 Program value, scaled in PV units. SP2 is set to this value when Program control.	Valid = SP2LLimit to SP2HLimit Default = 0.0
SP1Oper	REAL	SP1 Operator value, scaled in PV units. SP1 set to this value when Operator control. If value of SP1Prog or SP1Oper < SP1LLimit or > SP1HLimit, set bit in Status and limit value used for SP.	Valid = SP1LLimit to SP1HLimit Default = 0.0
SP2Oper	REAL	SP2 Operator value, scaled in PV units. SP2 set to this value when Operator control. If value of SP2Prog or SP2Oper < SP2LLimit or > SP2HLimit, set bit in Status and limit value used for SP.	Valid = SP2LLimit to SP2HLimit Default = 0.0
SP1HLimit	REAL	SP1 high limit value, scaled in PV units. • If SP1LLimit < PV1EUMin, or SP1HLimit > PV1EUMax, set bit in Status. • If SP1HLimit < SP1LLimit, set bit in Status and limit SP by using the value of S1PLLimit.	Valid = SP1LLimit to PV1EUMax Default = 100.0
SP2HLimit	REAL	SP2 high limit value, scaled in PV units. • If SP2LLimit < PV2EUMin, or SP2HLimit > PV2EUMax, set bit in Status. • If SP2HLimit < SP2LLimit, set bit in Status and limit SP by using the value of SPLLimit.	Valid = SP2LLimit to PV2EUMax Default = 100.0
SP1LLimit	REAL	SP1 low limit value, scaled in PV units. • If SP1LLimit < PV1EUMin, or SP1HLimit > PV1EUMax, set bit in Status. • If SP1HLimit < SP1LLimit, set bit in Status and limit SP by using the value of SPLLimit.	Valid = PV1EUMin to SP1HLimit Default = 0.0
SP2LLimit	REAL	SP2 low limit value, scaled in PV units. • If SP2LLimit < PV2EUMin, or SP2HLimit > PV2EUMax, set bit in Status. • If SP2HLimit < SP2LLimit, set bit in Status and limit SP by using the value of SPLLimit.	Valid = PV2EUMin to SP2HLimit Default = 0.0

Input Parameters	Data Type	Description	Values
CV1Fault	BOOL	Control variable 1 bad health indicator. If CV1EU controls an analog output, then CV1Fault will normally come from the analog output's fault status. If CV1Fault is TRUE, it indicates an error on the output module, set bit in Status.FALSE = Good Health	Default = FALSE
CV2Fault	BOOL	Control variable 2 bad health indicator. If CV2EU controls an analog output, then CV2Fault will normally come from the analog output's fault status. If CV2Fault is TRUE, it indicates an error on the output module, set bit in Status.FALSE = Good Health	Default = FALSE
CV3Fault	BOOL	Control variable 3 bad health indicator. If CV3EU controls an analog output, then CV3Fault will normally come from the analog output's fault status. If CV3Fault is TRUE, it indicates an error on the output module, set bit in Status.FALSE = Good Health	Default = FALSE
CV1InitReq	BOOL	CV1 initialization request. While TRUE, set CV1EU to the value of CV1InitValue. This signal will normally be controlled by the In Hold status on the analog output module controlled by CV1EU or from the InitPrimary output of a secondary loop. The instruction initialization is disabled when CV1Faulted or CV1EUSpanInv are TRUE.	Default = FALSE
CV2InitReq	BOOL	CV2 initialization request. While TRUE, set CV2EU to the value of CV2InitValue. This signal will normally be controlled by the In Hold status on the analog output module controlled by CV2EU or from the InitPrimary output of a secondary loop. The instruction initialization is disabled when CV2Faulted or CV2EUSpanInv are TRUE.	Default = FALSE
CV3InitReq	BOOL	CV3 initialization request. While TRUE, set CV3EU to the value of CV3InitValue. This signal will normally be controlled by the In Hold status on the analog output module controlled by CV3EU or from the InitPrimary output of a secondary loop. The instruction initialization is disabled when CV3Faulted or CV3EUSpanInv are TRUE.	Default = FALSE

Input Parameters	Data Type	Description	Values
CV1InitValue	REAL	CV1EU initialization value, scaled in CV1EU units. When CV1Initializing is TRUE set CV1EU equal to CV1InitValue and CV1 to the corresponding percentage value. CV1InitValue will normally come from the feedback of the analog output controlled by CV1EU or from the setpoint of a secondary loop. The instruction initialization is disabled when CV1Faulted or CV1EUSpanInv are TRUE.	Valid = any float Default = 0.0
CV2InitValue	REAL	CV2EU initialization value, scaled in CV2EU units. When CV2Initializing is TRUE set CV2EU equal to CV2InitValue and CV2 to the corresponding percentage value. CV2InitValue will normally come from the feedback of the analog output controlled by CV2EU or from the setpoint of a secondary loop. The instruction initialization is disabled when CV2Faulted or CV2EUSpanInv are TRUE.	Valid = any float Default = 0.0
CV3InitValue	REAL	CV3EU initialization value, scaled in CV3EU units. When CV3Initializing is TRUE set CV3EU equal to CV3InitValue and CV3 to the corresponding percentage value. CV3InitValue will normally come from the feedback of the analog output controlled by CV3EU or from the setpoint of a secondary loop. The instruction initialization is disabled when CV3Faulted or CV3EUSpanInv are TRUE.	Valid = any float Default = 0.0
CV1Prog	REAL	CV1 Program-Manual value. CV1 is set to this value when in Program control and Manual mode.	Valid = 0.0 through 100.0 Default = 0.0
CV2Prog	REAL	CV2 Program-Manual value. CV2 is set to this value when in Program control and Manual mode.	Valid = 0.0 through 100.0 Default = 0.0
CV3Prog	REAL	CV3 Program-Manual value. CV3 is set to this value when in Program control and Manual mode.	Valid = 0.0 through 100.0 Default = 0.0
CV1Oper	REAL	CV1 Operator-Manual value. • CV1 is set to this value when in Operator control and Manual mode. If not Operator-Manual mode, set CV1Oper to the value of CV1 at the end of each function block execution. • If value of CV1Prog or CV1Oper < 0 or > 100, or < CV1LLimit or > CV1HLimit when CVManLimiting is TRUE, set unique Status bit and limit value used for CV1.	Valid = 0.0 through 100.0 Default = 0.0

Input Parameters	Data Type	Description	Values
CV2Oper	REAL	CV2 Operator-Manual value. • CV2 is set to this value when in Operator control and Manual mode. If not Operator-Manual mode, set CV2Oper to the value of CV2 at the end of each function block execution. • If value of CV2Prog or CV2Oper < 0 or > 100, or < CV2LLimit or > CV2HLimit when CVManLimiting is TRUE, set unique Status bit and limit value used for CV2.	Valid = 0.0 through 100.0 Default = 0.0
CV3Oper	REAL	CV3 Operator-Manual value. • CV3 is set to this value when in Operator control and Manual mode. If not Operator-Manual mode, set CV3Oper to the value of CV3 at the end of each function block execution. • If value of CV3Prog or CV3Oper < 0 or > 100, or < CV3LLimit or > CV3HLimit when CVManLimiting is TRUE, set unique Status bit and limit value used for CV3.	Valid = 0.0 through 100.0 Default = 0.0
CV1OverrideValue	REAL	CV1 Override value. • CV1 set to this value when in Override mode. This value should correspond to a safe state output of the loop. • If value of CV1OverrideValue < 0 or >100, set unique Status bit and limit value used for CV1.	Valid = 0.0 through 100.0 Default = 0.0
CV2OverrideValue	REAL	CV2 Override value. • CV2 set to this value when in Override mode. This value should correspond to a safe state output of the loop. • If value of CV2OverrideValue < 0 or >100, set unique Status bit and limit value used for CV2.	Valid = 0.0 through 100.0 Default = 0.0
CV3OverrideValue	REAL	CV3 Override value. CV3 set to this value when in Override mode. This value should correspond to a safe state output of the loop. If value of CV3OverrideValue < 0 or >100, set unique Status bit and limit value used for CV3.	Valid = 0.0 through 100.0 Default = 0.0
CVManLimiting	BOOL	Limit CV(n), where (n) can be 1, 2, or 3, in Manual mode. If Manual mode and CVManLimiting is TRUE, CV(n) will be limited by the CV(n)HLimit and CV(n)LLimit values.	Default = FALSE
CV1EUMax	REAL	Maximum value for CV1EU. The value of CV1EU that corresponds to 100% CV1. If CVEUMax = CVEUMin, set bit in Status.	Valid = any float Default = 100.0
CV2EUMax	REAL	Maximum value for CV2EU. The value of CV2EU that corresponds to 100% CV2. If CVEUMax = CVEUMin, set bit in Status.	Valid = any float Default = 100.0

Input Parameters	Data Type	Description	Values
CV3EUMax	REAL	Maximum value for CV3EU. The value of CV3EU that corresponds to 100% CV3. If CVEUMax = CVEUMin, set bit in Status.	Valid = any float Default = 100.0
CV1EUMin	REAL	Minimum value of CV1EU. The value of CV1EU that corresponds to 0% CV1. If CVEUMax = CVEUMin, set bit in Status.	Valid = any float Default = 0.0
CV2EUMin	REAL	Minimum value of CV2EU. The value of CV2EU that corresponds to 0% CV2. If CVEUMax = CVEUMin, set bit in Status.	Valid = any float Default = 0.0
CV3EUMin	REAL	Minimum value of CV3EU. The value of CV3EU that corresponds to 0% CV3. If CVEUMax = CVEUMin, set bit in Status.	Valid = any float Default = 0.0
CV1HLimit	REAL	CV1 high limit value. This is used to set the CV1HAlarm output. It is also used for limiting CV1 when in Auto mode or in Manual if CVManLimiting is TRUE. • If CV1LLimit < 0, if CV1HLimit > 100, if CV1HLimit < CV1LLimit, set bit in Status. • If CV1HLimit < CV1LLimit, limit CV1 by using the value of CV1LLimit.	Valid = CV1LLimit < CV1HLimit ≤ 100.0 Default = 100.0
CV2HLimit	REAL	CV2 high limit value. This is used to set the CV2HAlarm output. It is also used for limiting CV2 when in Auto mode or in Manual if CVManLimiting is TRUE. • If CV2LLimit < 0, if CV2HLimit > 100, if CV2HLimit < CV2LLimit, set bit in Status. • If CV2HLimit < CV2LLimit, limit CV2 by using the value of CV2LLimit.	Valid = CV2LLimit < CV2HLimit ≤ 100.0 Default = 100.0
CV3HLimit	REAL	CV3 high limit value. This is used to set the CV3HAlarm output. It is also used for limiting CV3 when in Auto mode or in Manual if CVManLimiting is TRUE. • If CV3LLimit < 0, if CV3HLimit > 100, if CV3HLimit < CV3LLimit, set bit in Status. • If CV3HLimit < CV3LLimit, limit CV3 by using the value of CV3LLimit.	Valid = CV3LLimit < CV3HLimit ≤ 100.0 Default = 100.0
CV1LLimit	REAL	CV1 low limit value. This is used to set the CV1LAlarm output. It is also used for limiting CV1 when in Auto mode or in Manual mode if CVManLimiting is TRUE. • If CV1LLimit < 0, if CV1HLimit > 100, if CV1HLimit < CV1LLimit, set bit in Status. • If CV1HLimit < CV1LLimit, limit CV1 by using the value of CV1LLimit.	Valid = 0.0 ≤ CV1LLimit < CV1HLimit Default = 0.0
CV2LLimit	REAL	CV2 low limit value. This is used to set the CV2LAlarm output. It is also used for limiting CV2 when in Auto mode or in Manual mode if CVManLimiting is TRUE. • If CV2LLimit < 0, if CV2HLimit > 100, if CV2HLimit < CV2LLimit, set bit in Status. • If CV2HLimit < CV2LLimit, limit CV2 by using the value of CV2LLimit.	Valid = 0.0 ≤ CV2LLimit < CV1HLimit Default = 0.0

Input Parameters	Data Type	Description	Values
CV3LLimit	REAL	CV3 low limit value. This is used to set the CV3LAlarm output. It is also used for limiting CV3 when in Auto mode or in Manual mode if CVManLimiting is TRUE. • If CV3LLimit < 0, if CV3HLimit > 100, if CV3HLimit < CV3LLimit, set bit in Status. • If CV3HLimit < CV3LLimit, limit CV by using the value of CV3LLimit.	Valid = 0.0 ≤ CV3LLimit < CV1HLimit Default = 0.0
CV1ROCPosLimit	REAL	CV1 rate of change limit, in percent per second. Rate of change limiting is only used when in Auto mode or in Manual mode if CVManLimiting is TRUE. A value of zero disables CV1 ROC limiting. If value of CV1ROCLimit < 0, set bit in Status and disable CV1 ROC limiting.	Valid = 0.0 to maximum positive float Default = 0.0
CV2ROCPosLimit	REAL	CV2 rate of change limit, in percent per second. Rate of change limiting is only used when in Auto mode or in Manual mode if CVManLimiting is TRUE. A value of zero disables CV2 ROC limiting. If value of CV2ROCLimit < 0, set bit in Status and disable CV2 ROC limiting.	Valid = 0.0 to maximum positive float Default = 0.0
CV3ROCPosLimit	REAL	CV3 rate of change limit, in percent per second. Rate of change limiting is only used when in Auto mode or in Manual mode if CVManLimiting is TRUE. A value of zero disables CV3 ROC limiting. If value of CV3ROCLimit < 0, set bit in Status and disable CV3 ROC limiting.	Valid = 0.0 to maximum positive float Default = 0.0
CV1ROCNegLimit	REAL	CV1 rate of change limit, in percent per second. Rate of change limiting is only used when in Auto mode or in Manual mode if CVManLimiting is TRUE. A value of zero disables CV1 ROC limiting. If value of CV1ROCLimit < 0, set bit in Status and disable CV1 ROC limiting.	Valid = 0.0 to maximum positive float Default = 0.0
CV2ROCNegLimit	REAL	CV2 rate of change limit, in percent per second. Rate of change limiting is only used when in Auto mode or in Manual mode if CVManLimiting is TRUE. A value of zero disables CV2 ROC limiting. If value of CV2ROCLimit < 0, set bit in Status and disable CV2 ROC limiting.	Valid = 0.0 to maximum positive float Default = 0.0
CV3ROCNegLimit	REAL	CV3 rate of change limit, in percent per second. Rate of change limiting is only used when in Auto mode or in Manual mode if CVManLimiting is TRUE. A value of zero disables CV3 ROC limiting. If value of CV3ROCLimit < 0, set bit in Status and disable CV3 ROC limiting.	Valid = 0.0 to maximum positive float Default = 0.0

Input Parameters	Data Type	Description	Values
CV1HandFB	REAL	CV1 HandFeedback value. CV1 set to this value when in Hand mode and CV1HandFBFault is FALSE (good health). This value would typically come from the output of a field mounted hand/auto station and would be used to generate a bumpless transfer out of Hand mode. If value of CV1HandFB < 0 or > 100, set unique Status bit and limit value used for CV1.	Valid = 0.0 through 100.0 Default = 0.0
CV2HandFB	REAL	CV2 HandFeedback value. CV2 set to this value when in Hand mode and CV2HandFBFault is FALSE (good health). This value would typically come from the output of a field mounted hand/auto station and would be used to generate a bumpless transfer out of Hand mode. If value of CV2HandFB < 0 or > 100, set unique Status bit and limit value used for CV2.	Valid = 0.0 through 100.0 Default = 0.0
CV3HandFB	REAL	CV3 HandFeedback value. CV3 set to this value when in Hand mode and CV3HandFBFault is FALSE (good health). This value would typically come from the output of a field mounted hand/auto station and would be used to generate a bumpless transfer out of Hand mode. If value of CV3HandFB < 0 or > 100, set unique Status bit and limit value used for CV3.	Valid = 0.0 through 100.0 Default = 0.0
CV1HandFBFault	BOOL	CV1HandFB value bad health indicator. If the CV1HandFB value is read from an analog input, then CV1HandFBFault will normally be controlled by the status of the analog input channel. If CV1HandFBFault is TRUE, it indicates an error on the input module, set bit in Status. FALSE = Good Health	Default = FALSE
CV2HandFBFault	BOOL	CV2HandFB value bad health indicator. If the CV2HandFB value is read from an analog input, then CV2HandFBFault will normally be controlled by the status of the analog input channel. If CV2HandFBFault is TRUE, it indicates an error on the input module, set bit in Status. FALSE = Good Health	Default = FALSE

Input Parameters	Data Type	Description	Values
CV3HANDFBFault	BOOL	CV3HandFB value bad health indicator. If the CV3HandFB value is read from an analog input, then CV3HandFBFault will normally be controlled by the status of the analog input channel. If CV3HandFBFault is TRUE, it indicates an error on the input module, set bit in Status. FALSE = Good Health	Default = FALSE
CV1Target	REAL	Target value for control variable output 1.	Valid = 0.0 through 100.0 Default = 0.0
CV2Target	REAL	Target value for control variable output 2.	Valid = 0.0 through 100.0 Default = 0.0
CV3Target	REAL	Target value for control variable output 3.	Valid = 0.0 through 100.0 Default = 0.0
CV1WindupHIn	BOOL	CV1Windup high request. When TRUE, CV1 will not be allowed to increase in value. This signal will typically be the CV1WindupHOut output from a secondary loop.	Default = FALSE
CV2WindupHIn	BOOL	CV2Windup high request. When TRUE, CV2 will not be allowed to increase in value. This signal will typically be the CV2WindupHOut output from a secondary loop.	Default = FALSE
CV3WindupHIn	BOOL	CV3Windup high request. When TRUE, CV3 will not be allowed to increase in value. This signal will typically be the CV3WindupHOut output from a secondary loop.	Default = FALSE
CV1WindupLIn	BOOL	CV1 Windup low request. When TRUE, CV1 will not be allowed to decrease in value. This signal will typically be the CV1WindupLOut output from a secondary loop.	Default = FALSE
CV2WindupLIn	BOOL	CV2 Windup low request. When TRUE, CV2 will not be allowed to decrease in value. This signal will typically be the CV2WindupLOut output from a secondary loop.	Default = FALSE
CV3WindupLIn	BOOL	CV3 Windup low request. When TRUE, CV3 will not be allowed to decrease in value. This signal will typically be the CV3WindupLOut output from a secondary loop.	Default = FALSE
GainEUSpan	BOOL	ModelGain units in EU or as % of span.	Default = FALSE FALSE = Gain in % of span
CV1PV1ProcessGainSign	BOOL	Used only for Autotuning. Sign of the process gain (Delta PV1/Delta CV1). • Set to indicate a negative process gain (increase in output causes a decrease in PV1). • Reset to indicate a positive process gain (increase in output causes an increase in P1V).	Default = FALSE

Input Parameters	Data Type	Description	Values
CV2PV1ProcessGainSign	BOOL	Used only for Autotuning. Sign of the process gain (Delta PV1/Delta CV2). • Set to indicate a negative process gain (increase in output causes a decrease in PV1). • Reset to indicate a positive process gain (increase in output causes an increase in PV1).	Default = FALSE
CV3PV1ProcessGainSign	BOOL	Used only for Autotuning. Sign of the process gain (Delta PV1/Delta CV3). • Set to indicate a negative process gain (increase in output causes a decrease in PV1). • Reset to indicate a positive process gain (increase in output causes an increase in PV1).	Default = FALSE
CV1PV2ProcessGainSign	BOOL	Used only for Autotuning. Sign of the process gain (Delta PV2/Delta CV1). • Set to indicate a negative process gain (increase in output causes a decrease in PV2). • Reset to indicate a positive process gain (increase in output causes an increase in PV2).	Default = FALSE
CV1PV2ProcessGainSign	BOOL	Used only for Autotuning. Sign of the process gain (Delta PV2/Delta CV2). • Set to indicate a negative process gain (increase in output causes a decrease in PV2). • Reset to indicate a positive process gain (increase in output causes an increase in PV2).	Default = FALSE
CV1PV2ProcessGainSign	BOOL	Used only for Autotuning. Sign of the process gain (Delta PV2/Delta CV3). • Set to indicate a negative process gain (increase in output causes a decrease in PV2). • Reset to indicate a positive process gain (increase in output causes an increase in PV2).	Default = FALSE
ProcessType	DINT	Process type selection for both PV1 and PV2 (1=Integrating, 0=non-integrating)	Default = 0
CV1PV1ModelGain	REAL	The internal model gain parameter for CV1 - PV1. Enter a positive or negative gain depending on process direction. If CV1PV1ModelGain = INF or NAN, set bit in Status.	Valid = maximum negative float –> maximum positive float Default = 0.0
CV2PV1ModelGain	REAL	The internal model gain parameter for CV2 - PV1. Enter a positive or negative gain depending on process direction. If CV2PV1ModelGain = INF or NAN, set bit in Status.	Valid = maximum negative float –> maximum positive float Default = 0.0

Input Parameters	Data Type	Description	Values
CV3PV1ModelGain	REAL	The internal model gain parameter for CV3 - PV1. Enter a positive or negative gain depending on process direction. If CV3PV1ModelGain = INF or NAN, set bit in Status.	Valid = maximum negative float –> maximum positive float Default = 0.0
CV1PV2ModelGain	REAL	The internal model gain parameter for CV1 - PV2. Enter a positive or negative gain depending on process direction. If CV1PV2ModelGain = INF or NAN, set bit in Status.	Valid = maximum negative float –> maximum positive float Default = 0.0
CV2PV2ModelGain	REAL	The internal model gain parameter for CV2 - PV2. Enter a positive or negative gain depending on process direction. If CV2PV2ModelGain = INF or NAN, set bit in Status.	Valid = maximum negative float –> maximum positive float Default = 0.0
CV3PV2ModelGain	REAL	The internal model gain parameter for CV3 - PV2. Enter a positive or negative gain depending on process direction. If CV3PV2ModelGain = INF or NAN, set bit in Status.	Valid = maximum negative float –> maximum positive float Default = 0.0
CV1PV1ModelTC	REAL	The internal model time constant for CV1 - PV1 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV2PV1ModelTC	REAL	The internal model time constant for CV2 - PV1 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV3PV1ModelTC	REAL	The internal model time constant for CV3 - PV1 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV1PV2ModelTC	REAL	The internal model time constant for CV1 - PV2 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV2PV2ModelTC	REAL	The internal model time constant for CV2 - PV2 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV3PV2ModelTC	REAL	The internal model time constant for CV3 - PV2 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV1PV1ModelDT	REAL	The internal model deadtime for CV1 - PV1 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV2PV1ModelDT	REAL	The internal model deadtime for CV2 - PV1 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV3PV1ModelDT	REAL	The internal model deadtime for CV3 - PV1 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV1PV2ModelDT	REAL	The internal model deadtime for CV1 - PV2 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0

Input Parameters	Data Type	Description	Values
CV2PV2ModelDT	REAL	The internal model deadtime for CV2 - PV2 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV3PV2ModelDT	REAL	The internal model deadtime for CV3 - PV2 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV1PV1RespTC	REAL	The tuning parameter that determines the speed of the control variable action for CV1 - PV1 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV2PV1RespTC	REAL	The tuning parameter that determines the speed of the control variable action for CV2 - PV1 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV3PV1RespTC	REAL	The tuning parameter that determines the speed of the control variable action for CV3 - PV1 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV1PV2RespTC	REAL	The tuning parameter that determines the speed of the control variable action for CV1 - PV2 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV2PV2RespTC	REAL	The tuning parameter that determines the speed of the control variable action for CV2 - PV2 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
CV3PV2RespTC	REAL	The tuning parameter that determines the speed of the control variable action for CV3 - PV2 in seconds.	Valid = 0.0 to maximum positive float Default = 0.0
PV1Act1stCV	DINT	The first CV to act to compensate for PV1-SP1 deviation. 1=CV1, 2=CV2, 3=CV3	Valid = 1-3 Default = 1
PV1Act2ndCV	DINT	The second CV to act to compensate for PV1-SP1 deviation. 1=CV1, 2=CV2, 3=CV3	Valid = 1-3 Default = 2
PV1Act3rdCV	DINT	The third CV to act to compensate for PV1-SP1 deviation. 1=CV1, 2=CV2, 3=CV3	Valid = 1-3 Default = 3
PV2Act1stCV	DINT	The first CV to act to compensate for PV2-SP2 deviation. 1=CV1, 2=CV2, 3=CV3	Valid = 1-3 Default = 1
PV2Act2ndCV	DINT	The second CV to act to compensate for PV2-SP2 deviation. 1=CV1, 2=CV2, 3=CV3	Valid = 1-3 Default = 2
PV2Act3rdCV	DINT	The third CV to act to compensate for PV2-SP2 deviation. 1=CV1, 2=CV2, 3=CV3	Valid = 1-3 Default = 3
TargetCV	DINT	The CV to be driven to its target value. 1=CV1, 2=CV2, 3=CV3	Valid = 1-3 Default = 3
TargetRespTC	REAL	Determines the speed with which the control variables approach the target values.	Valid = 0.0 to maximum positive float Default = 0.0

Input Parameters	Data Type	Description	Values
PVTracking	BOOL	SP track PV request. Set TRUE to enable SP to track PV. Ignored when in Auto modes. SP will only track PV when all three outputs are in manual. As soon as any output returns to Auto, PVTracking stops.	Default = FALSE
ManualAfterInit	BOOL	Manual mode after initialization request. • When TRUE, the appropriate CV(n), where (n) can be 1, 2, or 3, will be placed in Manual mode when CV(n)Initializing is set TRUE unless the current mode is Override or Hand. • When ManualAfterInit is FALSE, the CV(n) mode will not be changed.	Default = FALSE
ProgProgReq	BOOL	Program Program Request. • Set TRUE by the user program to request Program control. Ignored if ProgOperReq is TRUE. Holding this TRUE and ProgOperReq FALSE can be used to lock the function block into program control. • When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
ProgOperReq	BOOL	Program Operator Request. • Set TRUE by the user program to request Operator control. Holding this TRUE can be used to lock the function block into operator control. • When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
ProgCV1AutoReq	BOOL	Program-Auto mode request for CV1. • Set TRUE by the user program to request Auto mode. • When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
ProgCV2AutoReq	BOOL	Program-Auto mode request for CV2. • Set TRUE by the user program to request Auto mode. • When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
ProgCV3AutoReq	BOOL	Program-Auto mode request for CV3. • Set TRUE by the user program to request Auto mode. • When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
ProgCV1ManualReq	BOOL	Program-Manual mode request for CV1. • Set TRUE by the user program to request Manual mode. • When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE

Input Parameters	Data Type	Description	Values
ProgCV2ManualReq	BOOL	Program-Manual mode request for CV2. • Set TRUE by the user program to request Manual mode. • When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
ProgCV3ManualReq	BOOL	Program-Manual mode request for CV3. • Set TRUE by the user program to request Manual mode. • When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
ProgCV1OverrideReq	BOOL	Program-Override mode request for CV1. • Set TRUE by the user program to request Override mode. • When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
ProgCV2OverrideReq	BOOL	Program-Override mode request for CV2. • Set TRUE by the user program to request Override mode. • When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
ProgCV3OverrideReq	BOOL	Program-Override mode request for CV3. • Set TRUE by the user program to request Override mode. • When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
ProgCV1HandReq	BOOL	Program-Hand mode request for CV1. • Set TRUE by the user program to request Hand mode. This value will usually be read as a digital input from a hand/auto station. • When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
ProgCV2HandReq	BOOL	Program-Hand mode request for CV2. • Set TRUE by the user program to request Hand mode. This value will usually be read as a digital input from a hand/auto station. • When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
ProgCV3HandReq	BOOL	Program-Hand mode request for CV3. • Set TRUE by the user program to request Hand mode. This value will usually be read as a digital input from a hand/auto station. • When ProgValueReset is TRUE, the function block resets the input to FALSE.	Default = FALSE
OperProgReq	BOOL	Operator Program Request. • Set TRUE by the operator interface to request Program control. The function block resets this parameter to FALSE.	Default = FALSE

Input Parameters	Data Type	Description	Values
OperOperReq	BOOL	Operator Operator Request. • Set TRUE by the operator interface to request Operator control. The function block resets this parameter to FALSE.	Default = FALSE
OperCV1AutoReq	BOOL	Operator-Auto mode request for CV1. Set TRUE by the operator interface to request Auto mode. The function block resets this parameter to FALSE.	Default = FALSE
OperCV2AutoReq	BOOL	Operator-Auto mode request for CV2. Set TRUE by the operator interface to request Auto mode. The function block resets this parameter to FALSE.	Default = FALSE
OperCV3AutoReq	BOOL	Operator-Auto mode request for CV3. Set TRUE by the operator interface to request Auto mode. The function block resets this parameter to FALSE.	Default = FALSE
OperCV1ManualReq	BOOL	Operator-Manual mode request for CV1. Set TRUE by the operator interface to request Manual mode. The function block sets this parameter to FALSE.	Default = FALSE
OperCV2ManualReq	BOOL	Operator-Manual mode request for CV2. Set TRUE by the operator interface to request Manual mode. The function block sets this parameter to FALSE.	Default = FALSE
OperCV3ManualReq	BOOL	Operator-Manual mode request for CV3. Set TRUE by the operator interface to request Manual mode. The function block sets this parameter to FALSE.	Default = FALSE
ProgValueReset	BOOL	Reset Program control values. When TRUE, the Prog_xxx_Req inputs are reset to FALSE. When TRUE and Program control, set SP(x)Prog = SP(x) and CV(y)Prog = CV(y), where x = 1,2 and y = 1,2,3	Default = FALSE
TimingMode	DINT	Selects Time Base Execution mode. Value/Description 0 = Periodic mode 1 = Oversample mode 2 = Real time sampling mode For more information about timing modes, see Function Block Attributes.	Valid = 0 through 2 Default = 0
OverSampleDT	REAL	Execution time for Oversample mode.	Valid = 0 to max. TON_Timer elapsed time (4194.303 seconds) Default = 0
RTSTime	DINT	Module update period for Real Time Sampling mode.	Valid = 0 through 32,767 1 count = 1 ms
RTSTimeStamp	DINT	Module time stamp value for Real Time Sampling mode.	Valid = 0 through 32,767 (wraps from 32,767…0) 1 count = 1 ms

Input Parameters	Data Type	Description	Values
PV1TuneLimit	REAL	PV1 tuning limit scaled in the PV1 units. When Autotune is running and predicted PV1 exceeds this limit, the tuning will be aborted.	Valid = any float Default=0
PV2TuneLimit	REAL	PV2 tuning limit scaled in the PV2 units. When Autotune is running and predicted PV2 exceeds this limit, the tuning will be aborted.	Valid = any float Default=0
PV1AtuneTimeLimit	REAL	Maximum time in minutes for PV1 autotune to complete following the CV1 step change. When PV1 autotune exceeds this time, tuning will be aborted.	Valid range: any float > 0. Default = 60 minutes
PV2AtuneTimeLimit	REAL	Maximum time in minutes for PV2 autotune to complete following the CV2 step change. When PV2 autotune exceeds this time, tuning will be aborted.	Valid range: any float > 0. Default = 60 minutes
PV1NoiseLevel	DINT	An estimate of the noise level expected on the PV1 to compensate for it during tuning. The selections are: 0=low, 1=medium, 2=high	Range: 0 through 2 Default=1
PV2NoiseLevel	DINT	An estimate of the noise level expected on the PV2 to compensate for it during tuning. The selections are: 0=low, 1=medium, 2=high	Range: 0 through 2 Default=1
CV1StepSize	REAL	CV1 step size in percent for the tuning step test. Step size is directly added to CV1 subject to high/low limiting.	Range: -100% ... 100% Default=10%
CV2StepSize	REAL	CV2 step size in percent for the tuning step test. Step size is directly added to CV2 subject to high/low limiting.	Range: -100% ... 100% Default=10%
CV3StepSize	REAL	CV3 step size in percent for the tuning step test. Step size is directly added to CV3 subject to high/low limiting.	Range: -100% ... 100% Default=10%
CV1PV1ResponseSpeed	DINT	Desired speed of closed loop response for CV1 - PV1. • Slow response: ResponseSpeed=0 • Medium response: ResponseSpeed=1 • Fast response: ResponseSpeed=2 If ResponseSpeed is less than 0, Slow response is used. If ResponseSpeed is greater than 2, Fast response is used.	Range: 0...2 Default=1
CV2PV1ResponseSpeed	DINT	Desired speed of closed loop response for CV2 PV1. • Slow response: ResponseSpeed=0 • Medium response: ResponseSpeed=1 • Fast response: ResponseSpeed=2 If ResponseSpeed is less than 0, Slow response is used. If ResponseSpeed is greater than 2, Fast response is used.	Range: 0...2 Default=1

Input Parameters	Data Type	Description	Values
CV3PV1ResponseSpeed	DINT	Desired speed of closed loop response for CV3 PV1. • Slow response: ResponseSpeed=0 • Medium response: ResponseSpeed=1 • Fast response: ResponseSpeed=2 If ResponseSpeed is less than 0, Slow response is used. If ResponseSpeed is greater than 2, Fast response is used.	Range: 0 through 2 Default=1
CV1PV2ResponseSpeed	DINT	Desired speed of closed loop response for CV1 PV2. • Slow response: ResponseSpeed=0 • Medium response: ResponseSpeed=1 • Fast response: ResponseSpeed=2 If ResponseSpeed is less than 0, Slow response is used. If ResponseSpeed is greater than 2, Fast response is used.	Range: 0 through 2 Default=1
CV2PV2ResponseSpeed	DINT	Desired speed of closed loop response for CV2 PV2. • Slow response: ResponseSpeed=0 • Medium response: ResponseSpeed=1 • Fast response: ResponseSpeed=2 If ResponseSpeed is less than 0, Slow response is used. If ResponseSpeed is greater than 2, Fast response is used.	Range: 0 through 2 Default=1
CV3PV2ResponseSpeed	DINT	Desired speed of closed loop response for CV3 PV2. • Slow response: ResponseSpeed=0 • Medium response: ResponseSpeed=1 • Fast response: ResponseSpeed=2 If ResponseSpeed is less than 0, Slow response is used. If ResponseSpeed is greater than 2, Fast response is used.	Range: 0 through 2 Default=1
CV1PV1ModelInit	BOOL	Internal model initialization switch for CV1 - PV1. Refer to Function Block Attributes.	Default = FALSE
CV2PV1ModelInit	BOOL	Internal model initialization switch for CV2 - PV1. Refer to Function Block Attributes.	Default = FALSE
CV3PV1ModelInit	BOOL	Internal model initialization switch for CV3 - PV1. Refer to Function Block Attributes.	Default = FALSE
CV1PV2ModelInit	BOOL	Internal model initialization switch for CV1 - PV2. Refer to Function Block Attributes.	Default = FALSE
CV2PV2ModelInit	BOOL	Internal model initialization switch for CV2 - PV2. Refer to Function Block Attributes.	Default = FALSE
CV3PV2ModelInit	BOOL	Internal model initialization switch for CV3 - PV2. Refer to Function Block Attributes.	Default = FALSE
PV1Factor	REAL	Non-integrating model approximation factor for PV1. Only used for integrating process types.	Default = 100
PV2Factor		Non-integrating model approximation factor for PV2. Only used for integrating process types.	Default = 100

Input Parameters	Data Type	Description	Values
AtuneCV1Start	BOOL	Start Autotune request for CV1. Set True to initiate auto tuning of the CV1 output for both PV1 and PV2. Ignored when CV1 is not in Manual mode. The function block resets the input to FALSE.	Default = FALSE
AtuneCV2Start	BOOL	Start Autotune request for CV2. Set True to initiate auto tuning of the CV2 output for both PV1 and PV2. Ignored when CV2 is not in Manual mode. The function block resets the input to FALSE.	Default = FALSE
AtuneCV3Start	BOOL	Start Autotune request for CV3. Set True to initiate auto tuning of the CV3 output for both PV1 and PV2. Ignored when CV3 is not in Manual mode. The function block resets the input to FALSE.	Default = FALSE
AtuneCV1PV1UseModel	BOOL	Use Autotune model request for CV1 - PV1. Set True to replace the current model parameters with the calculated Autotune model parameters. The function block resets the input parameter to FALSE.	Default = FALSE
AtuneCV2PV1UseModel	BOOL	Use Autotune model request for CV2 - PV1. Set True to replace the current model parameters with the calculated Autotune model parameters. The function block resets the input parameter to FALSE.	Default = FALSE
AtuneCV3PV1UseModel	BOOL	Use Autotune model request for CV3 - PV1. Set True to replace the current model parameters with the calculated Autotune model parameters. The function block resets the input parameter to FALSE.	Default = FALSE
AtuneCV1PV2UseModel	BOOL	Use Autotune model request for CV1 - PV2. Set True to replace the current model parameters with the calculated Autotune model parameters. The function block resets the input parameter to FALSE.	Default = FALSE
AtuneCV2PV2UseModel	BOOL	Use Autotune model request for CV2 - PV2. Set True to replace the current model parameters with the calculated Autotune model parameters. The function block resets the input parameter to FALSE.	Default = FALSE
AtuneCV3PV2UseModel	BOOL	Use Autotune model request for CV3 - PV2. Set True to replace the current model parameters with the calculated Autotune model parameters. The function block resets the input parameter to FALSE.	Default = FALSE
AtuneCV1Abort	BOOL	Abort Autotune request for CV1. Set True to abort the auto tuning of CV1 output for both PV1 and PV2. The function block resets input parameter to FALSE.	Default = FALSE
AtuneCV2Abort	BOOL	Abort Autotune request for CV2. Set True to abort the auto tuning of CV2 output or both PV1 and PV2. The function block resets input parameter to FALSE.	Default = FALSE

Input Parameters	Data Type	Description	Values
AtuneCV3Abort	BOOL	Abort Autotune request for CV3. Set True to abort the auto tuning of CV3 output or both PV1 and PV2. The function block resets input parameter to FALSE.	Default = FALSE

The following table describes the output parameters in the MMC function block.

Output Parameters	Data Type	Description	Values
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if any one of CV1EU, CV2EU or CV3EU overflows.	
CV1EU	REAL	Scaled control variable output for CV1. Scaled by using CV1EUMax and CV1EUMin, where CV1EUMax corresponds to 100% and CV1EUMin corresponds to 0%. This output is typically used to control an analog output module or a secondary loop. CV1EU = (CV1 * CV1EUSpan / 100) + CV1EUMin CV1EU span calculation: CV1EUSpan = (CV1EUMax – CV1EUMin)	
CV2EU	REAL	Scaled control variable output for CV2. Scaled by using CV2EUMax and CV2EUMin, where CV2EUMax corresponds to 100% and CV2EUMin corresponds to 0%. This output is typically used to control an analog output module or a secondary loop. CV2EU = (CV2 * CV2EUSpan / 100) + CV2EUMin CV2EU span calculation: CV2EUSpan = (CV2EUMax – CV2EUMin)	
CV3EU	REAL	Scaled control variable output for CV3. Scaled by using CV3EUMax and CV3EUMin, where CV3EUMax corresponds to 100% and CV3EUMin corresponds to 0%. This output is typically used to control an analog output module or a secondary loop. CV3EU = (CV3 * CV3EUSpan / 100) + CV3EUMin CV3EU span calculation: CV3EUSpan = (CV3EUMax – CV3EUMin)	
CV1	REAL	Control variable output for CV1. This value will always be expressed as 0...100%. CV1 is limited by CV1HLimit and CV1LLimit when in Auto mode or in Manual mode if CVManLimiting is TRUE; otherwise limited by 0 and 100%.	
CV2	REAL	Control variable output for CV2. This value will always be expressed as 0...100%. CV2 is limited by CV2HLimit and CV2LLimit when in Auto mode or in Manual mode if CVManLimiting is TRUE; otherwise limited by 0 and 100%.	

Output Parameters	Data Type	Description	Values
CV3	REAL	Control variable output for CV3. This value will always be expressed as 0...100%. CV3 is limited by CV3HLimit and CV3LLimit when in Auto mode or in Manual mode if CVManLimiting is TRUE; otherwise limited by 0 and 100%.	
CV1Initializing	BOOL	Initialization mode indicator for CV1. Set TRUE when CV1InitReq, function blockFirstScan or OLCFirstRun, are TRUE, or on a TRUE to FALSE transition of CV1Fault (bad to good). CV1Initializing is set FALSE after the function block has been initialized and CV1InitReq is no longer TRUE.	
CV2nitializing	BOOL	Initialization mode indicator for CV2. Set TRUE when CV2InitReq, function blockFirstScan or OLCFirstRun, are TRUE, or on a TRUE to FALSE transition of CV2Fault (bad to good). CV2Initializing is set FALSE after the function block has been initialized and CV2InitReq is no longer TRUE.	
CV3nitializing	BOOL	Initialization mode indicator for CV3. Set TRUE when CV3InitReq, function blockFirstScan or OLCFirstRun, are TRUE, or on a TRUE to FALSE transition of CV3Fault (bad to good). CV3Initializing is set FALSE after the function block has been initialized and CV3InitReq is no longer TRUE.	
CV1HAlarm	BOOL	CV1 high alarm indicator. TRUE when the calculated value for CV1 > 100 or CV1HLimit.	
CV2HAlarm	BOOL	CV2 high alarm indicator. TRUE when the calculated value for CV2 > 100 or CV2HLimit.	
CV3HAlarm	BOOL	CV3 high alarm indicator. TRUE when the calculated value for CV3 > 100 or CV3HLimit.	
CV1LAlarm	BOOL	CV1 low alarm indicator. TRUE when the calculated value for CV1 < 0 or CV1LLimit.	
CV2LAlarm	BOOL	CV2 low alarm indicator. TRUE when the calculated value for CV2 < 0 or CV2LLimit.	
CV3LAlarm	BOOL	CV3 low alarm indicator. TRUE when the calculated value for CV3 < 0 or CV3LLimit.	
CV1ROCPosAlarm	BOOL	CV1 rate of change alarm indicator. TRUE when the calculated rate of change for CV1 exceeds CV1ROCPosLimit.	
CV2ROCPosAlarm	BOOL	CV2 rate of change alarm indicator. TRUE when the calculated rate of change for CV2 exceeds CV2ROCPosLimit.	
CV3ROCPosAlarm	BOOL	CV3 rate of change alarm indicator. TRUE when the calculated rate of change for CV3 exceeds CV3ROCPosLimit.	
CV1ROCNegAlarm	BOOL	CV1 rate of change alarm indicator. TRUE when the calculated rate of change for CV1 exceeds CV1ROCNegLimit.	

Output Parameters	Data Type	Description	Values
CV2ROCNegAlarm	BOOL	CV2 rate of change alarm indicator. TRUE when the calculated rate of change for CV2 exceeds CV2ROCNegLimit.	
CV3ROCNegAlarm	BOOL	CV3 rate of change alarm indicator. TRUE when the calculated rate of change for CV3 exceeds CV3ROCNegLimit.	
SP1	REAL	Current setpoint 1 value. The value of SP1 is used to control CV when in the Auto or the PV1 Tracking mode, scaled in PV1 units.	
SP2	REAL	Current setpoint 2 value. The value of SP2 is used to control CV when in the Auto or the PV2 Tracking mode, scaled in PV2 units.	
SP1Percent	REAL	The value of SP1 expressed in percent of span of PV1. SP1Percent = ((SP1 – PV1EUMin) * 100) / PV1Span where PV1Span = PV1EUMax – PV1EUMin	
SP2Percent		The value of SP2 expressed in percent of span of PV2. SP2Percent = ((SP2 – PV2EUMin) * 100) / PV2Span where PV2Span = PV2EUMax – PV1EUMin	
SP1HAlarm	BOOL	SP1 high alarm indicator. TRUE when the SP1 >= SP1HLimit.	
SP2HAlarm	BOOL	SP2 high alarm indicator. TRUE when the SP2 >= SP2HLimit.	
SP1LAlarm	BOOL	SP1 low alarm indicator. TRUE when the SP1 <= SP1LLimit.	
SP2LAlarm	BOOL	SP2 low alarm indicator. TRUE when the SP2 <= SP2LLimit.	
PV1Percent	REAL	PV1 expressed in percent of span. PV1Percent = ((PV1 – PV1EUMin) * 100) / PV1Span PV1 Span calculation: PV1Span = (PV1EUMax - PV1EUMin)	
PV2Percent	REAL	PV2 expressed in percent of span. PV2Percent = ((PV2 – PV2EUMin) * 100) / PV2Span PV2 Span calculation: PV2Span = (PV2EUMax – PV2EUMin)	
E1	REAL	Process1 error. Difference between SP1 and PV1, scaled in PV1 units.	
E2	REAL	Process2 error. Difference between SP2 and PV2, scaled in PV2 units.	
E1Percent	REAL	The error expressed as a percent of span for process 1.	
E2Percent	REAL	The error expressed as a percent of span for process 2.	

Output Parameters	Data Type	Description	Values
CV1WindupHOut	BOOL	CV1 Windup high indicator. CV1WindupHOut is set to true when • SP1HAlarm or SP2HAlarm is true or • ModelGain is positive and CV1HAlarm is true or • ModelGain is negative and CV1LAlarm is true. This signal will typically be used by the WindupHIn input to limit the windup of the CV1 output on a primary loop.	
CV2WindupHOut	BOOL	CV2 Windup high indicator. CV2WindupHOut is set to true when • SP1HAlarm or SP2HAlarm is true or • ModelGain is positive and CV2HAlarm is true or • ModelGain is negative and CV2LAlarm is true. This signal will typically be used by the WindupHIn input to limit the windup of the CV2 output on a primary loop.	
CV3WindupHOut	BOOL	CV3 Windup high indicator. CV3WindupHOut is set to true when • SP1HAlarm or SP2HAlarm is true or • ModelGain is positive and CV3HAlarm is true or • ModelGain is negative and CV3LAlarm is true. This signal will typically be used by the WindupHIn input to limit the windup of the CV3 output on a primary loop.	
CV1WindupLOut	BOOL	CV1 Windup low indicator. CV1WindupLOut is set to true when • SP1LAlarm or SP2LAlarm is true or • ModelGain is positive and CV1LAlarm is true or • ModelGain is negative and CV1HAlarm is true. This signal will typically be used by the WindupLIn input to limit the windup of the CV1 output on a primary loop.	
CV2WindupLOut	BOOL	CV2 Windup low indicator. CV2WindupLOut is set to true when • SP1LAlarm or SP2LAlarm is true or • ModelGain is positive and CV2LAlarm is true or • ModelGain is negative and CV2HAlarm is true. This signal will typically be used by the WindupLIn input to limit the windup of the CV2 output on a primary loop.	

Output Parameters	Data Type	Description	Values
CV3WindupLOut	BOOL	CV3 Windup low indicator. CV3WindupLOut is set to true when • SP1LAlarm or SP2LAlarm is true or • ModelGain is positive and CV3LAlarm is true or • ModelGain is negative and CV3HAlarm is true. This signal will typically be used by the WindupLIn input to limit the windup of the CV3 output on a primary loop.	
ProgOper	BOOL	Program/Operator control indicator. TRUE when in Program control. FALSE when in Operator control.	
CV1Auto	BOOL	Auto mode indicator for CV1. TRUE when CV1 in the Auto mode.	
CV2Auto	BOOL	Auto mode indicator for CV2. TRUE when CV2 in the Auto mode.	
CV2Auto	BOOL	Auto mode indicator for CV3. TRUE when CV3 in the Auto mode.	
CV1Manual	BOOL	Manual mode indicator for CV1. TRUE when CV1 in the Manual mode.	
CV2Manual	BOOL	Manual mode indicator for CV2. TRUE when CV2 in the Manual mode.	
CV3Manual	BOOL	Manual mode indicator for CV3. TRUE when CV3 in the Manual mode.	
CV1Override	BOOL	Override mode indicator for CV1. TRUE when CV1 in the Override mode.	
CV2Override	BOOL	Override mode indicator for CV2. TRUE when CV2 in the Override mode.	
CV3Override	BOOL	Override mode indicator for CV3. TRUE when CV3 in the Override mode.	
CV1Hand	BOOL	Hand mode indicator for CV1. TRUE when CV1 in the Hand mode.	
CV2Hand	BOOL	Hand mode indicator for CV2. TRUE when CV2 in the Hand mode.	
CV3Hand	BOOL	Hand mode indicator for CV3. TRUE when CV3 in the Hand mode.	
DeltaT	REAL	Elapsed time between updates in seconds.	
CV1StepSizeUsed	REAL	Actual CV1 step size used for tuning.	
CV2StepSizeUsed	REAL	Actual CV2 step size used for tuning.	
CV3StepSizeUsed	REAL	Actual CV3 step size used for tuning.	
CV1PV1GainTuned	REAL	The calculated value of the internal model gain for CV1 - PV1 after tuning is completed.	
CV2PV1GainTuned	REAL	The calculated value of the internal model gain for CV2 - PV1 after tuning is completed.	
CV3PV1GainTuned	REAL	The calculated value of the internal model gain for CV3 - PV1 after tuning is completed.	
CV1PV2GainTuned	REA:	The calculated value of the internal model gain for CV1 - PV2 after tuning is completed.	

Output Parameters	Data Type	Description	Values
CV2PV2GainTuned	REAL	The calculated value of the internal model gain for CV2 - PV2 after tuning is completed.	
CV3PV2GainTuned	REA:	The calculated value of the internal model gain for CV3 - PV2 after tuning is completed.	
CV1PV1TCTuned	REAL	The calculated value of the internal model time constant for CV1 - PV1 after tuning is completed.	
CV2PV1TCTuned	REAL	The calculated value of the internal model time constant for CV2 - PV1 after tuning is completed.	
CV3PV1TCTuned	REAL	The calculated value of the internal model time constant for CV3 - PV1 after tuning is completed.	
CV1PV2TCTuned	REAL	The calculated value of the internal model time constant for CV1 - PV2 after tuning is completed.	
CV2PV2TCTuned	REAL	The calculated value of the internal model time constant for CV2 - PV2 after tuning is completed.	
CV3PV2TCTuned	REAL	The calculated value of the internal model time constant for CV3 - PV2 after tuning is completed.	
CV1PV1DTTuned	REAL	The calculated value of the internal model deadtime for CV1 - PV1 after tuning is completed.	
CV2PV1DTTuned	REAL	The calculated value of the internal model deadtime for CV2 - PV1 after tuning is completed.	
CV3PV1DTTuned	REAL	The calculated value of the internal model deadtime for CV3 - PV1 after tuning is completed.	
CV1PV2DTTuned	REAL	The calculated value of the internal model deadtime for CV1 - PV2 after tuning is completed.	
CV2PV2DTTuned	REAL	The calculated value of the internal model deadtime for CV2 - PV2 after tuning is completed.	
CV3PV2DTTuned	REAL	The calculated value of the internal model deadtime for CV3 - PV2 after tuning is completed.	
CV1PV1RespTCTunedS	REAL	The calculated value of the control variable time constant in slow response speed for CV1 - PV1 after tuning is completed.	
CV2PV1RespTCTunedS	REAL	The calculated value of the control variable time constant in slow response speed for CV2 - PV1 after tuning is completed.	
CV3PV1RespTCTunedS	REAL	The calculated value of the control variable time constant in slow response speed for CV3 - PV1 after tuning is completed.	
CV1PV2RespTCTunedS	REAL	The calculated value of the control variable time constant in slow response speed for CV1 - PV2 after tuning is completed.	

Output Parameters	Data Type	Description	Values
CV2PV2RespTCTunedS	REAL	The calculated value of the control variable time constant in slow response speed for CV2 - PV2 after tuning is completed.	
CV3PV2RespTCTunedS	REAL	The calculated value of the control variable time constant in slow response speed for CV3 - PV2 after tuning is completed.	
CV1PV1RespTCTunedM	REAL	The calculated value of the control variable time constant in medium response speed for CV1 - PV1 after tuning is completed.	
CV2PV1RespTCTunedM	REAL	The calculated value of the control variable time constant in medium response speed for CV2 - PV1 after tuning is completed.	
CV3PV1RespTCTunedM	REAL	The calculated value of the control variable time constant in medium response speed for CV3 - PV1 after tuning is completed.	
CV1PV2RespTCTunedM	REAL	The calculated value of the control variable time constant in medium response speed for CV1 - PV2 after tuning is completed.	
CV2PV2RespTCTunedM	REAL	The calculated value of the control variable time constant in medium response speed for CV2 - PV2 after tuning is completed.	
CV3PV2RespTCTunedM	REAL	The calculated value of the control variable time constant in medium response speed for CV3 - PV2 after tuning is completed.	
CV1PV1RespTCTunedF	REAL	The calculated value of the control variable time constant in fast response speed for CV1 - PV1 after tuning is completed.	
CV2PV1RespTCTunedF	REAL	The calculated value of the control variable time constant in fast response speed for CV2 - PV1 after tuning is completed.	
CV3PV1RespTCTunedF	REAL	The calculated value of the control variable time constant in fast response speed for CV3 - PV1 after tuning is completed.	
CV1PV2RespTCTunedF	REAL	The calculated value of the control variable time constant in fast response speed for CV1 - PV2 after tuning is completed.	
CV2PV2RespTCTunedF	REAL	The calculated value of the control variable time constant in fast response speed for CV2 - PV2 after tuning is completed.	
CV3PV2RespTCTunedF	REAL	The calculated value of the control variable time constant in fast response speed for CV3 - PV2 after tuning is completed.	
AtuneCV1PV1On	BOOL	Set True when auto tuning for CV1 - PV1 has been initiated.	
AtuneCV2PV1On	BOOL	Set True when auto tuning for CV2 - PV1 has been initiated.	
AtuneCV3PV1On	BOOL	Set True when auto tuning for CV3 - PV1 has been initiated.	
AtuneCV1PV1Done	BOOL	Set True when auto tuning for CV1 - PV1 has completed successfully.	
AtuneCV2PV1Done	BOOL	Set True when auto tuning for CV2 - PV1 has completed successfully.	

Output Parameters	Data Type	Description	Values
AtuneCV3PV1Done	BOOL	Set True when auto tuning for CV3 - PV1 has completed successfully.	
AtuneCV1PV1Aborted	BOOL	Set True when auto tuning for CV1 - PV1 has been aborted by user or due to errors that occurred during the auto tuning operation.	
AtuneCV2PV1Aborted	BOOL	Set True when auto tuning for CV2 - PV1 has been aborted by user or due to errors that occurred during the auto tuning operation.	
AtuneCV3PV1Aborted	BOOL	Set True when auto tuning for CV3 PV1 has been aborted by user or due to errors that occurred during the auto tuning operation.	
AtuneCV1PV2On	BOOL	Set True with auto tuning for CV1 - PV2 has be initiated.	
AtuneCV2PV2On	BOOL	Set True with auto tuning for CV2 - PV2 has be initiated.	
AtuneCV3PV2On	BOOL	Set True with auto tuning for CV3 - PV2 has be initiated.	
AtuneCV1PV2Done	BOOL	Set True when auto tuning for CV1 - PV2 has completed successfully.	
AtuneCV2PV2Done	BOOL	Set True when auto tuning for CV2 - PV2 has completed successfully.	
AtuneCV3PV2Done	BOOL	Set True when auto tuning for CV3 - PV2 has completed successfully.	
ATuneCV1PV2Aborted	BOOL	Set True when auto tuning for CV1-PV2 has been aborted by user or due to errors that occurred during the auto tuning operation.	
ATuneCV2PV2Aborted	BOOL	Set True when auto tuning for CV2-PV2 has been aborted by user or due to errors that occurred during the auto tuning operation.	
ATuneCV3PV2Aborted	BOOL	Set True when auto tuning for CV3-PV2 has been aborted by user or due to errors that occurred during the auto tuning operation.	
AtuneCV1PV1Status	DINT	Bit mapped status for CV1 - PV1. A value of 0 indicates that no faults have occurred.	
AtuneCV2PV1Status	DINT	Bit mapped status for CV2 - PV1. A value of 0 indicates that no faults have occurred.	
AtuneCV3PV1Status	DINT	Bit mapped status for CV3 - PV1. A value of 0 indicates that no faults have occurred.	
AtuneCV1PV2Status	DINT	Bit mapped status for CV1 - PV2. A value of 0 indicates that no faults have occurred.	
AtuneCV2PV2Status	DINT	Bit mapped status for CV2 - PV2. A value of 0 indicates that no faults have occurred.	
AtuneCV3PV2Status	DINT	Bit mapped status for CV3 - PV2. A value of 0 indicates that no faults have occurred.	
AtuneCV1PV1Fault	BOOL	CV1 - PV1 Autotune has generated any of the following faults.	Bit 0 of AtuneCV1PV1Status
AtuneCV2PV1Fault	BOOL	CV2 - PV1 Autotune has generated any of the following faults.	Bit 0 of AtuneCV2PV1Status
AtuneCV3PV1Fault	BOOL	CV3 - PV1 Autotune has generated any of the following faults.	Bit 0 of AtuneCV3PV1Status

Output Parameters	Data Type	Description	Values
AtuneCV1PV1OutOfLimit	BOOL	Either PV1 or the deadtime-step ahead prediction of PV1 exceeds PV1TuneLimit during CV1 - PV1 Autotuning. When True, CV1 - PV1 Autotuning is aborted.	Bit 1 of AtuneCV1PV1Status
AtuneCV2PV1OutOfLimit	BOOL	Either PV1 or the deadtime-step ahead prediction of PV1 exceeds PV1TuneLimit during CV2 - PV1 Autotuning. When True, CV2 - PV1 Autotuning is aborted.	Bit 1 of AtuneCV2PV1Status
AtuneCV3PV1OutOfLimit	BOOL	Either PV1 or the deadtime-step ahead prediction of PV1 exceeds PV1TuneLimit during CV3 - PV1 Autotuning. When True, CV3 - PV1 Autotuning is aborted.	Bit 1 of AtuneCV3PV1Status
AtuneCV1PV1ModeInv	BOOL	The MMC mode was not Manual at start of Autotuning or the MMC mode was changed from Manual during CV1 - PV1 Autotuning. When True, CV1 - PV1 Autotuning is not started or is aborted.	Bit 2 of AtuneCV1PV1Status
AtuneCV2PV1ModeInv	BOOL	The MMC mode was not Manual at start of Autotuning or the MMC mode was changed from Manual during CV2 - PV1 Autotuning. When True, CV2 - PV1 Autotuning is not started or is aborted.	Bit 2 of AtuneCV2PV1Status
AtuneCV3PV1ModeInv	BOOL	The MMC mode was not Manual at start of Autotuning or the MMC mode was changed from Manual during CV3 - PV1 Autotuning. When True, CV3 - PV1 Autotuning is not started or is aborted.	Bit 2 of AtuneCV3PV1Status
AtuneCV1PV1WindupFault	BOOL	CV1WindupHIn or CV1WindupLIn is True at start of CV1 - PV1 Autotuning or during CV1 - PV1 Autotuning. When True, CV1 - PV1 Autotuning is not started or is aborted.	Bit 3 of AtuneCV1PV1Status
AtuneCV2PV1WindupFault	BOOL	CV2WindupHIn or CV2WindupLIn is True at start of CV2 - PV1 Autotuning or during CV2 - PV1 Autotuning. When True, CV2 - PV1 Autotuning is not started or is aborted.	Bit 3 of AtuneCV2PV1Status
AtuneCV3PV1WindupFault	BOOL	CV3WindupHIn or CV3WindupLIn is True at start of CV3 - PV1 Autotuning or during CV3 - PV1 Autotuning. When True, CV3 - PV1 Autotuning is not started or is aborted.	Bit 3 of AtuneCV3PV1Status
AtuneCV1PV1StepSize0	BOOL	CV1StepSizeUsed = 0 at start of CV1 - PV1 Autotuning. When True, CV1 - PV1 Autotuning is not started.	Bit 4 of AtuneCV1PV1Status
AtuneCV2PV1StepSize0	BOOL	CV2StepSizeUsed = 0 at start of CV2 - PV1 Autotuning. When True, CV2 - PV1 Autotuning is not started.	Bit 4 of AtuneCV2PV1Status
AtuneCV3PV1StepSize0	BOOL	CV3StepSizeUsed = 0 at start of CV3 - PV1 Autotuning. When True, CV3 - PV1 Autotuning is not started.	Bit 4 of AtuneCV3PV1Status
AtuneCV1PV1LimitsFault	BOOL	CV1LimitsInv and CVManLimiting are True at start of CV1 - PV1 Autotuning or during CV1 - PV1 Autotuning. When True, CV1 - PV1 Autotuning is not started or is aborted.	Bit 5 of AtuneCV1PV1Status

Output Parameters	Data Type	Description	Values
AtuneCV2PV1LimitsFault	BOOL	CV2LimitsInv and CVManLimiting are True at start of CV2 - PV1 Autotuning or during CV2 - PV1 Autotuning. When True, CV2 _PV1 Autotuning is not started or is aborted.	Bit 5 of AtuneCV2PV1Status
AtuneCV3PV1LimitsFault	BOOL	CV3LimitsInv and CVManLimiting are True at start of CV3 - PV1 Autotuning or during CV3 - PV1 Autotuning. When True, CV3 - PV1 Autotuning is not started or is aborted.	Bit 5 of AtuneCV3PV1Status
AtuneCV1PV1InitFault	BOOL	CV1Initializing is True at start of CV1 - PV1 Autotuning or during CV1 - PV1 Autotuning. When True, CV1 - PV1 Autotuning is not started or is aborted.	Bit 6 of AtuneCV1PV1Status
AtuneCV2PV1InitFault	BOOL	CV2Initializing is True at start of CV2 - PV1 Autotuning or during CV2 - PV1 Autotuning. When True, CV2 - PV1 Autotuning is not started or is aborted.	Bit 6 of AtuneCV2PV1Status
AtuneCV3PV1InitFault	BOOL	CV3Initializing is True at start of CV3 - PV1 Autotuning or during CV3 - PV1 Autotuning. When True, CV3 - PV1 Autotuning is not started or is aborted.	Bit 6 of AtuneCV3PV1Status
AtuneCV1PV1EUSpanChanged	BOOL	CV1EUSpan or PV1EUSpan changes during CV1 - PV1 Autotuning. When True, CV1 - PV1 Autotuning is aborted.	Bit 7 of AtuneCV1PV1Status
AtuneCV2PV1EUSpanChanged	BOOL	CV2EUSpan or PV1EUSpan changes during CV2 - PV1 Autotuning. When True, CV2 - PV1 Autotuning is aborted.	Bit 7 of AtuneCV2PV1Status
AtuneCV3PV1EUSpanChanged	BOOL	CV3EUSpan or PV1EUSpan changes during CV3 - PV1 Autotuning. When True, CV3 - PV1 Autotuning is aborted.	Bit 7 of AtuneCV3PV1Status
AtuneCV1PV1Changed	BOOL	CV1Oper is changed when in Operator control or CV1Prog is changed when in Program control or CV1 becomes high/low or ROC limited during CV1 - PV1 Autotuning. When True, CV1 - PV1 Autotuning is aborted.	Bit 8 of AtuneCV1PV1Status
AtuneCV2PV1Changed	BOOL	CV2Oper is changed when in Operator control or CV2Prog is changed when in Program control or CV2 becomes high/low or ROC limited during CV2 - PV1 Autotuning. When True, CV2 - PV1 Autotuning is aborted.	Bit 8 of AtuneCV2PV1Status
AtuneCV3PV1Changed	BOOL	CV3Oper is changed when in Operator control or CV3Prog is changed when in Program control or CV3 becomes high/low or ROC limited during CV3 - PV1 Autotuning. When True, CV3 - PV1 Autotuning is aborted.	Bit 8 of AtuneCV3PV1Status
AtuneCV1PV1Timeout	BOOL	Elapsed time is greater then PV1AtuneTimeLimit since step test is started. When True, CV1 - PV1 Autotuning is aborted.	Bit 9 of AtuneCV1PV1Status
AtuneCV2PV1Timeout	BOOL	Elapsed time is greater then PV1AtuneTimeLimit since step test is started. When True, CV2 - PV1 Autotuning is aborted.	Bit 9 of AtuneCV2PV1Status
AtuneCV3PV1Timeout	BOOL	Elapsed time is greater then PV1AtuneTimeLimit since step test is started. When True, CV3 - PV1 Autotuning is aborted.	Bit 9 of AtuneCV3PV1Status

Output Parameters	Data Type	Description	Values
AtuneCV1PV1NotSettled	BOOL	The PV1 is changed too much to Autotune for CV1 - PV1. When True, CV1 - PV1 Autotuning is aborted. Wait until PV1 is more stable before autotuning CV1 - PV1.	Bit 10 of AtuneCV1PV1Status
AtuneCV2PV1NotSettled	BOOL	The PV1 is changed too much to Autotune for CV2 - PV1. When True, CV2 - PV1 Autotuning is aborted. Wait until PV1 is more stable before autotuning CV2 - PV1.	Bit 10 of AtuneCV2PV1Status
AtuneCV3PV1NotSettled	BOOL	The PV1 is changed too much to Autotune for CV3 - PV1. When True, CV3 - PV1 Autotuning is aborted. Wait until PV1 is more stable before autotuning CV3 - PV1.	Bit 10 of AtuneCV3PV1Status
AtuneCV1PV2Fault	BOOL	CV1 - PV2 Autotune has generated any of the following faults.	Bit 0 of AtuneCV1PV2Status
AtuneCV2PV2Fault	BOOL	CV2 - PV2 Autotune has generated any of the following faults.	Bit 0 of AtuneCV2PV2Status
AtuneCV3PV2Fault	BOOL	CV3 - PV2 Autotune has generated any of the following faults.	Bit 0 of AtuneCV3PV2Status
AtuneCV1PV2OutOfLimit	BOOL	Either PV2 or the deadtime-step ahead prediction of PV2 exceeds PV2TuneLimit during CV1 - PV2 Autotuning. When True, CV1 - PV2 Autotuning is aborted.	Bit 1 of AtuneCV1PV2Status
AtuneCV2PV2OutOfLimit	BOOL	Either PV2 or the deadtime-step ahead prediction of PV2 exceeds PV2TuneLimit during CV2 - PV2 Autotuning. When True, CV2 - PV2 Autotuning is aborted.	Bit 1 of AtuneCV2PV2Status
AtuneCV3PV2OutOfLimit	BOOL	Either PV2 or the deadtime-step ahead prediction of PV2 exceeds PV2TuneLimit during CV3 - PV2 Autotuning. When True, CV3 - PV2 Autotuning is aborted.	Bit 1 of AtuneCV3PV2Status
AtuneCV1PV2ModeInv	BOOL	The MMC mode was not Manual at start of Autotuning or the MMC mode was changed from Manual during CV1-PV2 Autotuning. When True, CV1-PV2 Autotuning is not started or is aborted.	Bit 2 of AtuneCV1PV2Status
AtuneCV2PV2ModeInv	BOOL	The MMC mode was not Manual at start of Autotuning or the MMC mode was changed from Manual during CV2-PV2 Autotuning. When True, CV2-PV2 Autotuning is not started or is aborted.	Bit 2 of AtuneCV2PV2Status
AtuneCV3PV2ModeInv	BOOL	The MMC mode was not Manual at start of Autotuning or the MMC mode was changed from Manual during CV3-PV2 Autotuning. When True, CV3-PV2 Autotuning is not started or is aborted.	Bit 2 of AtuneCV3PV2Status
AtuneCV1PV2WindupFault	BOOL	CV1WindupHIn or CV1WindupLIn is True at start of CV1 - PV2 Autotuning or during CV1 - PV2 Autotuning. When True, CV1 - PV2 Autotuning is not started or is aborted.	Bit 3 of AtuneCV1PV2Status
AtuneCV2PV2WindupFault	BOOL	CV2WindupHIn or CV2WindupLIn is True at start of CV2 - PV2 Autotuning or during CV2 - PV2 Autotuning. When True, CV2 - PV2 Autotuning is not started or is aborted.	Bit 3 of AtuneCV2PV2Status

Output Parameters	Data Type	Description	Values
AtuneCV3PV2WindupFault	BOOL	CV3WindupHIn or CV3WindupLIn is True at start of CV3 - PV2 Autotuning or during CV3 - PV2 Autotuning. When True, CV3 - PV2 Autotuning is not started or is aborted.	Bit 3 of AtuneCV3PV2Status
AtuneCV1PV2StepSize0	BOOL	CV1StepSizeUsed = 0 at start of CV1 - PV2 Autotuning. When True, CV1 - PV2 Autotuning is not started.	Bit 4 of AtuneCV1PV2Status
AtuneCV2PV2StepSize0	BOOL	CV2StepSizeUsed = 0 at start of CV2 - PV2 Autotuning. When True, CV2 - PV2 Autotuning is not started.	Bit 4 of AtuneCV2PV2Status
AtuneCV3PV2StepSize0	BOOL	CV3StepSizeUsed = 0 at start of CV3 - PV2 Autotuning. When True, CV3 - PV2 Autotuning is not started.	Bit 4 of AtuneC3PV2Status
AtuneCV1PV2LimitsFault	BOOL	CV1LimitsInv and CVManLimiting are True at start of CV1 - PV2 Autotuning or during CV1 - PV2 Autotuning. When True, CV1 - PV2 Autotuning is not started or is aborted.	Bit 5 of AtuneCV1PV2Status
AtuneCV2PV2LimitsFault	BOOL	CV2LimitsInv and CVManLimiting are True at start of CV2 - PV2 Autotuning or during CV2 - PV2 Autotuning. When True, CV2 - PV2 Autotuning is not started or is aborted.	Bit 5 of AtuneCV2PV2Status
AtuneCV3PV2LimitsFault	BOOL	CV3LimitsInv and CVManLimiting are True at start of CV3 - PV2 Autotuning or during CV3 - PV2 Autotuning. When True, CV3 - PV2 Autotuning is not started or is aborted.	Bit 5 of AtuneCV3PV2Status
AtuneCV1PV2InitFault	BOOL	CV1Initializing is True at start of CV1 - PV2 Autotuning or during CV1 - PV2 Autotuning. When True, CV1 - PV2 Autotuning is not started or is aborted.	Bit 6 of AtuneCV1PV2Status
AtuneCV2PV2InitFault	BOOL	CV2Initializing is True at start of CV2 - PV2 Autotuning or during CV2 - PV2 Autotuning. When True, CV2 - PV2 Autotuning is not started or is aborted.	Bit 6 of AtuneCV2PV2Status
AtuneCV3PV2InitFault	BOOL	CV3Initializing is True at start of CV3 - PV2 Autotuning or during CV3 - PV2 Autotuning. When True, CV3 - PV2 Autotuning is not started or is aborted.	Bit 6 of AtuneCV3PV2Status
AtuneCV1PV2EUSpanChanged	BOOL	CV1EUSpan or PV2EUSpan changes during CV1 - PV2 Autotuning. When True, CV1 - PV2 Autotuning is aborted.	Bit 7 of AtuneCV1PV2Status
AtuneCV2PV2EUSpanChanged	BOOL	CV2EUSpan or PV2EUSpan changes during CV2 - PV2 Autotuning. When True, CV2 - PV2 Autotuning is aborted.	Bit 7 of AtuneCV2PV2Status
AtuneCV3PV2EUSpanChanged	BOOL	CV3EUSpan or PV2EUSpan changes during CV2 - PV3 Autotuning. When True, CV3 - PV2 Autotuning is aborted.	Bit 7 of AtuneCV3PV2Status
AtuneCV1PV2Changed	BOOL	CV1Oper is changed when in Operator control or CV1Prog is changed when in Program control or CV1 becomes high/low or ROC limited during CV1 - PV2 Autotuning. When True, CV1 - PV2 Autotuning is aborted.	Bit 8 of AtuneCV1PV2Status

Output Parameters	Data Type	Description	Values
AtuneCV2PV2Changed	BOOL	CV2Oper is changed when in Operator control or CV2Prog is changed when in Program control or CV2 becomes high/low or ROC limited during CV2 - PV2 Autotuning. When True, CV2 - PV2 Autotuning is aborted.	Bit 8 of AtuneCV2PV2Status
AtuneCV3PV2Changed	BOOL	CV3Oper is changed when in Operator control or CV3Prog is changed when in Program control or CV3 becomes high/low or ROC limited during CV3 - PV2 Autotuning. When True, CV3 - PV2 Autotuning is aborted.	Bit 8 of AtuneCV3PV2Status
AtuneCV1PV2Timeout	BOOL	Elapsed time is greater then PV2AtuneTimeLimit since step test is started. When True, CV1 - PV2 Autotuning is aborted.	Bit 9 of AtuneCV1PV2Status
AtuneCV2PV2Timeout	BOOL	Elapsed time is greater then PV2AtuneTimeLimit since step test is started. When True, CV2 - PV2 Autotuning is aborted.	Bit 9 of AtuneCV2PV2Status
AtuneCV3PV2Timeout	BOOL	Elapsed time is greater then PV2AtuneTimeLimit since step test is started. When True, CV3 - PV2 Autotuning is aborted.	Bit 9 of AtuneCV3PV2Status
AtuneCV1PV2NotSettled	BOOL	The PV2 is changed too much to Autotune for CV1-PV2. When True, CV1-PV2 Autotuning is aborted. Wait until PV2 is more stable before autotuning CV1-PV2.	Bit 10 of AtuneCV1PV2Status
AtuneCV2PV2NotSettled	BOOL	The PV2 is changed too much to Autotune for CV2-PV2. When True, CV2-PV2 Autotuning is aborted. Wait until PV2 is more stable before autotuning CV2-PV2.	Bit 10 of AtuneCV2PV2Status
AtuneCV3PV2NotSettled	BOOL	The PV2 is changed too much to Autotune for CV3-PV2. When True, CV3-PV2 Autotuning is aborted. Wait until PV2 is more stable before autotuning CV3-PV2.	Bit 10 of AtuneCV3PV2Status
Status1	DINT	Bit mapped status of the function block. A value of 0 indicates that no faults have occurred. Any parameter that could be configured with an invalid value must have a status parameter bit to indicate its invalid status.	
Status2	DINT	Additional bit mapped status for the function block. A value of 0 indicates that no faults have occurred. Any parameter that could be configured with an invalid value must have a status parameter bit to indicate its invalid status.	
Status3CV1	DINT	Additional bit mapped CV1 status for the function block. A value of 0 indicates that no faults have occurred.	
Status3CV2	DINT	Additional bit mapped CV2 status for the function block. A value of 0 indicates that no faults have occurred.	
Status3CV3	DINT	Additional bit mapped CV3 status for the function block. A value of 0 indicates that no faults have occurred.	

Output Parameters	Data Type	Description	Values
InstructFault	BOOL	The function block has generated a fault. Indicates state of bits in Status1, Status2, and Status3CV(n), where (n) can be 1, 2, or 3.	Bit 0 of Status1
PV1Faulted	BOOL	Process variable PV1 health bad.	Bit 1 of Status1
PV2Faulted	BOOL	Process variable PV2 health bad.	Bit 2 of Status1
PV1SpanInv	BOOL	The span of PV1 inValid, PV1EUMax < PV1EUMin.	Bit 3 of Status1
PV2SpanInv	BOOL	The span of PV2 inValid, PV2EUMax < PV2EUMin.	Bit 4 of Status1
SP1ProgInv	BOOL	SP1Prog < SP1LLimit or > SP1HLimit. Limit value used for SP1.	Bit 5 of Status1
SP2ProgInv	BOOL	SP2Prog < SP2LLimit or > SP2HLimit. Limit value used for SP2.	Bit 6 of Status1
SP1OperInv	BOOL	SP1Oper < SP1LLimit or > SP1HLimit. Limit value used for SP1.	Bit 7 of Status1
SP2OperInv	BOOL	SP2Oper < SP2LLimit or > SP2HLimit. Limit value used for SP2.	Bit 8 of Status1
SP1LimitsInv	BOOL	Limits inValid: SP1LLimit < PV1EUMin, SP1HLimit > PV1EUMax, or SP1HLimit < SP1LLimit. If SP1HLimit < SP1LLimit, then limit value by using SP1LLimit.	Bit 9 of Status1
SP2LimitsInv	BOOL	Limits inValid: SP2LLimit < PV2EUMin, SP2HLimit > PV2EUMax, or SP2HLimit < SP2LLimit. If SP2HLimit < SP2LLimit, then limit value by using SP2LLimit.	Bit 10 of Status1
SampleTimeTooSmall	BOOL	Model DeadTime / DeltaT must be less than or equal to 200.	Bit 11 of Status1
PV1FactorInv	BOOL	Entered value for Factor1 < 0.	Bit 12 of Status1
PV2FactorInv	BOOL	Entered value for Factor2 < 0.	Bit 13 of Status1
TimingModeInv	BOOL	Entered TimingMode inValid. If the current mode is not Override or Hand then set to Manual mode.	Bit 27 of Status2
RTSMissed	BOOL	Only used when in Real Time Sampling mode. Is TRUE when ABS(DeltaT – RTSTime) > 1 millisecond.	Bit 28 of Status2.
RTSTimeInv	BOOL	Entered RTSTime inValid.	Bit 29 of Status2.
RTSTimeStampInv	BOOL	RTSTimeStamp inValid. If the current mode is not Override or Hand, then set to Manual mode.	Bit 30 of Status2.
DeltaTInv	BOOL	DeltaT inValid. If the current mode is not Override or Hand then set to Manual mode.	Bit 31 of Status2.
CV1Faulted	BOOL	Control variable CV1 health bad.	Bit 0 of Status3CV1
CV2Faulted	BOOL	Control variable CV2 health bad.	Bit 0 of Status3CV2
CV3Faulted	BOOL	Control variable CV3 health bad.	Bit 0 of Status3CV3
CV1HandFBFaulted	BOOL	CV1 HandFB value health bad.	Bit 1 of Status3CV1
CV2HandFBFaulted	BOOL	CV2 HandFB value health bad.	Bit 1 of Status3CV2
CV3HandFBFaulted	BOOL	CV3 HandFB value health bad.	Bit 1 of Status3CV3

Output Parameters	Data Type	Description	Values
CV1ProgInv	BOOL	CV1Prog < 0 or > 100, or < CV1LLimit or > CV1HLimit when CVManLimiting is TRUE. Limit value used for CV1.	Bit 2 of Status3CV1
CV2ProgInv	BOOL	CV2Prog < 0 or > 100, or < CV2LLimit or > CV2HLimit when CVManLimiting is TRUE. Limit value used for CV2.	Bit 2 of Status3CV2
CV3ProgInv	BOOL	CV3Prog < 0 or > 100, or < CV3LLimit or > CV3HLimit when CVManLimiting is TRUE. Limit value used for CV3.	Bit 2 of Status3CV3
CV1OperInv	BOOL	CV1Oper < 0 or > 100, or < CV1LLimit or > CV1HLimit when CVManLimiting is TRUE. Limit value used for CV1.	Bit 3 of Status3CV1
CV2OperInv	BOOL	CV2Oper < 0 or > 100, or < CV2LLimit or > CV2HLimit when CVManLimiting is TRUE. Limit value used for CV2.	Bit 3 of Status3CV2
CV3OperInv	BOOL	CV3Oper < 0 or > 100, or < CV3LLimit or > CV3HLimit when CVManLimiting is TRUE. Limit value used for CV3.	Bit 3 of Status3CV3
CV1OverrideValueInv	BOOL	CV1OverrideValue < 0 or > 100. Limit value used for CV1.	Bit 4 of Status3CV1
CV2OverrideValueInv	BOOL	CV2OverrideValue < 0 or > 100. Limit value used for CV2.	Bit 4 of Status3CV2
CV3OverrideValueInv	BOOL	CV3OverrideValue < 0 or > 100. Limit value used for CV3.	Bit 4 of Status3CV3
CV1EUSpanInv	BOOL	The span of CV1EU invalid, CV1EUMax equals CV1EUMin.	Bit 5 of Status3CV1
CV2EUSpanInv	BOOL	The span of CV2EU invalid, CV2EUMax equals CV2EUMin.	Bit 5 of Status3CV2
CV3EUSpanInv	BOOL	The span of CV3EU invalid, CV3EUMax equals CV3EUMin.	Bit 5 of Status3CV3
CV1LimitsInv	BOOL	CV1LLimit < 0, CV1HLimit > 100, or CV1HLimit <= CV1LLimit. If CV1HLimit <= CV1LLimit, limit CV1 by using CV1LLimit.	Bit 6 of Status3CV1
CV2LimitsInv	BOOL	CV2LLimit < 0, CV2HLimit > 100, or CV2HLimit <= CV2LLimit. If CV2HLimit <= CV2LLimit, limit CV2 by using CV2LLimit.	Bit 6 of Status3CV2
CV3LimitsInv	BOOL	CV3LLimit < 0, CV3HLimit > 100, or CV3HLimit <= CV3LLimit. If CV3HLimit <= CV3LLimit, limit CV3 by using CV3LLimit.	Bit 6 of Status3CV3
CV1ROCLimitInv	BOOL	Entered value for CV1ROCLimit < 0, disables CV1 ROC limiting.	Bit 7 of Status3CV1
CV2ROCLimitInv	BOOL	Entered value for CV2ROCLimit < 0, disables CV2 ROC limiting.	Bit 7 of Status3CV2
CV3ROCLimitInv	BOOL	Entered value for CV3ROCLimit < 0, disables CV3 ROC limiting.	Bit 7 of Status3CV3
CV1HandFBInv	BOOL	CV1 HandFB < 0 or > 100. Limit value used for CV1.	Bit 8 of Status3CV1
CV2HandFBInv	BOOL	CV2 HandFB < 0 or > 100. Limit value used for CV2.	Bit 8 of Status3CV2
CV3HandFBInv	BOOL	CV3 HandFB < 0 or > 100. Limit value used for CV3.	Bit 8 of Status3CV3

Output Parameters	Data Type	Description	Values
CV1PV1ModelGainInv	BOOL	CV1PV1ModelGain is 1.#QNAN or -1.#IND. (Not A Number), or ± 1.$ (Infinity ∞).	Bit 9 of Status3CV1
CV2PV1ModelGainInv	BOOL	Entered value for CV2 - PV1 Model Gain is 1.#QNAN or -1.#IND. (Not A Number), or ± 1.$ (Infinity ∞).	Bit 9 of Status3CV2
CV3PV1ModelGainInv	BOOL	Entered value for CV3 - PV1 Model Gain is 1.#QNAN or -1.#IND. (Not A Number), or ± 1.$ (Infinity ∞).	Bit 9 of Status3CV3
CV1PV2ModelGainInv	BOOL	CV1PV2ModelGain is 1.#QNAN or -1.#IND. (Not A Number), or ± 1.$ (Infinity ∞).	Bit 10 of Status3CV1
CV2PV2ModelGainInv	BOOL	Entered value for CV2 - PV2 Model Gain is 1.#QNAN or -1.#IND. (Not A Number), or ± 1.$ (Infinity ∞).	Bit 10 of Status3CV2
CV3PV2ModelGainInv	BOOL	Entered value for CV3 - PV2 Model Gain is 1.#QNAN or -1.#IND. (Not A Number), or ± 1.$ (Infinity ∞).	Bit 10 of Status3CV3
CV1PV1ModelTCInv	BOOL	Entered value for CV1 - PV1 Model Time Constant < 0.	Bit 11 of Status3CV1
CV2PV1ModelTCInv	BOOL	Entered value for CV2 - PV1 Model Time Constant < 0.	Bit 11 of Status3CV2
CV3PV1ModelTCInv	BOOL	Entered value for CV3 - PV1 Model Time Constant < 0.	Bit 11 of Status3CV3
CV1PV2ModelTCInv	BOOL	Entered value for CV1 - PV2 Model Time Constant < 0.	Bit 12 of Status3CV1
CV2PV2ModelTCInv	BOOL	Entered value for CV2 - PV2 Model Time Constant < 0.	Bit 12 of Status3CV2
CV3PV2ModelTCInv	BOOL	Entered value for CV3 - PV2 Model Time Constant < 0.	Bit 12 of Status3CV3
CV1PV1ModelDTInv	BOOL	Entered value for CV1-PV1 Model Deadtime < 0.	Bit 13 of Status3CV1
CV2PV1ModelDTInv	BOOL	Entered value for CV2-PV1 Model Deadtime < 0.	Bit 13 of Status3CV2
CV3PV1ModelDTInv	BOOL	Entered value for CV3 - PV1 Model Deadtime < 0.	Bit 13 of Status3CV3
CV1PV2ModelDTInv	BOOL	Entered value for CV1-PV2 Model Deadtime < 0.	Bit 14 of Status3CV1
CV2PV2ModelDTInv	BOOL	Entered value for CV2-PV2 Model Deadtime < 0.	Bit 14 of Status3CV2
CV3PV2ModelDTInv	BOOL	Entered value for CV3 - PV2 Model Deadtime < 0.	Bit 14 of Status3CV3
CV1PV1RespTCInv	BOOL	Entered value for CV1-PV1 Response Time Constant < 0.	Bit 15 of Status3CV1
CV2PV1RespTCInv	BOOL	Entered value for CV2-PV1 Response Time Constant < 0.	Bit 15 of Status3CV2
CV3PV1RespTCInv	BOOL	Entered value for CV3 - PV1 Response Time Constant < 0.	Bit 15 of Status3CV3
CV1PV2RespTCInv	BOOL	Entered value for CV1-PV2 Response Time Constant < 0.	Bit 16 of Status3CV1
CV2PV2RespTCInv	BOOL	Entered value for CV2-PV2 Response Time Constant < 0.	Bit 16 of Status3CV2

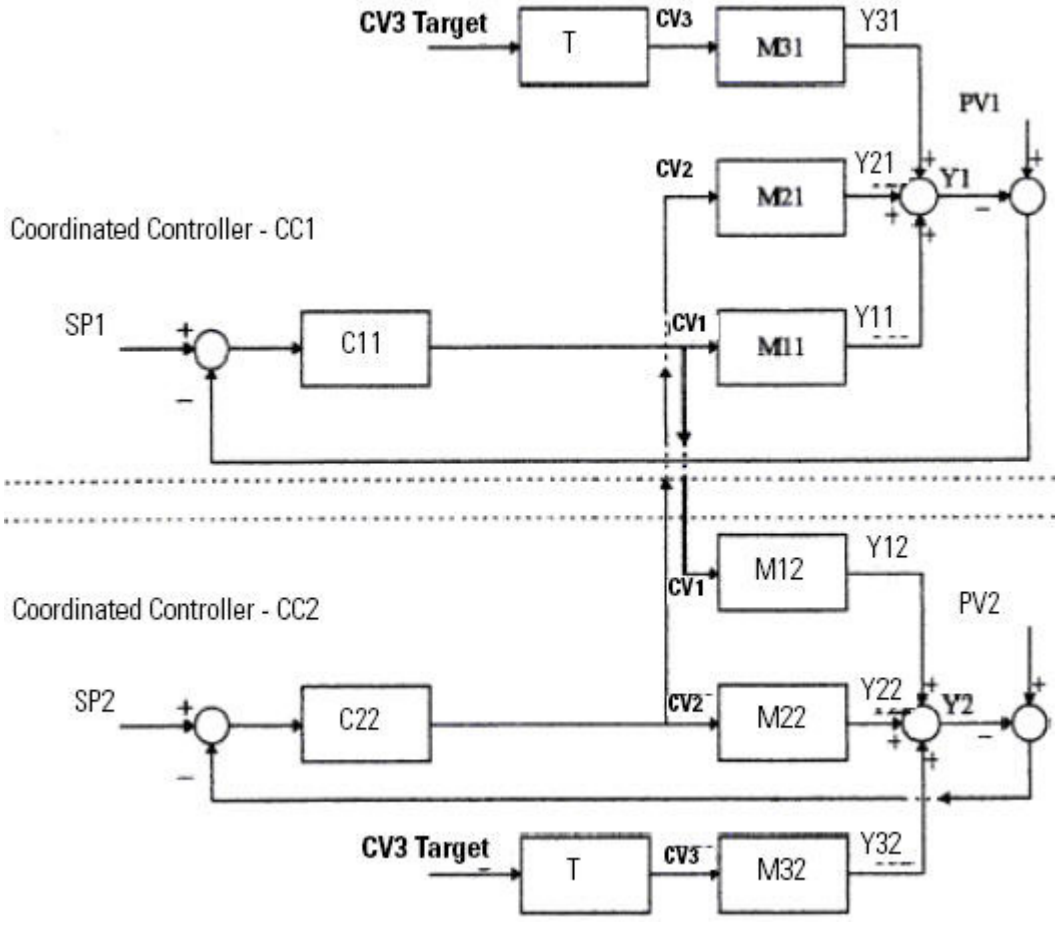
Output Parameters	Data Type	Description	Values
CV3PV2RespTCInv	BOOL	Entered value for CV3 - PV2 Response Time Constant < 0.	Bit 16 of Status3CV3
CV1TargetInv	BOOL	Entered value for CV1 Target < 0. or > 100.	Bit 17 of Status3CV1
CV2TargetInv	BOOL	Entered value for CV2 Target < 0. or > 100.	Bit 17 of Status3CV2
CV3TargetInv	BOOL	Entered value for CV3 Target < 0. or > 100.	Bit 17 of Status3CV3

Description

The MMC is a flexible model-based algorithm that can be used in two basic configuration modes:

- Three control variables used to control two interacting process variables
- Two control variables used to control two interacting process variables

Following is an MMC function block splitter example configuration.



Item	Description
M11	Internal model CV1 - PV1

M21	Internal model CV2 - PV1
M31	Internal model CV3 - PV1
M12	Internal model CV1 - PV2
M22	Internal model CV2 - PV2
M32	Internal model CV3 - PV2
T	Target response
C11, C22	Model-predictive function blocks (IMC) currently controlling PV1 and PV2 to SP1 and SP2, respectively
Y11, Y21, Y31, Y12, Y22, Y32	Model outputs of M11, M21, M31, M12, M22, M32
Y1	PV1 prediction
Y2	PV2 prediction
CV1 (Reflux ratio)	Controls PV1 (Top composition) in Coordinated Control (CC1).
CV2 (Stream Flow)	Controls PV2 (Bottom composition) in Coordinated Control (CC2)
CV3	Drives the Target value through a target response.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for Array-Indexing Faults

Execution

Note that iin Structured Text, EnableIn is always true during a normal scan. Therefore, if the instruction is in the control path activated by the logic, it will execute.

Refer to Refer to Function Block Attributes.for more details including definitions and general behavior for all Function Block instructions.

All conditions below the shaded area can only occur during Normal Scan mode.

Condition/State	Action Taken
Prescan	.EnableIn and .EnableOut bits are cleared to false.
Postscan	.EnableIn and .EnableOut bits are cleared to false.
EnableIn is false	.EnableIn and .EnableOut bits are cleared to false.
Instruction first run	No state specific action taken. Primary algorithm not executed, however will validate input parameters.
Instruction first scan	No state specific action taken. Primary algorithm not executed, however will validate input parameters.

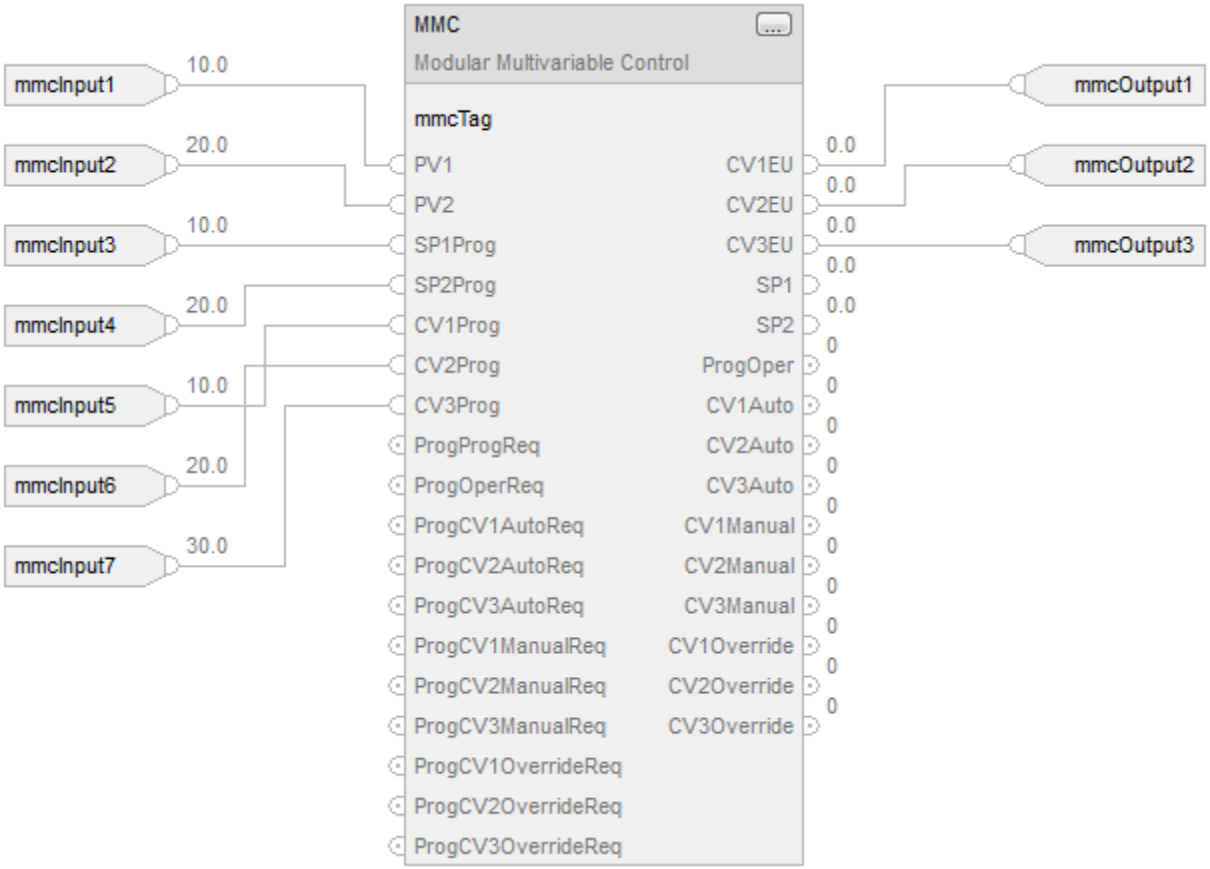
EnableIn is true	.EnableIn and .EnableOut bits are set to true. The instruction's main algorithm will be executed and outputs will be updated.

Native Implementation

Platform	Intrinsics/Main Function
ABRisc / ARM	ABRisc assembly code void FB_ModularMultivariableControl(UINT32 *pulArgOPtr)
RCA	MMC(instance) void FB_ModularMultivariableControl(UINT32 *pulArgOPtr)
SoftLogix (X86)	void rts$MMC(UINT32 *pFbdBlock)

Example

Function Block



Structured Text

```
mmcTag.PV1 := mmcInput1;
mmcTag.PV2 := mmcInput2;
mmcTag.SP1Prog := mmcInput3;
mmcTag.SP2Prog := mmcInput4;
mmcTag.CV1Prog := mmcInput5;
mmcTag.CV2Prog := mmcInput6;
mmcTag.CV3Prog := mmcInput7;
MMC(mmcTag);
mmcOutput1 := mmcTag.CV1EU;
mmcOutput2 := mmcTag.CV2EU;
mmcOutput3 := mmcTag.CV3EU;
```

See also

Convert the PV and SP Values to Percent on page 252

Instruction First Scan on page 250

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Attributes on page 1043

MMC Function Block Configuration

Starting with the default configuration, configure the following parameters.

Parameter	Description
PV1EUMax	Maximum scaled value for PV1.
PV1EUMin	Minimum scaled value for PV1.
PV2EUMax	Maximum scaled value for PV2.
PV2EUMin	Minimum scaled value for PV2.
SP1HLimit	SP1 high limit value, scaled in PV units.
SP1LLimit	SP1 low limit value, scaled in PV units.
SP2HLimit	SP2 high limit value, scaled in PV units.
SP2LLimit	SP2 low limit value, scaled in PV units.
CV1InitValue	An initial value of the control variable CV1 output.
CV2InitValue	An initial value of the control variable CV2 output.
CV3InitValue	An initial value of the control variable CV3 output.

If you have the process models available, you can intuitively tune the MMC function block by entering the following parameters. At this point, you have completed the basic configuration. You did not configure the built-in tuner.

The function block variables are ready to be put on-line in either auto or Manual mode. For tuning, the default settings will be used.

If you do not know the process models, you need to identify the models and tune the function block by using the built-in tuner (modeler) for the function block to operate correctly in the Auto mode.

Parameter	Description
ModelGains	Nonzero numbers (negative for direct acting control variable, positive for reverse acting control variable)
ModelTimeConstants	Always positive numbers
ModelDeadtimes	Always positive numbers
RespTimeConstants	Always positive numbers
Active 1st, 2nd and 3rd CV for PV1 and PV2	Specify the order in which CV's will be used to compensate for PV - SP error.
TargetCV	Specify which CV will be driven to its target value.
CVTargetValues	Specify to which values should the control variable drive the individual CV's if selected as the TargetCV.
TargetRespTC	Specify the speed of CV's to approach the target values.

For integrating process types (such as level control and position control), internal nonintegrating models are used to approximate the integrating process. The Factor parameters are used to convert the identified integrating process models to nonintegrating internal models used for CV calculation. This is necessary to provide for stable MMC execution. The MMC function block can handle any combinations of PV1 and PV2 that are integrating or nonintegrating process types.

The function block uses first order lag with deadtime internal process models and first order filters (total of up to 24 tuning parameters-6 models, 4 parameters each) to calculate the CV's. Each CV is calculated such that each process variable (PV) follows a first order lag trajectory when approaching the setpoint value.

Speed of response depends on the value of the response time constants. The smaller the response time constants, the faster the control variable response will be. The response time constants should be set such that the PV's reach the setpoints in reasonable time based on the process dynamics. The larger that the response time constants, the slower the control variable response will be, but the control variable also becomes more robust.

In the Manual mode, the control variables (CV) are set equal to the operator-entered Manual CV parameters. For the Manual to Auto mode bumpless transfer and for safe operation of the control variable, the CV rate of change limiters are implemented such that CV's cannot move from current states by more than specified CV units at each scan.

Set the CVROCPosLimit and CVROCNegLimit to limit the CV rate of change. Rate limiting is not imposed when control variable is in Manual mode unless CVManLimiting is set.

MMC Function Block Model Initialization

A model initialization occurs:

- During First Scan of the block
- When the ModelInit request parameter is set
- When DeltaT changes

You may need to manually adjust the internal model parameters or the response time constants. You can do so by changing the appropriate parameters and setting the appropriate ModelInit bit. The internal states of the function block will be initialized, and the bit will automatically reset.

For example, if you modify the model for CV2 - PV1 model, set the CV2PV1ModelInit parameter to TRUE to initialize the CV2 - PV1 internal model parameters and for the new model to take effect.

MMC Function Block Tuning

The MMC function block is equipped with an internal tuner (modeler). The purpose of the tuner is to identify the process model parameters and to use these parameters as internal model parameters (gain, time constant, and deadtime). The tuner also calculates an optimal response time constant.

Set the tuner by configuring the following parameters for each CV - PV process.

ProcessType	Integrating (level, position control) or nonintegrating (flow, pressure control)
ProcessGainSign	Set to indicate a negative process gain (increase in output causes a decrease in PV); reset to indicate a positive process gain (increase in output causes an increase in PV).
ResponseSpeed	Slow, medium, or fast, based on control objective
NoiseLevel	An estimate of noise level on PV-low, medium, or high-such that the tuner can distinguish which PV change is a random noise and which is caused by the CV step change
StepSize	A nonzero positive or negative number defining the magnitude of CV step change in either positive or negative direction, respectively
PVTuneLimit	(only for integrating process type) in PV engineering units, defines how much of PV change that is caused by CV change to tolerate before aborting the tuning test due to exceeding this limit

The tuner is started by setting the AtuneStart bit (AtuneCV1Start, for example). You can stop the tuning by setting the appropriate AtuneAbort bit.

After the tuning is completed successfully, the appropriate GainTuned, TCTuned, DTTuned, and RespTCTuned parameters are updated with the tuning results, and the AtuneStatus code is set to indicate complete.

You can copy these parameters to the ModelGain, ModelTC, ModelDT, and RespTC, respectively, by setting the AtuneUseModel bit. The MMC function block automatically initializes the internal variables and continue normal operation. It automatically resets the AtuneUseModel bit.

See also

MMC Function Block Tuning Procedure on page 244

MMC Function Block Tuning Errors on page 244

Use an MMC Function Block for Splitter Control

The following example describes using an MMC function block to control a splitter. Refer to the MMC Function Block Splitter Example Configuration in Module Multivariable Control (MMC).

Item	Description
PV1	Top composition (more important)
PV2	Bottom composition (less important)
Active 1st CV for PV1	CV1 (reflux ratio)
Active 2nd CV for PV1	CV3 (pressure setpoint)
Active 3rd CV for PV1	CV2 (steam (flow)
Active 1st CV for PV2	CV2
Active 2nd CV for PV2	CV3
Active 3rd CV for PV2	CV1
TargetCV	CV3 (pressure should be held constant if possible)
CV3Target	60% (of pressure range)

The MMC calculates CV1, CV2, and CV3 so that the control goals are accomplished in the following order of importance:

1. Control PV1 to SP1 (PV1 is always considered more important than PV2)
2. Control PV2 to SP2
3. Control CV3 to its target value

CV1 is selected as the most active control for PV1 and CV2 as the most active for PV2. If either CV1 or CV2 saturates or is put in Manual mode, the control variable will use CV3 to maintain PV1 and PV2 at the setpoints.

See also

Module Multivariable Control (MMC) on page 203

MMC Function Block Tuning Errors

If an error occurs during the tuning procedure, the tuning is aborted, and an appropriate AtuneStatus bit is set. Also, a user can abort the tuning by setting the AtuneAbort parameter.

After an abort, the CV assumes its value before the step change, and the GainTuned, TCTuned, DTTuned, and RespTCTuned parameters are not updated. The AtuneStatus parameter identifies the reason for the abort.

MMC Function Block Tuning Procedure

Follow these steps to configure the tuner.

1. Put all three CV parameters into Manual mode.
2. Set the appropriate AtuneStart parameter.

   The tuner starts collecting PV and CV data for noise calculation.
3. After collecting 60 samples (60*DeltaT) period, the tuner adds StepSize to the CV.

After successfully collecting the PV data as a result of the CV step change, the CV assumes its value before the step change and the AtuneStatus, GainTuned, TCTuned, DTTuned, and RespTCTuned parameters are updated.

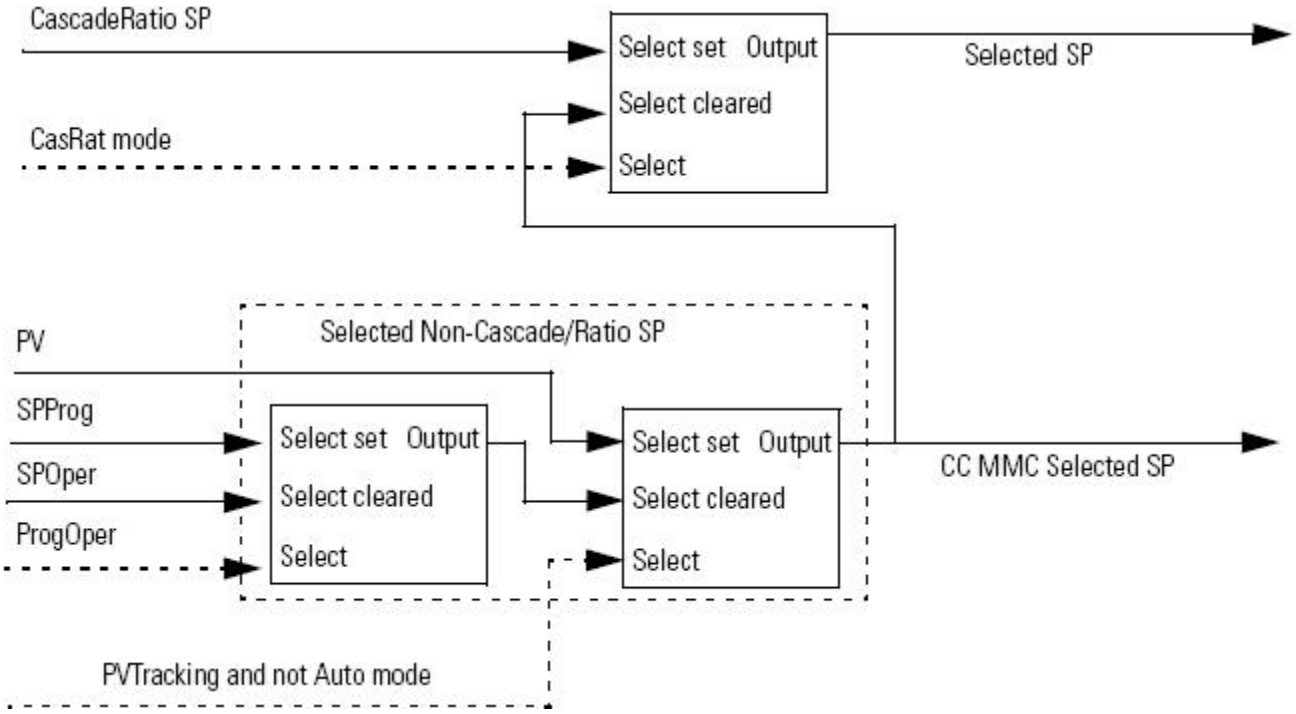
4. Set the appropriate AtuneUseModel parameter to copy the tuned parameters to the model parameters

   The function block then resets the AtuneUseModel parameter. After a successful AutoTuneDone, the Atune parameter is set to one (1). Tuning completed successfully.

To identify models and to calculate response time constants for all six CV - PV processes, run the tuner up to three times to obtain CV1 - PV2, CV2 - PV2, and CV3 - PV2 models and tuning, respectively. After each run, two process models are identified: CV - PV1 and CV - PV2 (two process variables respond as a result of one CV step change)

Current SP

The current SP is based on the Cascade/Ratio mode, the PVTracking value, auto mode, and the ProgOper value.



Use the Coordinated Control Function Block to Control

This is an example of how you could use the Coordinated Control function block to control the temperature in a process.

Name	Description
PV	Temperature
Act1stCV	CV3 (high pressure steam)
Act2ndCV	CV2 (cooling)
Act3rdCV	CV1 (low pressure steam)
Target1stCV	CV2
Target2ndCV	CV3

Target3rdCV	CV1
CV1Target	0% This value is irrelevant since in the target list setup, CV1 has the lowest priority, and will assume the steady state load to maintain PV at the setpoint.
CV2Target	0%
CV3Target	10%

Temperature Example Explanation

Manipulating the PV at the setpoint is the top priority. The high pressure steam and cooling are selected as the most active actuators. At steady state, the same two controls should assume their target values: CV3 at 10% and CV2 at 0%. CV1 will assume any value needed to maintain PV at the setpoint; therefore, its target value is irrelevant since manipulating the PV at the setpoint is a higher priority control objective. Target CV priorities and target CV values can be changed on-line.

The CC function block calculates CV1, CV2 and CV3 such that the control goals are accomplished in the following order of importance:

1. Control PV to SP
2. Control CV2 to its target value
3. Control CV3 to its target value

At this point, you have completed the basic configuration. You did not configure the built-in tuner. The control variable is ready to be put on-line in either auto or Manual mode. For tuning, the default settings will be used. Refer to CC Function Block Tuning.

If you do not know the process models, you need to identify the models and tune the function block by using the built-in tuner (modeler) for the function block to operate correctly in the Auto mode.

The function block uses first order lag with deadtime internal process models and first order filters (total of up to twelve tuning parameters) to calculate the CV's. Each CV is calculated such that the process variable (PV) follows a first order lag trajectory when approaching the setpoint value.

Speed of response depends on the value of the response time constants. The smaller the response time constants, the faster the control variable response will be. The response time constants should be set such that the PV reaches the setpoint in reasonable time based on the process dynamics. The larger the response time constants are, the slower the control variable response will be, but the control variable also becomes more robust. See the tuning section for more details.

In the Manual mode, the control variables (CV) are set equal to the operator-entered or program-generated CVnOper or CVnProg parameters. For the Manual to Auto mode bumpless transfer and for safe operation of the control variable, the CV rate of change limiters are implemented such that CV's

cannot move from current states by more than specified CV units at each scan.

To limit the CV rate of change:

- Set the CVnROCPosLimit and CVnROCNegLimit

Rate limiting is not imposed when control variable is in Manual mode unless CVManLimiting is set.

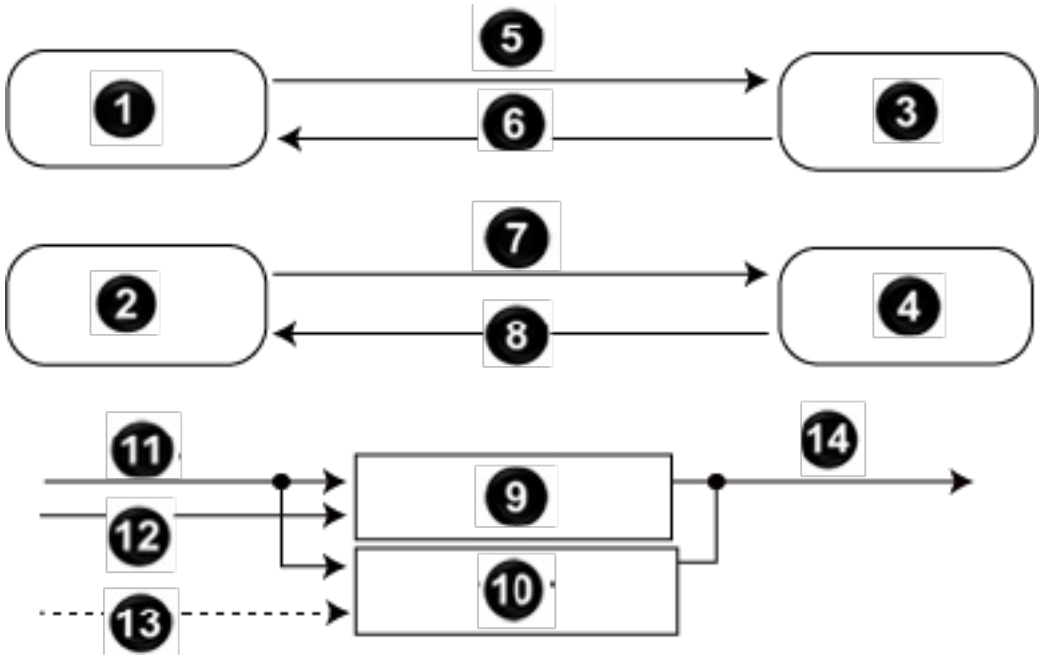
See also

CC Function Block Tuning on page 181

CV High/Low Limiting

The instruction always performs alarming based on CVHLimit and CVLLimit. Limit CV by CVHLimit and CVLLimit when in auto or cascade/ratio mode. When in manual mode, limit CV by CVHLimit and CVLLimit when CVManLimiting is set. Otherwise limit CV by 0 and 100%.

Follow the guidelines in this diagram:



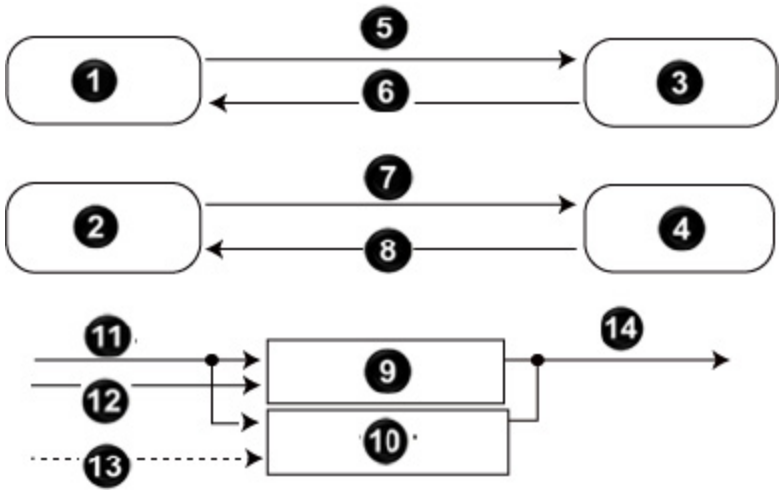
Item	Description
1	CVHAlarm is cleared[1]
2	CVLAlarm is cleared[1]
3	CVHAlarm is set
4	CVLAlarm is set
5	CV is greater than CVHLimit
6	CV is less than or equal to CVHLimit

Item	Description
7	CV is less than CVLLimit
8	CV is greater than or equal to CVLLimit
9	If CVHAlarm is set CV = CVHLimit
10	If CVLAlarm is set CV = CVLLimit
11	CV from 0 to 100% limit algorithm
12	CVHAlarm is set and auto or cascasde/ratio or (manual and CVManLimiting is set)
13	CVLAlarm is set and auto or cascasde/ratio or (manual and CVManLimiting is set).
14	CV limited to CV high/low limits.

(1) During instruction first scan, the instruction clears the alarm outputs.

CV Percent Limiting

The following diagram illustrates how the instruction determines CV percent limiting.



Item	Description
1	CVHAlarm is cleared(1)
2	CVLAlarm is cleared(1)
3	CVHAlarm is set
4	CVLAlarm is set
5	CV is greater than 100
6	CV is less than or equal to 0
7	CV is less than 0
8	CV is greater than or equal to 0
9	If CVHAlarm is set CV = 100

Item	Description
⑩	If CVLAlarm is set CV = 100
⑪	CV from windup algorithm
⑫	CVHAlarm
⑬	CVLAlarm
⑭	CV limited to 0 to 100%

(1) During instruction first scan, the instruction clears the alarm outputs.

CV Rate-of-Change Limiting

The PIDE instruction limits the rate-of-change of CV when in Auto or Cascade/Ratio mode or when in Manual mode and CVManLimiting is set. A value of zero disables CV rate-of-change limiting.

The CV rate-of-change is calculated as:

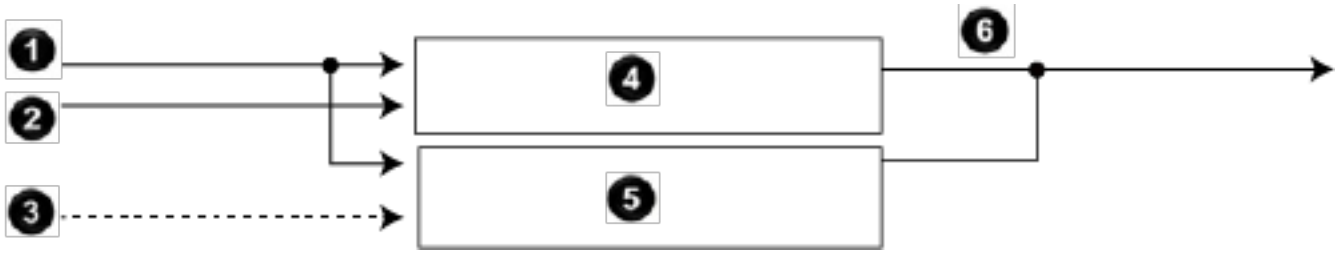
$$CVROC = \left| CV_n - CV_{n-1} \right|$$

$$CVROCDelta = CVROCLimit \times DeltaT$$

where DeltaT is in seconds.

CV Windup Limiting

Limit the CV such that its value cannot increase when WindupHIn is set or decrease when WindupLIn is set. These inputs are typically the WindupHOut or WindupLOut outputs from a secondary loop. The WindupHIn and WindupLIn inputs are ignored if CVInitializing, CVFault, or CVEUSpanInv is set.



Item	Description
❶	selected CV
❷	WindupHIn
❸	WindupLIn
❹	if WindupHIn and CV is greater than CV_{n-1} CV = CV_{n-1}
❺	if WindupHIn and CV is less than CV_{n-1} CV = CV_{n-1}
❻	CV from windup algorithm

Execution

Math status flags are set for the CV output.

Condition	Function Block Action	Structured Text Action
Prescan	InstructionFirstScan is set	InstructionFirstScan is set
Instruction First Scan	If CVFault and CVEUSpanInv are set, see Processing Faults. If CVFault and CVEUSpanInv are cleared: 1. CVInitializing is set. 2. If PVFault is set, PVSpanInv and SPLimitsInv are cleared. See Processing Faults. 3. The PID control algorithm is not executed. 4. The instruction sets CVEU = CVInitValue and CV = corresponding percentage. CVInitValue is not limited by CVEUmaximum or CVEUMin. When the instruction calculates CV as the corresponding percentage, it is limited to 0-100. $CVEU = CVInitValue$ $CV_{n-1} = CV = \frac{CVEU - CVEUMin}{CVEUMax - CVEUMin} \times 100$ $CVOper = CV$ 5. When CVInitializing and ManualAfterInit are set, the instruction disables auto and cascade/ratio modes. If the current mode is not Override or Hand mode, the instruction changes to Manual mode. If ManualAfterInit is cleared the mode is not changed. 6. All the operator request inputs are cleared. 7. If ProgValueReset set, all the program request inputs are cleared 8. All the PV high-low, PV rate-of-change, and deviation high-low alarm outputs are cleared. 9. If CVInitReq is cleared, CVInitializing is cleared.	
Instruction first run	ProgOper is cleared. The instruction changes to manual mode.	ProgOper is cleared. The instruction changes to manual mode.
EnableIn is cleared	EnableOut is cleared, the instruction does nothing, and the outputs are not updated.	N/A
EnableIn is set	The instruction executes. EnableOut is set.	EnableIn is always set. The instruction executes.
Postscan	No action taken.	No action taken.

See also

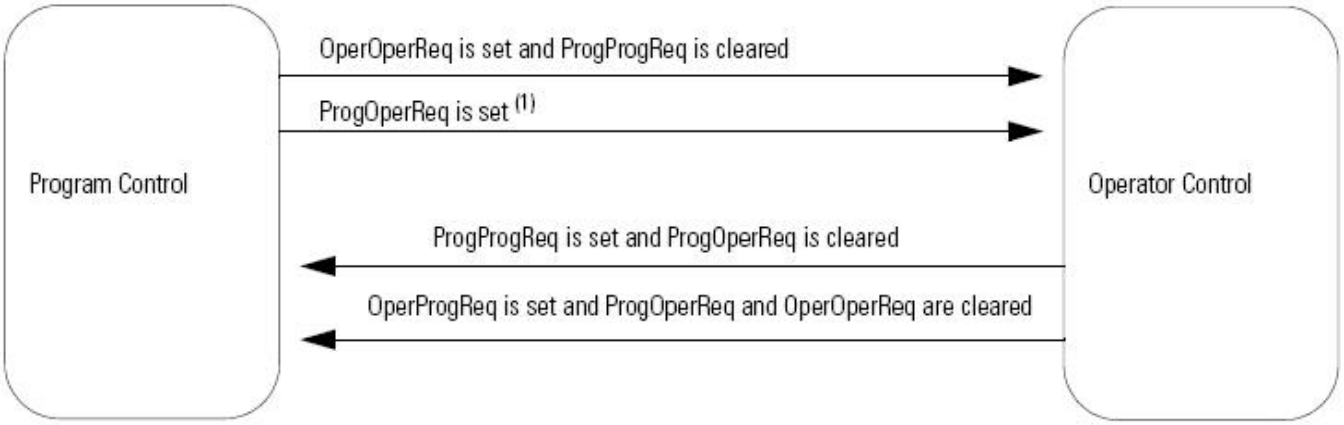
Processing Faults on page 254

Switch Between Program Control and Operator Control

The PIDE instruction can be controlled by either a user program or an operator interface. You can change the control mode at any time. Program and

Operator control use the same ProgOper output. When ProgOper is set, control is Program; when ProgOper is cleared, control is Operator.

The following diagram shows how the PIDE instruction changes between Program control and Operator control.



(1) The instruction remains in Operator control mode when ProgOperReq is set.

For more information on program and operator control, refer to Program/Operator Control.

See also

Program/Operator Control on page 1052

Operating Modes

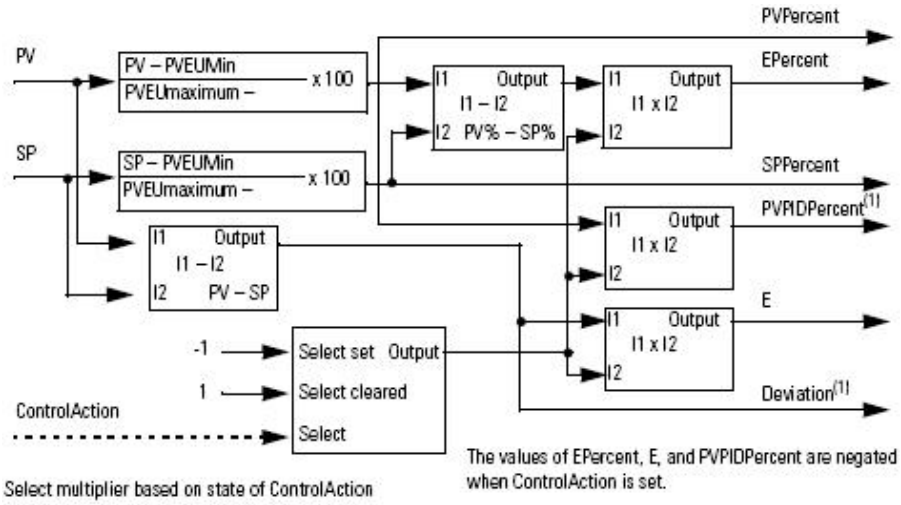
The Cascade/Ratio, Auto, and Manual modes can be controlled by a user program when in Program control or by an operator interface when in Operator control. The Override and Hand modes have a mode request input that can only be controlled by a user program; these inputs operate in both Program and Operator control.

Operating Mode	Description
Cascade/Ratio	While in Cascade/Ratio mode the instruction computes the change in CV. The instruction regulates CV to maintain PV at either the SPCascade value or the SPCascade value multiplied by the Ratio value. SPCascade comes from either the CVEU of a primary PID loop for cascade control or from the "uncontrolled" flow of a ratio-controlled loop. Select Cascade/Ratio mode using either OperCasRatReq or ProgCasRatReq: Set OperCasRatReq to request Cascade/Ratio mode. Ignored when ProgOper, ProgOverrideReq, ProgHandReq, OperAutoReq, or OperManualReq is set, or when AllowCasRat is cleared. Set ProgCasRatReq to request Cascade/Ratio mode. Ignored when ProgOper or AllowCasRat is cleared or when ProgOverrideReq, ProgHandReq, ProgAutoReq, or ProgManualReq is set.
Auto	While in Auto mode the instruction computes the change in CV. The instruction regulates CV to maintain PV at the SP value. If in program control, SP = SPProg; if in Operator control, SP = SPOper. Select Auto mode using either OperAutoReq or ProgAutoReq: Set OperAutoReq to request Auto mode. Ignored when ProgOper, ProgOverrideReq, ProgHandReq, or OperManualReq is set. Set ProgAutoReq to request Auto mode. Ignored when ProgOper is cleared or when ProgOverrideReq, ProgHandReq, or ProgManualReq is set.

Operating Mode	Description
Manual	While in Manual mode the instruction does not compute the change in CV. The value of CV is determined by the control. If in Program control, CV = CVProg; if in Operator control, CV = CVOper. Select Manual mode using either OperManualReq or ProgManualReq: Set OperManualReq to request Manual mode. Ignored when ProgOper, ProgOverrideReq, or ProgHandReq is set. Set ProgManualReq to request Manual mode. Ignored when ProgOper is cleared or when ProgOverrideReq or ProgHandReq is set.
Override	While in Override mode the instruction does not compute the change in CV. CV = CVOverride, regardless of the control mode. Override mode is typically used to set a "safe state" for the PID loop. Select Override mode using ProgOverrideReq: Set ProgOverrideReq to request Override mode. Ignored when ProgHandReq is cleared.
Hand	While in Hand mode the PID algorithm does not compute the change in CV. CV = HandFB, regardless of the control mode. Hand mode is typically used to indicate that control of the final control element was taken over by a field hand/auto station. Select Hand mode using ProgHandReq: Set ProgHandReq to request hand mode. This value is usually read as a digital input from a hand/auto station.

Convert the PV and SP Values to Percent

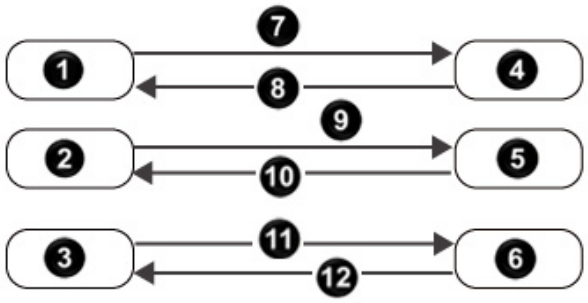
The instruction converts PV and SP to a percent and calculates the error before performing the PID control algorithm. The error is the difference between the PV and SP values. When ControlAction is set, the values of EPercent, E, and PVPIDPercent are negated before being used by the PID algorithm.



(1) PVPIDPercent and Deviation are internal parameters used by the PID control algorithm.

Primary Loop Control

Primary loop control is typically used by a primary PID loop to obtain bumpless switching and anti-reset windup when using Cascade/Ratio mode. The primary loop control includes the initialize primary loop output and the anti-reset windup outputs. The InitPrimary output is typically used by the CVInitReq input of a primary PID loop. The windup outputs are typically used by the windup inputs of a primary loop to limit the windup of its CV output.



Item	Description
1	InitPrimary is cleared
2	WindupHOut is cleared[4]
3	WindupLOut is cleared[4]
4	InitPrimary is set[1]
5	WindupHOut is set
6	WindupLOut is set
7	CVInitializing is set or not Cascade/Ratio mode[2]
8	CVInitializing is cleared and Cascade/Ration mode[3]
9	SHPAlarm is set or appropriate Cv alarm[5]
10	SHPAlarm is cleared and no CV alarm[6]
11	SPAlarm is set or appropriate CV alarm[7]
12	SPAlarm is cleared and no CV alarm[8]

1. During first scan, the instruction sets InitPrimary
2. When CVInitializing is set or when not in Cascade/Ratio mode, the instruction sets InitPrimary.
3. When CVInitializing is cleared and in Cascasde/Ratio mode, the instruction clears InitPrimary.
4. During instruction first scan, the instruction clears the windup outputs. The instruction also clears the windup outputs and disables the CV windup algorithm when CVInitializing is set or if either CVFaulted or CVEUSpanlnv is set.
5. The instruction sets WindupHOut when SPHalarm is set, or when ControlAction is cleared and CVHAlarm is set, or when ControlAction is set and CVLAarm is set.
   The SP and CV limits operate independently. A SP high limit does not prevent CV from increasing in value. Likewise, a CV high or low limit does not prevent SP from increasing in value.
6. The instruction clears WindupHOut when SPHalarm is cleared, and not (ControlAction is cleared and CVHAlarm is set), and not (ControlAction is set and CVLAlarm is set).
7. The instruction sets WinindupLOut when SPLAlarm is set, or when ControlAction is cleared and CVLAlarm is set or when ControlAction is set and CVHAlarm is set.

The SP and CV limits operate independently. A SP low limit does not prevent CV from increasing in value likewise a CV low or high limit does not prevent SP from increasing in value.

8. The instruction clears WindupLOut when SPLAlarm is cleared and not (ControlAction is cleared and CVLAlarm is set) and not (ControlAction is set and CVHAlarm is set).

Processing Faults

The following table describes how the instruction handles execution faults:

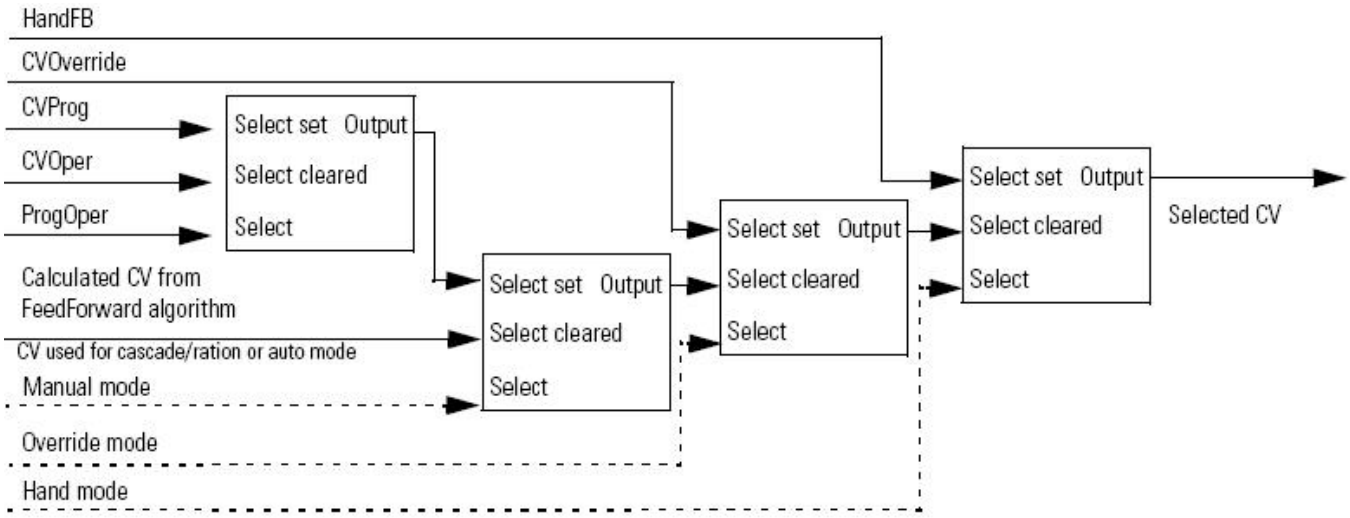
Fault Condition	Action
CVFaulted or CVEUSpanInv is set	• Instruction is not initialized, CVInitializing is cleared. • Compute PV and SP percent, calculate error, update internal parameters for EPercent and PVPIDPercent • PID control algorithm is not executed. • Disable the Auto and Cascade/Ratio modes. If Override or Hand is not the current mode, set to Manual mode. • Set CV to value determined by Program or Operator control and mode (Manual, Override, or Hand).
CVinitRequest	Refer to Execution.
PV Health Bad	• Disable the Auto and CasRat modes. If Override or Hand is not the current mode then set to the Manual mode. • Set PV high-low, PV rate-of-change, and deviation high-low alarm outputs FALSE • PID control algorithm is not executed. • Set CV to value by determined by Program or Operator control and mode (Manual, Override or Hand).
PVFaulted is set	• Disable the Auto and Cascade/Ratio modes. If Override or Hand is not the current mode, set to Manual mode. • PV high-low, PV rate-of-change, and deviation high-low alarm outputs are cleared • PID control algorithm is not executed. • Set CV to value by determined by Program or Operator control and mode (Manual, Override, or Hand).
PVSpanInv is set or SPLimitsInv is set	• Disable the Auto and Cascade/Ratio modes. If Override or Hand is not the current mode, set to Manual mode. • Do not compute PV and SP percent. • PID control algorithm is not executed. • Set CV to value by determined by Program or Operator control and mode (Manual, Override, or Hand).
RatioLimitsInv is set and CasRat is set and UseRatio is set	• If not already in Hand or Override, set to Manual mode. • Disable the Cascade/Ratio mode. • Set CV to value determined by Program or Operator control and mode (Manual, Override, or Hand).
TimingModeInv is set or RTSTimeStampInv is set or DeltaTInv is set	• If not already in Hand or Override, set to Manual mode.

See also

Execution on page 250

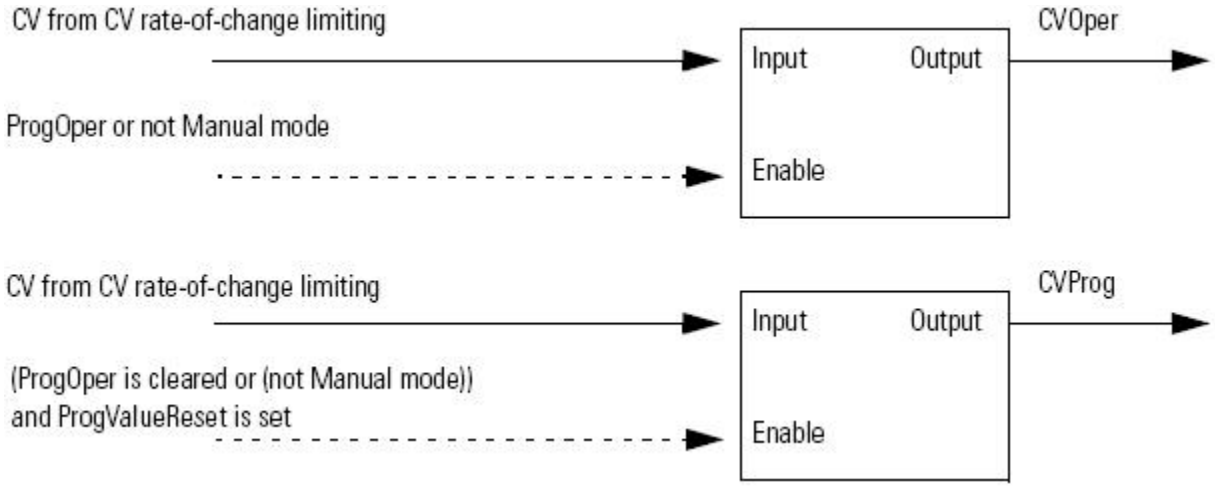
Select the Control Variable

Once the PID algorithm has been executed, select the CV based on program or operator control and the current PID mode.



Update the CVOper and CVProg Values

If not in the Operator Manual mode, the PIDE instruction sets CVOper = CV. This obtains bumpless mode switching from any control to the Operator Manual mode.



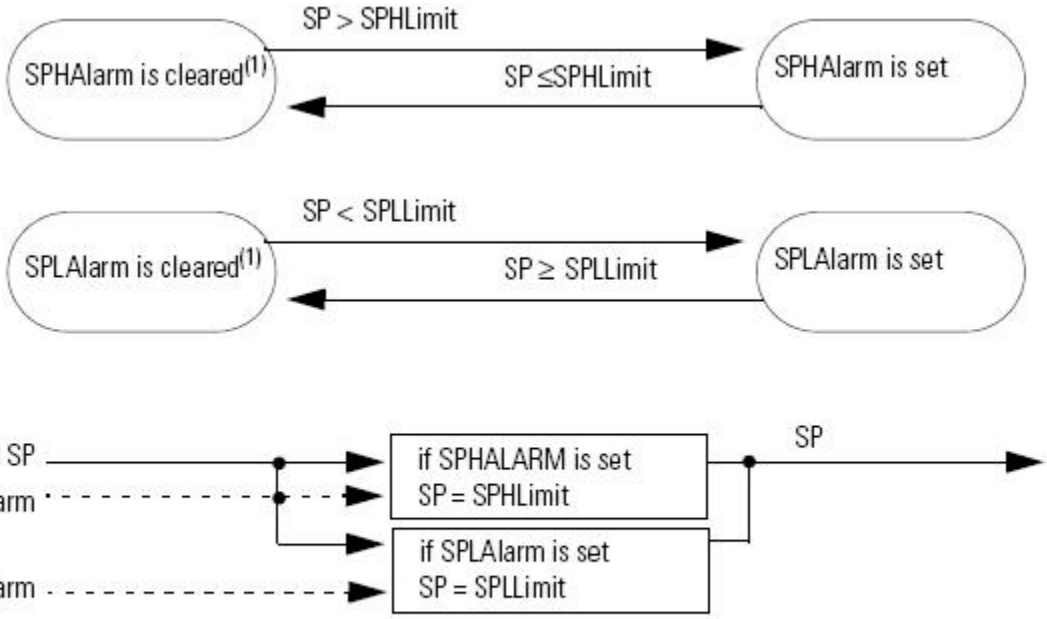
Select the Setpoint

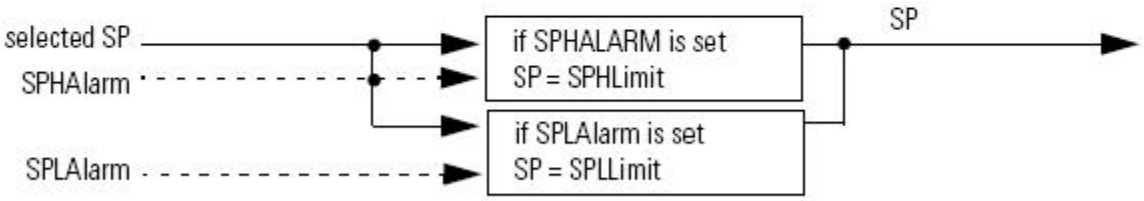
Once the instruction determines program or operator control and the PID mode, the instruction can obtain the proper SP value.

- Enhanced PID on page 65
- Current SP on page 245
- SP High/Low Limiting on page 255

SP High/Low Limiting

The high-to-low alarming algorithm compares SP to the SPHLimit and SPLLimit alarm limits. SPHLimit cannot be greater than PVEUmaximum and SPLLimit cannot be less than PVEUMin.





(1) During instruction first scan, the instruction clears the SP alarm outputs. The instruction also clears the SP alarm limits and disables the alarming algorithm when PVSpanInv is set.

PlantPAx

PlantPAx instructions

The PlantPAx built-in instructions monitor and process discrete and analog inputs and outputs for controlling devices.

Available Instructions

Ladder Diagram, Function Block, Structured Text

PAH	PAI	PAID	PAIM	PAO	PBL	PCMDSRC	PDBC

PDI	PDO	PDOSE	PFO	PHLS	PINTLK	PLLS	PMTR

PPERM	PPID	PPTC	PRI	PRT	PTST	PVLV	PVLVS

PVSD

To:	Use this instruction:
Provide HART digital data for an intelligent analog device alongside the analog input (PAI) or analog output (PAO) instruction for that device.	Process Analog HART (PAH)
Monitor one analog value, such as from a channel of analog input module.	Process Analog Input (PAI)
Monitor one analog Process Variable (PV) by using two analog input signals.	Process Dual Sensor Analog Input (PAID)
Monitors one analog process variable (PV) by using up to eight analog input signals.	Process Multi Sensor Analog Input (PAIM)
Manipulate an analog output to control a field device.	Process Analog Output (PAO)
Executes up to eight gates of configurable Boolean logic.	Process Boolean Logic (PBL)
Select the Command Source for a device.	Process Command Source (PCMDSRC)
Generate outputs to provide data and alarms on deadbands and thresholds.	Process Deadband Controller (PDBC)

To:	Use this instruction:
Monitor one discrete condition, such as from a channel of a discrete input module.	Process Discrete Input (PDI)
Manipulate a discrete output to control a field device.	Process Discrete Output (PDO)
Control an ingredient addition that uses a flow meter to measure the quantity of ingredient added.	Process Dosing (PDOSE)
Send one primary analog output signal to multiple secondary users or devices.	Process Analog Fanout (PFO)
Select the highest or the lowest of up to six incoming controlled variables (CVs) and send the selected CV as output.	Process High or Low Selector (PHLS)
Collect, or sum up, the interlock conditions that stop or de-energize a running or energized piece of equipment.	Process Interlocks (PINTLK)
Control a parallel group of motors, such as a set of pumps with a common intake source and discharge destination.	Process Lead Lag Standby Motor Group (PLLS)
Monitor and control a fixed single-speed, two-speed, or reversing motor.	Process Motor (PMTR)
Collect, or sum up, the permissive conditions that allow a piece of equipment to energize.	Process Permissives (PPERM)
Manipulate the Control Variable (CV) in regulatory control loops in response to Process Variable (PV) readings and Setpoint (SP, the target PV) settings.	Process Proportional + Integral + Derivative (PPID)
Calculate a flow at standard temperature and pressure.	Process Pressure/Temperature Compensated Flow (PPTC)
Prevent large motors from starting repeatedly.	Process Restart Inhibit (PRI)
Record the total run time and number of instances the motor or other equipment starts.	Process Run Time and Start Counter (PRT)
Calculate the volume of product in an upright cylindrical tank, given the level of the product and the tank calibration table.	Process Tank Strapping Table (PTST)
Operate a two-position, single-solenoid operated valve, a dual-solenoid operated valve, or a motor-operated valve in various modes; monitor hand-operated two-position valves.	Process Valve (PVLV)
Monitor a two-state (open and close) valve and record statistics for stroke times and stroke counts.	Process Valve Statistics (PVLVS)

See also

Drives Instructions on page 823

Filter Instructions on page 875

Move/Logical Process Instructions on page 955

Process Analog HART (PAH)

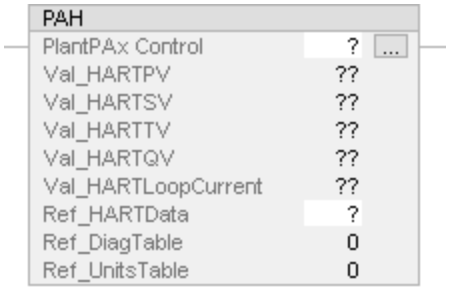
This information applies to the ControlLogix 5380P and 5580P controllers.

The Process Analog HART (PAH) instruction is used to provide HART digital data for an intelligent analog device alongside the analog input (PAI) or analog output (PAO) instruction for that device. It provides:

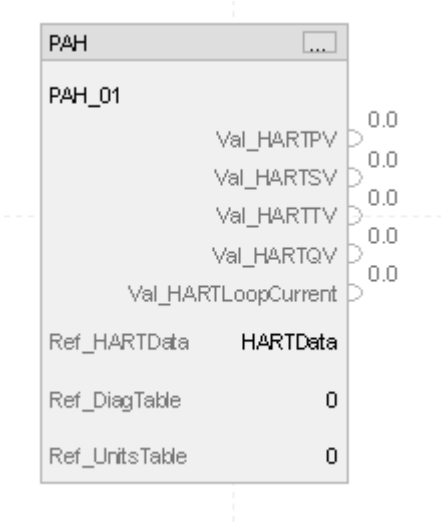
- HART Dynamic Variables (PV, SV, TV and QV) values with engineering units and variable status, and the digital value of the device's analog loop current (in milliamps DC).
- Analog range (min, at 4 mA, and max, at 20 mA).
- Analog units of measure.
- Device information such as Tag and Description text.
- Additional device status (HART "Command 48 additional status" bit array).
- Lookup of diagnostic message and severity based on the Command 48 bits that are set (for the first three diagnostics found in the array).
- Additional status information received from the device via HART, such as Field Device Status bits.

Available Languages

Ladder Diagram



Function Block Diagram



Structured Text

PAH(PAH_tag, Ref_HARTData, Ref_DiagTable, Ref_UnitsTable)

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PlantPAx ControlPlantPAx Control	P_ANALOG_HART	tag	Data structure required for proper operation of instruction.
Ref_HARTData	PAX_HART_DEVICE:I:0	tag	HART device data for PlantPAx.
Ref_DiagTable	P_HART_CODE_DESC_STATUS[2]	tag	Lookup table for diagnostic bit number (to message and status).
Ref_UnitsTable	RAC_CODE_DESCRIPTION[2]	tag	Lookup table for units of measure code (to units text).

The PAX_HART_DEVICE:I:0 data type is associated with the Add-On Profile for Highly Integrated HART modules such as the 5094-IF8IH.

P_ANALOG_HART Structure

Public members are standard, visible tag members that are programmatically accessible. Private, hidden members are used in HMI faceplates and are not programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
EnableIn	BOOL	Not Visible	Not Required	Input	Enable input. Ladder Diagram: Corresponds to the rung condition. Default is true.
Inp_InitializeReq	BOOL	Not Visible	Not Required	Input	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request to re-initialize. The instruction clears this operand automatically. Default is true.
Cfg_HasHARTPV	BOOL	Not Visible	Not Required	Input	1 = Has a HART digital PV, display on faceplate; 0 = HART digital PV not used. Default is false.
Cfg_HasHARTSV	BOOL	Not Visible	Not Required	Input	1 = Has a HART digital SV, display on faceplate; 0 = HART digital SV not used. Default is false.
Cfg_HasHARTTV	BOOL	Not Visible	Not Required	Input	1 = Has a HART digital TV, display on faceplate; 0 = HART digital TV not used. Default is false.
Cfg_HasHARTQV	BOOL	Not Visible	Not Required	Input	1 = Has a HART digital QV, display on faceplate; 0 = HART digital QV not used. Default is false.
Cfg_UseHARTVarSts	BOOL	Not Visible	Not Required	Input	1 = Use HART Communication Status to generate SrcQ, 0 = assume good. Default is true.
Cfg_UseHARTText	BOOL	Not Visible	Not Required	Input	1 = Use text received from HART device, 0 = use extended properties for text. Default is false.
Cfg_HARTPVDecPlcs	SINT	Not Visible	Not Required	Input	Number of decimal places for HART PV display. Valid = 0,1,2,3,4,5,6. Default is 2.
Cfg_HARTSVDecPlcs	SINT	Not Visible	Not Required	Input	Number of decimal places for HART SV display. Valid = 0,1,2,3,4,5,6. Default is 2.
Cfg_HARTTVDecPlcs	SINT	Not Visible	Not Required	Input	Number of decimal places for HART TV display. Valid = 0,1,2,3,4,5,6. Default is 2.
Cfg_HARTQVDecPlcs	SINT	Not Visible	Not Required	Input	Number of decimal places for HART QV display. Valid = 0,1,2,3,4,5,6. Default is 2.
Cfg_HasMoreObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI an object with more information is available for navigation. Default is false.
Cfg_HasNav	BOOL	Not Visible	Not Required	Input	1 = Tells HMI a related analog input or output object is available for navigation. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Set_VirtualHARTPV	REAL	Not Visible	Not Required	Input	HART PV used in Virtual (when Sts_Virtual = 1) (PV engineering units). Default is 0.0.
Set_VirtualHARTSV	REAL	Not Visible	Not Required	Input	HART SV used in Virtual (when Sts_Virtual = 1) (SV engineering units). Default is 0.0.
Set_VirtualHARTTV	REAL	Not Visible	Not Required	Input	HART TV used in Virtual (when Sts_Virtual = 1) (TV engineering units). Default is 0.0.
Set_VirtualHARTQV	REAL	Not Visible	Not Required	Input	HART QV used in Virtual (when Sts_Virtual = 1) (QV engineering units). Default is 0.0.
PCmd_Virtual	BOOL	Not Visible	Not Required	Input	Program command to select Virtual (simulated) device operation. The instruction clears this operand automatically. Default is false.
PCmd_Physical	BOOL	Not Visible	Not Required	Input	Program command to select Physical device operation (not simulated). The instruction clears this operand automatically. Default is false.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
EnableOut	BOOL	Not Visible	Not Required	Output	This output state always reflects EnableIn input state.
Val_HARTPV	REAL	Visible	Not Required	Output	Digital HART PV value in PV engineering units (after Substitution, if used).
Val_HARTSV	REAL	Visible	Not Required	Output	Digital HART SV value in SV engineering units (after Substitution, if used).
Val_HARTTV	REAL	Visible	Not Required	Output	Digital HART TV value in TV engineering units (after Substitution, if used).
Val_HARTQV	REAL	Visible	Not Required	Output	Digital HART QV value in QV engineering units (after Substitution, if used).
Val_HARTLoopCurrent	REAL	Visible	Not Required	Output	Digital HART value for Loop Current in milliamps.
Val_InpRawMinFromHART	REAL	Not Visible	Not Required	Output	Analog input unscaled signal minimum from HART module (in module units).
Val_InpRawMaxFromHART	REAL	Not Visible	Not Required	Output	Analog input unscaled signal maximum from HART module (in module units).
Val_PVEUMinFromHART	REAL	Not Visible	Not Required	Output	Analog input scaled range minimum from HART device (in engineering units).
Val_PVEUMaxFromHART	REAL	Not Visible	Not Required	Output	Analog input scaled range maximum from HART device (in engineering units).
Sts_eHARTDiagCode1	INT	Not Visible	Not Required	Output	HART Diagnostic Code #1 (bit number in Command 48, 255 = none).
Sts_eHARTDiagCode2	INT	Not Visible	Not Required	Output	HART Diagnostic Code #2 (bit number in Command 48, 255 = none).
Sts_eHARTDiagCode3	INT	Not Visible	Not Required	Output	HART Diagnostic Code #3 (bit number in Command 48, 255 = none).

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_bHARTDiagSts	SINT	Not Visible	Not Required	Output	Overall HART diagnostic status, .0 = Info, .1 = Maintenance Required, .2 = Off Specification, .3 = Function Check, .4 = Failed.
Sts_bHARTDiagSts1	SINT	Not Visible	Not Required	Output	Diagnostic status for HART Diagnostic Code #1, .0 = Info, .1 = Maintenance Required, .2 = Off Specification, .3 = Function Check, .4 = Failed.
Sts_bHARTDiagSts2	SINT	Not Visible	Not Required	Output	Diagnostic status for HART Diagnostic Code #2, .0 = Info, .1 = Maintenance Required, .2 = Off Specification, .3 = Function Check, .4 = Failed.
Sts_bHARTDiagSts3	SINT	Not Visible	Not Required	Output	Diagnostic status for HART Diagnostic Code #3, .0 = Info, .1 = Maintenance Required, .2 = Off Specification, .3 = Function Check, .4 = Failed.
Sts_Initialized	BOOL	Not Visible	Not Required	Output	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.
Sts_Virtual	BOOL	Not Visible	Not Required	Output	1 = Using virtual PV instead of the input from the device (Inp_PVData) to calculate output. 0 = The instruction uses input parameter Inp_PVData to calculate output. Sts_Virtual is a copy of Inp_Virtual.
Sts_ConnectionFault	BOOL	Not Visible	Not Required	Output	1 = HART data input connection fault, 0 = connection OK.
Sts_DvcMalfunction	BOOL	Not Visible	Not Required	Output	1 = HART device reports it has a malfunction.
Sts_CurrentSaturated	BOOL	Not Visible	Not Required	Output	1 = HART reports analog current is limited.
Sts_CurrentFixed	BOOL	Not Visible	Not Required	Output	1 = Loop Current set to fixed value via HART command.
Sts_CurrentMismatch	BOOL	Not Visible	Not Required	Output	1 = Loop Current reported via HART does not match analog signal.
Sts_DiagnosticActive	BOOL	Not Visible	Not Required	Output	1 = HART data input diagnostic active.
Val_DiagnosticSeqCount	SINT	Not Visible	Not Required	Output	HART data input diagnostic sequence count (per change in diagnostic data, wraps).

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
SrcQ_IO	SINT	Not Visible	Not Required	Output	Source and quality of primary input or output (enumerated): 0=Good, live, confirmed good 1=Good, live, assumed good 2=Good, no feedback, assumed good 8=Test, virtualized 9=Test, loopback 10=Test, manually entered 16=Uncertain, live, off-spec 17=Uncertain, substituted at device or bus 18=Uncertain, substituted at instruction 19=Uncertain, using last known good 20=Uncertain, using replacement value 32=Bad, signal failure 33=Bad, channel fault 34=Bad, module or communication fault 35=Bad, invalid configuration
SrcQ	SINT	Not Visible	Not Required	Output	Source and quality of primary value or status (enumerated): 0=Good, live, confirmed good 1=Good, live, assumed good 2=Good, no feedback, assumed good 8=Test, virtualized 9=Test, loopback 10=Test, manually entered 16=Uncertain, live, off-spec 17=Uncertain, substituted at device or bus 18=Uncertain, substituted at instruction 19=Uncertain, using last known good 20=Uncertain, using replacement value 32=Bad, signal failure 33=Bad, channel fault 34=Bad, module or communication fault 35=Bad, invalid configuration
SrcQ_HARTPV	SINT	Not Visible	Not Required	Output	Source and quality of HART digital PV (enumerated): 0=Good, live, confirmed good 1=Good, live, assumed good 2=Good, no feedback, assumed good 8=Test, virtualized 9=Test, loopback 10=Test, manually entered 16=Uncertain, live, off-spec 17=Uncertain, substituted at device or bus 18=Uncertain, substituted at instruction 19=Uncertain, using last known good 20=Uncertain, using replacement value 32=Bad, signal failure 33=Bad, channel fault 34=Bad, module or communication fault 35=Bad, invalid configuration

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
SrcQ_HARTSV	SINT	Not Visible	Not Required	Output	Source and quality of HART digital SV (enumerated): 0=Good, live, confirmed good 1=Good, live, assumed good 2=Good, no feedback, assumed good 8=Test, virtualized 9=Test, loopback 10=Test, manually entered 16=Uncertain, live, off-spec 17=Uncertain, substituted at device or bus 18=Uncertain, substituted at instruction 19=Uncertain, using last known good 20=Uncertain, using replacement value 32=Bad, signal failure 33=Bad, channel fault 34=Bad, module or communication fault 35=Bad, invalid configuration
SrcQ_HARTTV	SINT	Not Visible	Not Required	Output	Source and quality of HART digital TV (enumerated): 0=Good, live, confirmed good 1=Good, live, assumed good 2=Good, no feedback, assumed good 8=Test, virtualized 9=Test, loopback 10=Test, manually entered 16=Uncertain, live, off-spec 17=Uncertain, substituted at device or bus 18=Uncertain, substituted at instruction 19=Uncertain, using last known good 20=Uncertain, using replacement value 32=Bad, signal failure 33=Bad, channel fault 34=Bad, module or communication fault 35=Bad, invalid configuration
SrcQ_HARTQV	SINT	Not Visible	Not Required	Output	Source and quality of HART digital QV (enumerated): 0=Good, live, confirmed good 1=Good, live, assumed good 2=Good, no feedback, assumed good 8=Test, virtualized 9=Test, loopback 10=Test, manually entered 16=Uncertain, live, off-spec 17=Uncertain, substituted at device or bus 18=Uncertain, substituted at instruction 19=Uncertain, using last known good 20=Uncertain, using replacement value 32=Bad, signal failure 33=Bad, channel fault 34=Bad, module or communication fault 35=Bad, invalid configuration

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
SrcQ_HARTLoopCurrent	SINT	Not Visible	Not Required	Output	Source and quality of HART loop current value (enumerated): 0=Good, live, confirmed good 1=Good, live, assumed good 2=Good, no feedback, assumed good 8=Test, virtualized 9=Test, loopback 10=Test, manually entered 16=Uncertain, live, off-spec 17=Uncertain, substituted at device or bus 18=Uncertain, substituted at instruction 19=Uncertain, using last known good 20=Uncertain, using replacement value 32=Bad, signal failure 33=Bad, channel fault 34=Bad, module or communication fault 35=Bad, invalid configuration
Sts_eSts	SINT	Not Visible	Not Required	Output	Device confirmed status: 0 = Live, 1 = diagnostic information, 2 = maintenance required, 3 = off-spec (uncertain), 4 = function check (substituted), 5 = failure, 6 = HART communication lost, 7 = Virtualized.
Sts_eFault	INT	Not Visible	Not Required	Output	Device fault status: 0 = None, 1 = a dynamic variable is bad, 2 = device diagnostic indicates a failure, 3 = HART communication lost, 4 = module connection fault, 5 = device reports malfunction.
Val_HARTRevision	SINT	Not Visible	Not Required	Output	HART Spec major revision received from device.

Private Input Members	Data Type	Description
MCmd_Physical	BOOL	Maintenance command to select Physical device operation (not simulated). The instruction clears this operand automatically. Default is false.
MCmd_Virtual	BOOL	Maintenance command to select Virtual (simulated) device operation. The instruction clears this operand automatically. Default is false.

Private Output Members	Data Type	Description
MRdy_Physical	BOOL	1 = Ready for MCmd_Physical, enable HMI button.
MRdy_Virtual	BOOL	1 = Ready for MCmd_Virtual, enable HMI button.
Sts_sHARTDesc	String_16	Description text from HART device.
Sts_sHARTDiagMsg1	String_32	HART device diagnostic message #1.
Sts_sHARTDiagMsg2	String_32	HART device diagnostic message #2.
Sts_sHARTDiagMsg3	String_32	HART device diagnostic message #3.

Private Output Members	Data Type	Description
Sts_sHARTPVEU	String_16	Text of HART digital PV's engineering units.
Sts_sHARTQVEU	String_16	Text of HART digital QV's engineering units.
Sts_sHARTSVEU	String_16	Text of HART digital SV's engineering units.
Sts_sHARTTag	String_32	Tag text from HART device.
Sts_sHARTTVEU	String_16	Text of HART digital TV's engineering units.
Sts_sPVEU	String_16	Text of analog PV's engineering units.

Public InOut Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Ref_HARTData	PAX_HART_DEVICE:I:0	Visible	Required	InOut	HART data from I/O module assembly.
Ref_DiagTable	P_HART_CODE_DESC_STATUS[2]	Visible	Required	InOut	Lookup table for diagnostic bit number (to message and status).
Ref_UnitsTable	RAC_CODE_DESCRIPTION[2]	Visible	Required	InOut	Lookup table for units of measure code (to units text).

PAX_HART_DEVICE:I:0 Structure

The PAX_HART_DEVICE:I:O structure is the input assembly subtype used by Highly-Integrated HART I/O modules to provide all the HART data required for this instruction:

Members	Data Type	Description
RunMode	BOOL	Always 0, not used by this instruction
ConnectionFaulted	BOOL	The network connection to the I/O module has been lost
DiagnosticActive	BOOL	I/O module has at least one diagnostic available
DiagnosticSequenceCount	SINT	This count increments each time the diagnostic information from the I/O module changes. It counts to +127, then loops back to -128, skipping zero.
CurrentSaturated	BOOL	The analog signal has reached its minimum or maximum value and does not represent the actual process variable
CurrentFixed	BOOL	The analog signal has been fixed by command and does not represent the actual process variable
MoreStatusAvailable	BOOL	At least one Command 48 (additional device status) bit is set and diagnostics should be displayed
CurrentMismatch	BOOL	I/O module reports digital value for loop current and actual analog loop current disagree significantly
ConfigurationChanged	BOOL	Device reports configuration data (scaling, text, units) have changed
Malfunction	BOOL	Device reports malfunction detected
LoopCurrent	CHANNEL_AI:I:0	HART digital value for device analog loop current (mA DC)
PV	CHANNEL_AI_HART:I:0	HART dynamic Primary Variable with units and status
SV	CHANNEL_AI_HART:I:0	HART dynamic Secondary Variable with units and status
TV	CHANNEL_AI_HART:I:0	HART dynamic Tertiary Variable with units and status
QV	CHANNEL_AI_HART:I:0	HART dynamic Quaternary Variable with units and status
Static	AB_5000_HART_Static_Struct:I:0	HART "static" data, such as device scale range, analog signal units of measure, and device text strings for description and tag name
ChDataAtSignal4	REAL	The value provided by the I/O module analog channel when a 4.0 mA DC signal is received, provided for scaling use by an associated PAI or PAO instruction

Members	Data Type	Description
ChDataAtSignal20	REAL	The value provided by the I/O module analog channel when a 20.0 mA DC signal is received, provided for scaling use by an associated PAI or PAO instruction

P_HART_CODE_DESC_STATUS Structure

The P_HART_CODE_DESC_STATUS structure is used to look up the diagnostic text and device status associated with a particular Command 48 diagnostic bit. The device provides a 200-bit (25 byte) array of data, where each bit set indicates a particular diagnostic condition. An array of members of this type is used to allow the instruction to display a text description and status for a given bit.

This parameter links the instruction to an external tag that contains necessary data for the instruction to operate. The external tag must be of the data type shown, or may be NULL. If NULL, the diagnostic bit lookup function is not performed.

Members	Data Type	Description
Code	DINT	Diagnostic code. This is the diagnostic bit number (0 to 199) in the array of 200 bits returned in HART Command 48 (Additional Device Status), or -1 if no diagnostic bit is set.
Desc	STRING_32	Diagnostic text.
bSts	SINT	Device Status (bitmapped): .0 = Information .1 = Maintenance Required .2 = Off Specification .3 = Function Check .4 = Failed

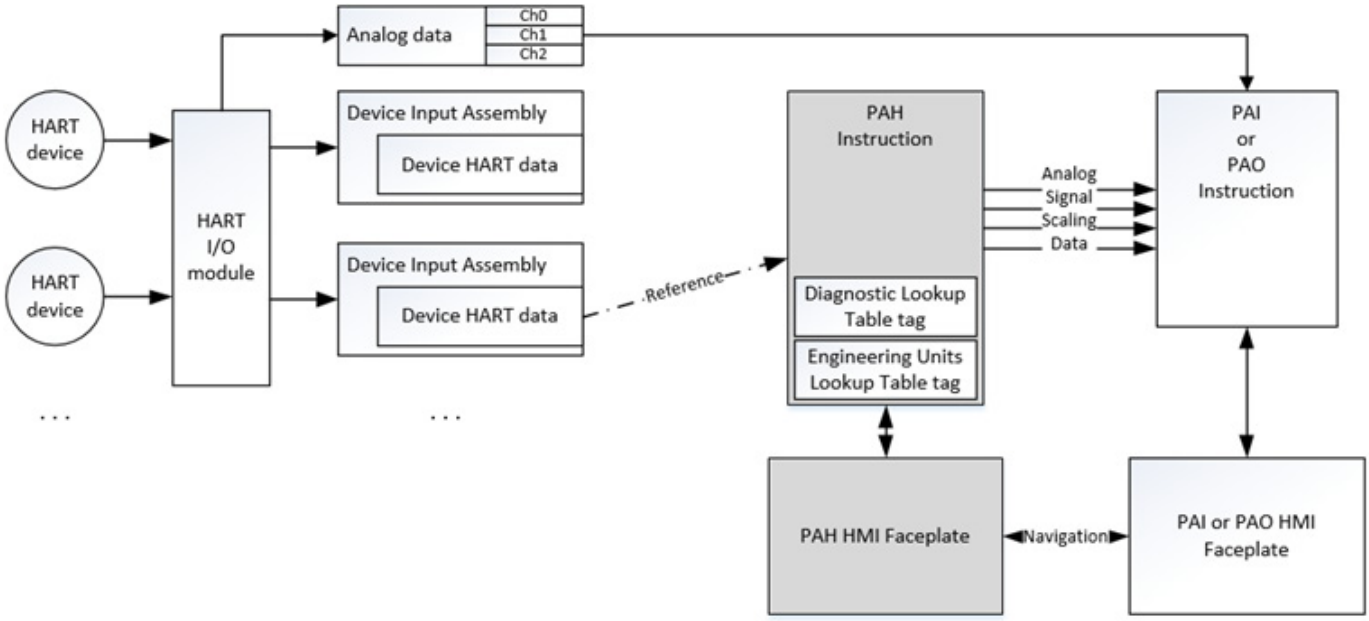
RAC_CODE_DESCRIPTION[x] Structure

The RAC_CODE_DESCRIPTION[x] structure is an array of engineering unit code numbers and corresponding engineering units text pairs, used as a lookup table. The instruction searches the table for the engineering units code received from the device and displays the corresponding engineering unit text for the variable. This parameter links the instruction to an external tag that contains necessary data for the instruction to operate. The external tag must be of the data type shown, or may be NULL. If NULL, the unit code lookup function is not performed.

Members	Data Type	Description
Code	DINT	Code for which to look up Description.
Desc	STRING	Description for given Code.

Operation

This diagram illustrates functionality of the PAH instruction:



Virtualization

Use virtualization for instruction testing and operator training. Command virtual operation using program command PCmd_Virtual or maintenance command MCmd_Virtual. After finishing virtual operation, use program command PCmd_Physical or maintenance command MCmd_Physical to return to normal physical device operation.

When Virtualization is active, the output dynamic variable (PV, SV, TV, QV) values of the PAH instruction are set using Virtual value settings (Set_VirtualPV, etc.) and I/O faults are ignored. Manipulate the instruction to operate as if a working HART process device were present.

Initialization

The instruction is normally initialized in the instruction first run. Request re-initialization by setting Inp_InitializeReq = 1. For proper initialization, when adding the instruction while performing an online edit of the code, make sure that Inp_InitializeReq = 1, the default value.

Configuration of Strings for HMI

Configure strings for HMI faceplates (FT View) and for the Logix Designer configuration dialog box. The strings are set to extended properties of tag items.

- Description
- Label for graphic symbol

- Display Library for HMI Faceplate call-up
- Instruction name
- Area name
- URL link
- Engineering units for raw analog signal
- Engineering units for analog signal PV
- Engineering units for HART PV
- Engineering units for HART SV
- Engineering units for HART TV
- Engineering units for HART QV
- Label for HART PV
- Label for HART SV

Monitor the PAH Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Instruction flagged for initialization on first scan / first run
Instruction first run	Internal data such as pointers and timers are initialized
Rung-condition-in is false	Set rung-condition-out to rung-condition-in. The instruction executes. HART data, units and status are provided.
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes. HART data, units and status are provided.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	Instruction flagged for initialization on first scan / first run
Instruction first run	Internal data such as pointers and timers are initialized
Instruction first scan	Internal data such as pointers and timers are initialized
EnableIn is false	EnableOut is set to false. The instruction executes. HART data, units and status are provided.
EnableIn is true	EnableOut is set to true. The instruction executes. HART data, units and status are provided.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

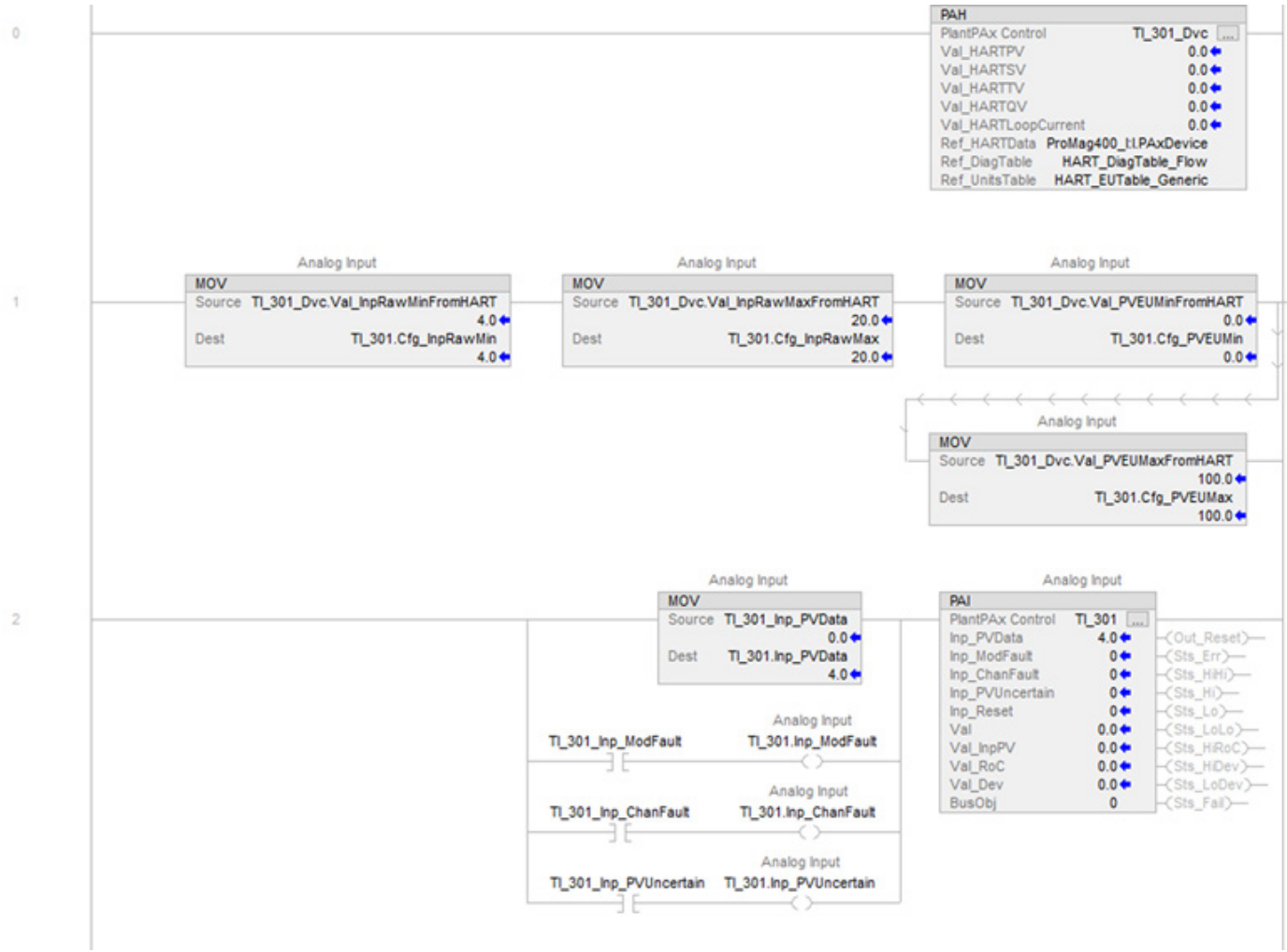
In Structured Text, EnableIn is always true during normal scan. Therefore, if the instruction is in the control path activated by the logic it will execute.

Condition/State	Action Taken
Prescan	See Prescan in the Function Block Diagram table.
Instruction first run	See Instruction first run in the Function Block Diagram table.
EnableIn is true	See EnableIn is true in the Function Block Diagram table.
Postscan	See Postscan in the Function Block Diagram table.

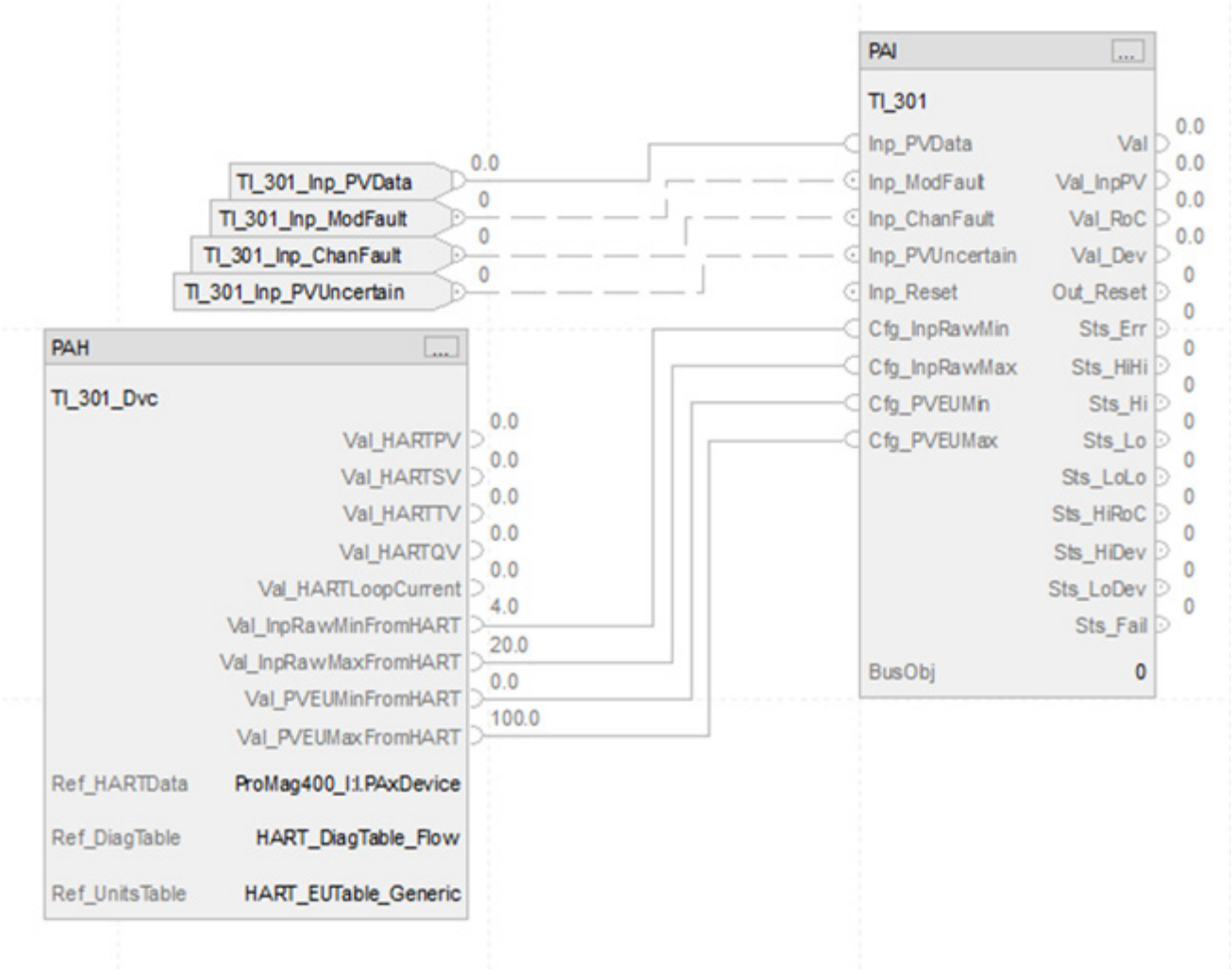
Example

In the following example, the PAH instruction is used alongside a PAI (Process Analog Input) instruction. The PAI provides processing for the analog (4 to 20 mA DC) signal from the HART analog input module, and the PAH instruction provides processing for the digital HART data overlaid upon the analog signal, received from the same field device. As a result, an analog real-time signal is provided for closed-loop control (not shown), and four digital dynamic variables are provided for additional process monitoring.

Ladder Diagram



Function Block Diagram



Structured Text

```
PAH(TI_301_Dvc, ProMag400_I:I.PAxDevice, HART_DiagTable_Flow,
HART_EUTable_Generic);

TI_301.Cfg_InpRawMin:=TI_301_Dvc.Val_InpRawMinFromHART;

TI_301.Cfg_InpRawMax:=TI_301_Dvc.Val_InpRawMaxFromHART;

TI_301.Cfg_PVEUMin:=TI_301_Dvc.Val_PVEUMinFromHART;

TI_301.Cfg_PVEUMax:=TI_301_Dvc.Val_PVEUMaxFromHART;

TI_301.Inp_PVData:=TI_301_Inp_PVData;

TI_301.Inp_ModFault:=TI_301_Inp_ModFault;

TI_301.Inp_ChanFault:=TI_301_Inp_ChanFault;

TI_301.Inp_PVUncertain:=TI_301_Inp_PVUncertain;
```

PAI(TI_301, 0);

See also

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Process Analog Input (PAI)

This information applies to the ControlLogix 5380P and 5580P controllers.

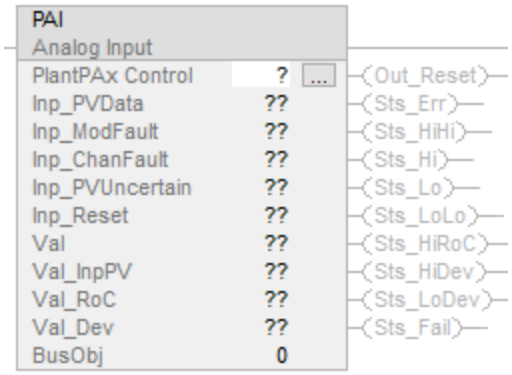
The Process Analog Input (PAI) instruction monitors an analog input and checks for alarm conditions. Use the PAI instruction to process a signal from a channel of an analog input module. Use the PAI instruction with any analog (REAL) signal.

The PAI instruction:

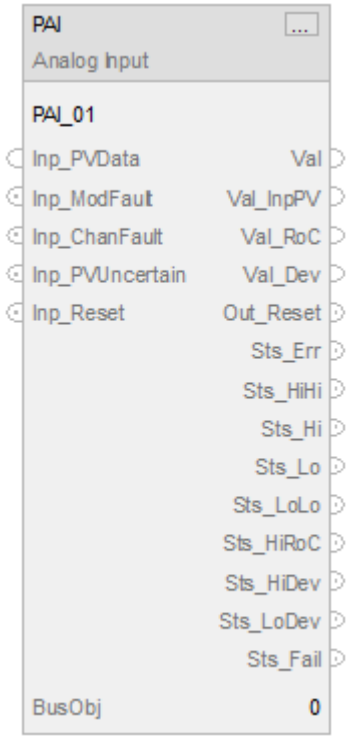
- Monitors one analog input channel.
- Scales the input Process Value (PV) from raw, input card units to engineering display units.
- Filters PV to reduce signal noise.
- Monitors PV source, PV quality and PV out-of-range condition.
- Overrides input PV in maintenance.
- Supports virtual PV for use in instruction testing, demonstration, or operator training.
- Calculates the PV deviation from reference, or setpoint, value.
- Calculates the PV rate of change.
- Captures Min and Max PV excursion values.
- Triggers alarms on PV failure, PV level, and PV deviation from the reference and PV rate of change.

Available Languages

Ladder Diagram



Function Block Diagram



Structured Text

PAI(PAI tag, BusObj);

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PlantPAx Control	P_ANALOG_INPUT	tag	Data structure required for proper operation of the instruction.

P_ANALOG_INPUT Structure

Public members are standard, visible tag members that are programmatically accessible. Private, or hidden, members are used in HMI faceplates and are not programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	Description
EnableIn	BOOL	Enable input. Ladder Diagram: Corresponds to the rung condition. Default is true.
Inp_InitializeReq	BOOL	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request to re-initialize. The instruction clears this operand automatically. Default is true.
Inp_PVData	REAL	PV signal from sensor or input (PV units). Valid = any float. Default is 4.0.
Inp_SmartDvcSts	DINT	Current code provided by SMART device on Inp_PVData. The code is copied to Out_SmartDvcSts allowing a user to monitor the device status on HMI for diagnostic lookup purposes. Valid = 0 to maximum positive number. Default is 0.
Inp_SmartDvcDiagAvailable	BOOL	1 = SMART Device diagnostics is available. Typically used to indicate device requires action to keep operating as expected. Default is false.
Inp_ModFault	BOOL	1 = I/O module failure or module communication status bad, 0 = OK. Default is false.
Inp_ChanFault	BOOL	1 = I/O channel fault or failure, 0 = OK. Default is false.
Inp_OutOfSpec	BOOL	1 = PV out of specification (PV uncertain, from device). Default is false.
Inp_FuncCheck	BOOL	1 = Function check (PV substituted, from device). Default is false.
Inp_MaintReqd	BOOL	1 = Maintenance required (from device). Default is false.

Public Input Members	Data Type	Description
Inp_PVUncertain	BOOL	Indicates the channel data accuracy is undetermined. 1 = The channel data is uncertain. This input sets Sts_PVUncertain if not in Virtual. Default is false.
Inp_PVNotify	SINT	Related PV object alarm priority and acknowledgement status. 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged. Default is 0.
Inp_HiHiGate	BOOL	The gate input used for status detection. 1 = The corresponding analog input threshold monitoring is enabled, 0 = detection is disabled and the corresponding status output is forced off. Default is true.
Inp_HiGate	BOOL	The gate input used for status detection. 1 = The corresponding analog input threshold monitoring is enabled, 0 = detection is disabled and the corresponding status output is forced off. Default is true.
Inp_LoGate	BOOL	The gate input used for status detection. 1 = The corresponding analog input threshold monitoring is enabled, 0 = detection is disabled and the corresponding status output is forced off. Default is true.
Inp_LoLoGate	BOOL	The gate input used for status detection. 1 = The corresponding analog input threshold monitoring is enabled, 0 = detection is disabled and the corresponding status output is forced off. Default is true.
Inp_HiRoCGate	BOOL	The gate input used for status detection. 1 = The corresponding analog input threshold monitoring is enabled, 0 = detection is disabled and the corresponding status output is forced off. Default is true.
Inp_HiDevGate	BOOL	The gate input used for status detection. 1 = The corresponding analog input threshold monitoring is enabled, 0 = detection is disabled and the corresponding status output is forced off. Default is true.
Inp_LoDevGate	BOOL	The gate input used for status detection. 1 = The corresponding analog input threshold monitoring is enabled, 0 = detection is disabled and the corresponding status output is forced off. Default is true.
Inp_OoRGate	BOOL	The gate input used for status detection. 1 = The corresponding analog input threshold monitoring is enabled, 0 = detection is disabled and the corresponding status output is forced off. Default is true.
Inp_Reset	BOOL	1 = Reset shed latches and cleared alarms. Default is false.
Cfg_AllowDisable	BOOL	1 = Allow maintenance to disable alarms. Default is true.
Cfg_AllowShelve	BOOL	1 = Allow operator to shelve alarms. Default is true.

Public Input Members	Data Type	Description
Cfg_ClampSB	REAL	Clamping snap-to band, to clamp when PV gets near to limit (% of PV span). Valid = 0.0 to 100.0 percent of span. Default is 0.0.
Cfg_InpRawMin	REAL	Input (unscaled) minimum for scaling. Must be set to the range of the signal connected to the Inp_PVData (raw PV) input. The input is then scaled to the values set by Cfg_PVEUMin and Cfg_PVEUMax. Valid = any float not equal to Cfg_InpRawMax Default is 4.0.
Cfg_InpRawMax	REAL	Input (unscaled) maximum for scaling. Must be set to the range of the signal connected to the Inp_PVData (raw PV) input. The input is then scaled to the values set by Cfg_PVEUMin and Cfg_PVEUMax. Valid = any float not equal to Cfg_InpRawMin Default is 20.0.
Cfg_PVEUMin	REAL	PV (output) minimum for scaling to engineering units. Valid = any float not equal to Cfg_PVEUMax Default is 0.0.
Cfg_PVEUMax	REAL	PV (output) maximum for scaling to engineering units. Valid = any float not equal to Cfg_PVEUMin. Tip: The analog input instruction supports reverse scaling. Either the raw (Input) or engineering (Scaled) range can be reversed (maximum less than minimum). Default is 100.0.
Cfg_Ref	REAL	Reference setting for deviation alarms (engineering units). Valid = any float. Default is 0.0.
Cfg_FiltWLag	REAL	Filter cutoff frequency (radian/second). Valid = any float >= 0.0. Default is 0.0.
Cfg_FiltOrder	DINT	Filter order: 0 = no filtering, 1 = 1st order low-pass filter, 2 = 2nd order low-pass filter. Default is 0.
Cfg_RateTime	REAL	PV Rate of Change time base (seconds), 1 = /second, 60 = /minute, 3600 = /hour, 86400 = /day. Default is 1.0.
Cfg_PVHiLim	REAL	PV High clamping threshold (engineering units). Valid = any float between Cfg_PVEUMax and Cfg_PVEUMin. Default is 1.50E+38.
Cfg_PVLoLim	REAL	PV Low clamping threshold (engineering units). Valid = any float between Cfg_PVEUMax and Cfg_PVEUMin. Default is - 1.5E+38.
Cfg_PVReplaceVal	REAL	Value to use to replace PV when action = replace (engineering units). Valid = any float. Default is 0.0.
Cfg_HiHiLim	REAL	High-High status threshold (engineering units). Valid = any float. Default is 1.50E+38.
Cfg_HiHiDB	REAL	The deadband that is applied to the alarm limit. This is used to prevent a noisy signal from generating spurious alarms. Valid = any float >= 0.0. Tip: If the High-High alarm limit (Cfg_HiHiLim) is 90 and the High-High alarm deadband (Cfg_HiHiDB) is 5, the High-High alarm is generated when the output (PV filtered, Val) rises above 90 and is cleared once the output (Val) falls below 85 (90 minus 5). Default is 1.0.
Cfg_HiHiGateDly	REAL	The time (seconds) after the gate input activates before the threshold detection is enabled. Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_HiLim	REAL	High status threshold (engineering units). Valid = any float. Default is 1.50E+38.
Cfg_HiDB	REAL	The deadband that is applied to the alarm limit (engineering units). Default is 1.0.
Cfg_HiGateDly	REAL	The time (seconds) after the gate input activates before the threshold detection is enabled. Valid = 0.0 to 2147483.0 seconds. Default is 0.0.

Public Input Members	Data Type	Description
Cfg_LoLim	REAL	Low status threshold (engineering units). Valid = any float. Default is -1.5E+38.
Cfg_LoDB	REAL	The deadband that is applied to the alarm limit (engineering units). Valid = any float >= 0.0. Default is 1.0.
Cfg_LoGateDly	REAL	The time (seconds) after the gate input activates before the threshold detection is enabled. Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_LoLoLim	REAL	Low-Low status threshold (engineering units). Valid = any float. Default is -1.5E+38.
Cfg_LoLoDB	REAL	The deadband that is applied to the alarm limit (engineering units). Default is 1.0.
Cfg_LoLoGateDly	REAL	The time (seconds) after the gate input activates before the threshold detection is enabled. Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_HiRoCLim	REAL	High rate of change status threshold (engineering units). Valid = any float >= 0.0. Default is 1.50E+38.
Cfg_HiRoCDB	REAL	The deadband that is applied to the alarm limit (engineering units). Valid any float >= 0.0 and <Cfg_HiRoCLim. If Cfg_HiRoCLim=0.0 then the only valid setting is Cfg_HiRoCDB=0.0. Default is 1.0.
Cfg_HiRoCGateDly	REAL	The time (seconds) after the gate input activates before the threshold detection is enabled. Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_HiDevLim	REAL	High deviation status threshold (engineering units). Valid = any float >= 0.0. Default is 1.50E+38.
Cfg_HiDevDB	REAL	The deadband that is applied to the alarm limit (engineering units). Valid = any float >= 0.0. Default is 1.0.
Cfg_HiDevGateDly	REAL	The time (seconds) after the gate input activates before the threshold detection is enabled. Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_LoDevLim	REAL	Low Deviation status threshold (engineering units). Valid = any float <= 0.0. Default is -1.5E+38.
Cfg_LoDevDB	REAL	The deadband that is applied to the alarm limit (engineering units). Valid = any float >= 0.0. Default is 1.0.
Cfg_LoDevGateDly	REAL	The time (seconds) after the gate input activates before the threshold detection is enabled. Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_OoRHiLim	REAL	High out of range status threshold (raw units). Valid = any float. Default is 20.733334.
Cfg_OoRLoLim	REAL	Low out of range status threshold (raw units). Valid = any float. Default is 3.6666667.
Cfg_OoRDB	REAL	The deadband that is applied to the alarm limit (raw units). Valid = any float >= 0.0. Default is 0.06666667.
Cfg_OoRGateDly	REAL	The time (seconds) after the gate input activates before the threshold detection is enabled. Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_OoROnDly	REAL	The minimum time (seconds) the gated raw PV must remain above the upper (Cfg_OoRHiLim) or below the lower (Cfg_OoRLoLim) limit for the status Sts_OoR to be set. On-delay time is used to avoid unnecessary alarm when the raw PV only briefly overshoots Cfg_OoRHiLim or undershoots Cfg_OoRLoLim. Valid = 0.0 to 2147483.0 seconds. Default is 0.0.

Public Input Members	Data Type	Description
Cfg_OoROffDly	REAL	The time (second) the gated raw PV must stay within each status threshold to clear the status. Off-delay time is used to reduce chattering alarm. Tip: If Cfg_OoROffDly is five seconds, the gated raw PV must be below the status limit (Cfg_OoRHiLim) minus deadband (Cfg_OoRDB) for five seconds before the status is returned to normal. Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_StuckTime	REAL	Time with no change in input to raise stuck status (second). Valid = 0.0 to 2147483.0 seconds. Default is 60.0.
Cfg_InpOoRAction	SINT	PV action on out of range: 1 = Pass input PV through unchanged, 2 = Hold last good PV value, 3 = Replace PV value with Cfg_PVReplaceVal. Default is 1.
Cfg_InpOoRQual	SINT	Out of range flag as: 1 = Good, 2 = Uncertain, 3 = Bad. Default is 3.
Cfg_InpStuckAction	SINT	PV action on stuck (unchanging): 1 = Pass input PV through unchanged, 2 = Hold last good PV value, 3 = Replace PV value with Cfg_PVReplaceVal. Default is 1.
Cfg_InpStuckQual	SINT	Stuck (unchanging) flag as: 1 = Good, 2 = Uncertain, 3 = Bad. Default is 1.
Cfg_InpNaNAction	SINT	PV action on not a number: 1 = Pass input PV through unchanged, 2 = Hold last good PV value, 3 = Replace PV value with Cfg_PVReplaceVal. Default is 2.
Cfg_InpNaNQual	SINT	PV not a number flag as: 1 = Good, 2 = Uncertain, 3 = Bad. Default is 3.
Cfg_ModFaultAction	SINT	PV action on I/O module fault: 1 = Pass input PV through unchanged, 2 = Hold last good PV value, 3 = Replace PV value with Cfg_PVReplaceVal. Default is 2.
Cfg_ModFaultQual	SINT	I/O module fault Flag as: 1 = Good, 2= Uncertain, 3 = Bad. Default is 3.
Cfg_ChanFaultAction	SINT	PV action on channel fault: 1 = Pass input PV through unchanged, 2 = Hold last good PV value, 3 = Replace PV value with Cfg_PVReplaceVal. Default is 2.

Public Input Members	Data Type	Description
Cfg_ChanFaultQual	SINT	I/O channel fault flag as: 1 = Good, 2 = Uncertain, 3 = Bad. Default is 3.
Cfg_OutOfSpecAction	SINT	PV action on out of spec (from device): 1 = Pass input PV through unchanged, 2 = Hold last good PV value, 3 = Replace PV value with Cfg_PVReplaceVal. Default is 1.
Cfg_OutOfSpecQual	SINT	Inp_PVUncertain flag as: 1 = Good, 2 = Uncertain, 3 = Bad. Default is 2.
Cfg_FuncCheckAction	SINT	PV action on function check (from device): 1 = Pass input PV through unchanged, 2 = Hold last good PV value, 3 = Replace PV value with Cfg_PVReplaceVal. Default is 3.
Cfg_FuncCheckQual	SINT	Function check flag as: 1 = Good, 2 = Uncertain, 3 = Bad. Default is 3.
Cfg_MaintReqdAction	SINT	PV action on maintenance required (from device): 1 = Pass input PV through unchanged, 2 = Hold last good PV value, 3 = Replace PV value with Cfg_PVReplaceVal. Default is 3.
Cfg_MaintReqdQual	SINT	Maintenance required flag as: 1 = Good, 2 = Uncertain, 3 = Bad. Default is 3.
Cfg_CfgErrAction	SINT	PV action on Instruction configuration error: 1 = Pass input PV through unchanged, 2 = Hold last good PV value, 3 = Replace PV value with Cfg_PVReplaceVal. Default is 3.
Cfg_CfgErrQual	SINT	Instruction configuration error flag as: 1 = Good, 2 = Uncertain, 3 = Bad. Default is 3.
Cfg_CtrlHiHiLim	REAL	Current high-high control threshold (engineering units). Valid = any float. Default is 1.50E+38.
Cfg_CtrlHiHiDB	REAL	High-High control deadband (engineering units). Valid = any float >= 0.0. Default is 1.0.
Cfg_CtrlHiLim	REAL	Current high control threshold (engineering units). Valid = any float. Default is 1.50E+38.
Cfg_CtrlHiDB	REAL	High control deadband (engineering units). Valid = any float >= 0.0. Default is 1.0.

Public Input Members	Data Type	Description
Cfg_CtrlLoLim	REAL	Current low control threshold (engineering units). Valid = any float. Default is -1.5E+38.
Cfg_CtrlLoDB	REAL	Low control deadband (engineering units). Valid = any float >= 0.0. Default is 1.0.
Cfg_CtrlLoLoLim	REAL	Current low-low control threshold (engineering units). Valid = any float. Default is -1.5E+38.
Cfg_CtrlLoLoDB	REAL	Low-Low control deadband (engineering units). Valid = any float >= 0.0. Default is 1.0.
Cfg_HasSmartDvc	BOOL	1 = Enable a button on the HMI that could be used to call up a SMART Device faceplate (diagnostics). Default is false.
Cfg_HasRoC	BOOL	1 = PV rate of change made visible in HMI. Default is false.
Cfg_HasDev	BOOL	1 = PV Deviation made visible in HMI. Default is false.
Cfg_HasMoreObj	BOOL	1 = Tells HMI an object with more info is available. Default is false.
Cfg_HasOutNav	BOOL	1 = Tells HMI to enable navigation to a connected output object, 0 = No connected output object. Default is false.
Cfg_HasPVNav	BOOL	1 = Tells HMI to enable navigation to a connected process variable object. Default is false.
Cfg_HasHistTrend	SINT	Has Historical Trend. This enables navigation to the Device Historical Trend Faceplate from the HMI. 0 = No external historical trend, 1 = Datalog historical trend, 2 = Historian historical trend. Default is 0.
Cfg_FailOnUncertain	BOOL	1 = Raise Sts_Fail (and fail alarm) if PV quality is uncertain, 0 = Raise Sts_Fail (and fail alarm) only if PV quality is bad (scaling configuration error, PV is NaN or Inf, I/O fault or raw PV is out of range. Default is false.
Cfg_NoSubstPV	BOOL	Disables the maintenance substitution feature. 0 = The Substitute PV Maintenance function is enabled, 1 = The Substitute PV Maintenance function is disabled. When Cfg_NoSubstPV is 0, the commands MCmd_SubstPV and MCmd_InpPV are used to select the input PV or the substitute PV. Sts_SubstPV is set to 1 when the substitute PV is selected. Default is false.
Cfg_SetTrack	BOOL	1 = Set_VirtualPV tracks Val_InpPV in virtual. MSet_SubstPV tracks Val_InpPV when substitution is not active. 0 = No tracking. Default is true.
Cfg_SclngTyp	SINT	Scaling Type 0 = none, 1 = Linear, 2 = Square Root. Default is 1.
Cfg_PVDecPlcs	SINT	Number of decimal places for process variable display. Valid = 0 to 6. Default is 2.

Public Input Members	Data Type	Description
Cfg_CnfrmReqd	SINT	Operator command confirmation required. Represents the type of command confirmation required. 0 = None, 1 = Command confirmation required, 2 = Performer e-signature required, 3 = Performer and approver e-signature required. Default is 0.
PSet_Owner	DINT	Program owner request ID (non-zero) or release (zero). Default is 0.
Set_VirtualPV	REAL	PV used in virtual (Sts_Virtual = 1) (engineering units). Default is 0.0.
PCmd_ClearCapt	BOOL	Set PCmd_ClearCapt to 1 to clear the captured minimum/maximum PV excursion values. The instruction clears this operand automatically. Default is false.
PCmd_Reset	BOOL	Program command to reset all alarms and latched shed conditions requiring reset. The instruction clears this operand automatically. Default is false.
PCmd_Physical	BOOL	Program command to select Physical device operation (not simulated). The instruction clears this operand automatically. Default is false.
PCmd_Virtual	BOOL	Program command to select Virtual (simulated) device operation. The instruction clears this operand automatically. Default is false.
XCmd_ClearCapt	BOOL	External command to clear the captured minimum/maximum PV excursion values. The instruction clears this operand automatically. Default is false.
XCmd_Reset	BOOL	External command to reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.
XCmd_ResetAckAll	BOOL	External command to acknowledge and reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.

Public Output Members	Data Type	Description
EnableOut	BOOL	This output state always reflects EnableIn input state.
Val	REAL	Analog input value in engineering units (after Substitute PV, if used). Extended Properties of this member: Units - Engineering units (text) used for the analog input.
Val_InpPV	REAL	Analog input value in engineering units (actual, before Substitute PV selection).
Val_RoC	REAL	Analog value Rate of Change (engineering units/rate time).
Val_Dev	REAL	Calculated deviation from reference (engineering units).
Val_PVMinCapt	REAL	Captured PV minimum (excursion) since last cleared (engineering units). Default is 1.5E+38.
Val_PVMaxCapt	REAL	Captured PV maximum (excursion) since last cleared (engineering units). Default is -1.5E+38.
Val_PVEUMin	REAL	Minimum of scaled range = MIN (Cfg_PVEUMin, Cfg_PVEUMax).
Val_PVEUMax	REAL	Maximum of scaled range = MAX (Cfg_PVEUMin, Cfg_PVEUMax).
Out_SmartDvcSts	DINT	Status code of a SMART Device provided by Inp_SmartDvcSts. Out_SmartDvcSts is a copy of Inp_SmartDvcSts.
Sts_Initialized	BOOL	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.

Public Output Members	Data Type	Description
Sts_SmartDvcDiagAvailable	BOOL	1 = Diagnostics of a SMART Device is currently available. Typically used to indicate device requires action to keep operating as expected. Sts_SmartDvcDiagAvailable is a copy of Inp_SmartDvcDiagAvailable.
Sts_PVGood	BOOL	1 = PV quality is Good (not flagged as Bad or Uncertain).
Sts_PVUncertain	BOOL	Indicates the channel data accuracy is undetermined. 1 = The channel data is uncertain. This output is set by Inp_PVUncertain (if not in Virtual).
Sts_PVBad	BOOL	1 = PV quality is flagged as Bad.
Sts_InpStuck	BOOL	1 = Input is stuck (unchanging).
Sts_InpNaN	BOOL	1 = Input is not a number (floating point exception).
Sts_OutOfSpec	BOOL	1 = Working outside specifications (from device).
Sts_FuncCheck	BOOL	1 = Function check (PV simulated/replaced at device).
Sts_MaintReqd	BOOL	1 = Maintenance is required (from device).
Sts_UseInp	BOOL	1 = Using input to calculate PV (not replaced or held).
Sts_HoldLast	BOOL	1 = Analog PV being held at last good value.
Sts_Clamped	BOOL	1 = Analog PV being clamped at Low or High Limit.
Sts_Replaced	BOOL	1 = Analog PV being replaced with configured value.
Sts_SubstPV	BOOL	1 = Using substitute PV (Override).
Sts_InpPV	BOOL	1 = Using input PV (Normal).
Sts_Virtual	BOOL	1 = Using virtual PV instead of the input from the device (Inp_PVData) to calculate output. 0 = The instruction uses input operand Inp_PVData to calculate output. Sts_Virtual is a copy of Inp_Virtual.
SrcQ_IO	SINT	Source and quality of primary input or output (enumerated): 0 = Good, live, confirmed good, 1 = Good, live, assumed good, 2 = Good, no feedback, assumed good, 8 = Test, virtualized, 9 = Test, loopback, 10 = Test, manually entered, 16 = Uncertain, live, off-spec, 17 = Uncertain, substituted at device or bus, 18 = Uncertain, substituted at instruction, 19 = Uncertain, using last known good, 20 = Uncertain, using replacement value, 32 = Bad, signal failure, 33 = Bad, channel fault, 34 = Bad, module or communication fault, 35 = Bad, invalid configuration.

Public Output Members	Data Type	Description
SrcQ	SINT	Source and quality of primary value or status (enumerated): 0 = Good, live, confirmed good, 1 = Good, live, assumed good, 2 = Good, no feedback, assumed good, 8 = Test, virtualized, 9 = Test, loopback, 10 = Test, manually entered, 16 = Uncertain, live, off-spec, 17 = Uncertain, substituted at device or bus, 18 = Uncertain, substituted at instruction, 19 = Uncertain, using last known good, 20 = Uncertain, using replacement value, 32 = Bad, signal failure, 33 = Bad, channel fault, 34 = Bad, module or communication fault, 35 = Bad, invalid configuration.
Sts_bSts	SINT	Device confirmed status: 0 = PV Good, Sts_bSts.0: PV Uncertain, Sts_bSts.1: PV Bad, Sts_bSts.2: PV Substituted. PV is Good if Sts_PVUncertain = 0 and Sts_PVBad = 0, PV is Uncertain if Sts_PVUncertain = 1, PV is Bad if Sts_PVBad = 1, PV is Substituted if Sts_SubstPV or Sts_Virtual = 1.
Sts_bFault	INT	Device fault status: 0 = None, Sts_bFault.0: Low, Sts_bFault.1: High, Sts_bFault.2: Low Deviation, Sts_bFault.3: High Deviation, Sts_bFault.4: Low Low, Sts_bFault.5: High High, Sts_bFault.6: High Rate of Change, Sts_bFault.7: Fail, Sts_bFault.8: Configuration Error.
Sts_eNotify	SINT	All alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.

Public Output Members	Data Type	Description
Sts_eNotifyAll	SINT	All alarm status enumerated values including related objects: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyHiHi	SINT	HiHi alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyHi	SINT	Hi alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyLo	SINT	Lo alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.

Public Output Members	Data Type	Description
Sts_eNotifyLoLo	SINT	LoLo alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyHiRoC	SINT	HiRoC alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyHiDev	SINT	HiDev alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyLoDev	SINT	LoDev alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.

Public Output Members	Data Type	Description
Sts_eNotifyFail	SINT	Fail alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_UnackAlmCount	DINT	Count of unacknowledged alarms.
Sts_MaintByp	BOOL	1 = The Device has a maintenance bypass function active.
Sts_Err	BOOL	1 = Error in configuration: See detail bits (Sts_Errxxx) for reason.
Sts_ErrRaw	BOOL	1 = Error in configuration: Raw input scaling Min = Max.
Sts_ErrEU	BOOL	1 = Error in configuration: Scaled EU Min = Max.
Sts_ErrFiltWLag	BOOL	1 = Error in configuration: Filter cutoff frequency.
Sts_ErrFiltOrder	BOOL	1 = Error in configuration: Filter order.
Sts_ErrRateTime	BOOL	1 = Error in configuration: PV Rate of Change time base.
Sts_ErrHiHiDB	BOOL	1 = Error in configuration: Cfg_HiHiDB deadband is < 0.0.
Sts_ErrHiHiGateDly	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrHiDB	BOOL	1 = Error in configuration: Cfg_HiDB deadband is < 0.0.
Sts_ErrHiGateDly	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrLoDB	BOOL	1 = Error in configuration: Cfg_LoDB deadband is < 0.0.
Sts_ErrLoGateDly	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrLoLoDB	BOOL	1 = Error in configuration: Cfg_LoLoDB deadband is < 0.0.
Sts_ErrLoLoGateDly	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrHiRoCDB	BOOL	1 = Error in configuration: Cfg_HiRoCDB deadband is invalid.
Sts_ErrHiRoCGateDly	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrHiDevDB	BOOL	1 = Error in configuration: Cfg_HiDevDB deadband is < 0.0.
Sts_ErrHiDevGateDly	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrLoDevDB	BOOL	1 = Error in configuration: Cfg_LoDevDB deadband is < 0.0.
Sts_ErrLoDevGateDly	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrOoRDB	BOOL	1 = Error in configuration: Cfg_OoRDB deadband is < 0.0.
Sts_ErrOoRGateDly	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrOoROnDly	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrOoROffDly	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrStuckTime	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrCmdCnfrmTimeOutTime	BOOL	1 = Error in configuration: Command confirmation timer preset (use 0.0 to 2147483.0).
Sts_ErrAlm	BOOL	1 = Error in tag-based alarm settings.
Sts_Alm	BOOL	1 = An alarm is active.
Sts_AlmInh	BOOL	1 = An alarm is shelved or disabled.
Sts_IOFault	BOOL	IO Fault status is set to 1 if there is a Module fault (Inp_ModFault = 1) or Channel fault (Inp_ChanFault = 1) and PV is not virtual.
Sts_HiHiCmp	BOOL	PV comparison result, 1 = High-High.
Sts_HiHiGate	BOOL	PV High-High gate delay status, 1 = done.

Public Output Members	Data Type	Description
Sts_HiHi	BOOL	1 = Analog input is above High-High limit. There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PAITag.@Alarms.Alm_HiHi.AlarmElement
Sts_HiCmp	BOOL	PV comparison result 1 = High.
Sts_HiGate	BOOL	PV High gate delay status, 1 = done.
Sts_Hi	BOOL	1 = Analog input is above High limit. There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PAITag.@Alarms.Alm_Hi.AlarmElement
Sts_LoCmp	BOOL	PV comparison result 1 = Low.
Sts_LoGate	BOOL	PV Low gate delay status, 1 = done.
Sts_Lo	BOOL	1 = Analog input is below Low limit. There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PAITag.@Alarms.Alm_Lo.AlarmElement
Sts_LoLoCmp	BOOL	PV comparison result 1 = Low-Low.
Sts_LoLoGate	BOOL	PV Low-Low gate delay, status 1 = done.
Sts_LoLo	BOOL	1 = Analog input is below Low-Low limit. There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PAITag.@Alarms.Alm_LoLo.AlarmElement
Sts_HiRoCCmp	BOOL	PV comparison result 1 = High Rate of Change.
Sts_HiRoCGate	BOOL	PV High Rate of Change gate delay status, 1 = done.
Sts_HiRoC	BOOL	1 = Analog input Rate of Change is above High limit. There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PAITag.@Alarms.Alm_HiRoC.AlarmElement
Sts_HiDevCmp	BOOL	PV comparison result 1 = High Deviation.
Sts_HiDevGate	BOOL	PV High Deviation gate delay, status 1 = done.
Sts_HiDev	BOOL	1 = Analog input Deviation is above High limit. There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PAITag.@Alarms.Alm_HiDev.AlarmElement
Sts_LoDevCmp	BOOL	PV comparison result, 1 = Low Deviation.
Sts_LoDevGate	BOOL	PV Low Deviation delay status, 1 = done.
Sts_LoDev	BOOL	1 = Analog input Deviation is below Low limit. There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PAITag.@Alarms.Alm_LoDev.AlarmElement
Sts_OoRHiCmp	BOOL	PV comparison result, 1 = High Out of Range.
Sts_OoRLoCmp	BOOL	PV comparison result, 1 = Low Out of Range.
Sts_OoRCmp	BOOL	PV comparison result, 1 = Out of Range.
Sts_OoRGate	BOOL	PV Out of Range gate delay status, 1 = done.

Public Output Members	Data Type	Description
Sts_OoR	BOOL	1 = Analog raw input is above High raw limit or below Low raw limit.
Sts_Fail	BOOL	1 = Analog input failed. At least one of the following conditions holds: PV scaling configuration error, raw PV is out of range or not a number, input module or input channel fault, device reports PV uncertain (if configured for). There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PAITag.@Alarms.Alm_Fail.AlarmElement
Sts_CnfrmOperCmdReq	BOOL	1 = Operator command request is awaiting confirmation.
Sts_RdyAck	BOOL	1 = An alarm is ready to be acknowledged.
Sts_RdyReset	BOOL	1 = A latched alarm or shed condition is ready to be reset.
XRdy_ClearCapt	BOOL	1 = Ready for XCmd_ClearCapt, enable HMI button.
XRdy_Reset	BOOL	1 = Ready for XCmd_Reset, enable HMI button.
XRdy_ResetAckAll	BOOL	1 = Ready for XCmd_ResetAckAll, enable HMI button.
Val_Owner	DINT	Current object owner ID (0 = not owned).

Private Input Members	Data Type	Description
MSet_SubstPV	REAL	Maintenance-entered substitute PV in engineering units that overrides input PV when MCmd_SubstPV is 1. If not using the substitute (MCmd_SubstPV is false), the MSet_SubstPV setting tracks the Out value for bumpless transfer from input PV to substitute PV. Default = 0.0.
MCmd_SubstPV	BOOL	Maintenance command to use Substitute PV. The instruction clears this operand automatically. Default is false.
MCmd_InpPV	BOOL	Maintenance command to use Input PV (normal). The instruction clears this operand automatically. Default is false.
OCmd_ClearCapt	BOOL	Operator command to clear the captured minimum/maximum PV excursion values. The instruction clears this operand automatically. Default is false.
OCmd_Reset	BOOL	Operator command to reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.
OCmd_ResetAckAll	BOOL	Operator command to acknowledge and reset all alarms and latched shed conditions. The use of OCmd_ResetAckAll is restricted to HMI. The instruction clears this operand automatically. Default is false.
OCmd_CmdCncl	BOOL	Operator command to cancel command request. The instruction clears this operand automatically. Default is false.
OCmd_CmdCnfrm	BOOL	Operator command to confirm command request. The instruction clears this operand automatically. Default is false.

Private Output Members	Data Type	Description
MRdy_SubstPV	BOOL	1 = Ready for MCmd_SubstPV.
MRdy_InpPV	BOOL	1 = The instruction is ready for MCmd_InpPV command.
ORdy_ClearCapt	BOOL	1 = Ready for OCmd_ClearCapt, enable HMI button.
ORdy_Reset	BOOL	1 = A latched alarm or shed condition is ready to be reset.
ORdy_ResetAckAll	BOOL	1 = A latched alarm or shed condition is ready to be reset or acknowledged.

Alarms

Discrete tag-based alarms are defined for these members.

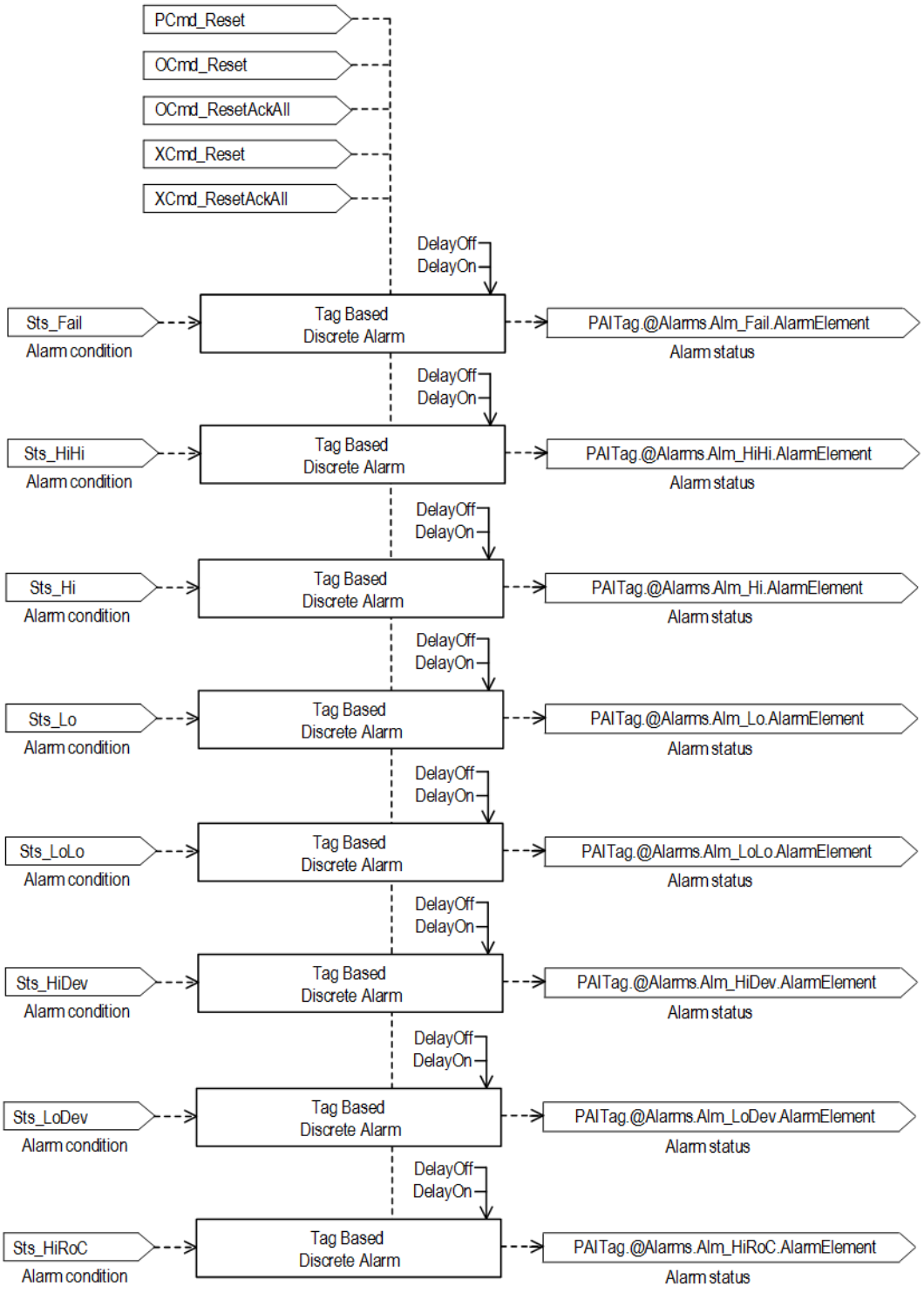
Member	Alarm Name	Description
Sts_Fail	Alm_Fail	Raised when any of the following is true: • The PV quality is bad. The PV quality is bad if either Inp_ChanFault or Inp_ModFault input is 1. The PV bad quality check is skipped in Virtual. • The Inp_PVUncertain input is true and the instruction is configured for PV uncertain status taking effect on failure. The PV uncertain check is skipped in Virtual. • The PV is outside the configured failure limits. • The PV is infinite or not a number (floating-point exception). • The raw or engineering unit range configuration used in scaling is invalid.
Sts_HiHi	Alm_HiHi	Raised when the PV is above the High-High threshold and the associated gate is opened (Inp_HiHiGate = 1). The threshold, deadband, and gate delay are set in alarm configuration.
Sts_Hi	Alm_Hi	Raised when the PV is above the High threshold and the associated gate is opened (Inp_HiGate = 1). The threshold, deadband, and gate delay are set in alarm configuration.
Sts_Lo	Alm_Lo	Raised when the PV is below the Low threshold and the associated gate is opened (Inp_LoGate = 1). The threshold, deadband, and gate delay are set in alarm configuration.
Sts_LoLo	Alm_LoLo	Raised when the PV is below the Low-Low threshold and the associated gate is opened (Inp_LoLoGate = 1). The threshold, deadband, and gate delay are set in alarm configuration.
Sts_HiDev	Alm_HiDev	Raised when the amount by which the PV exceeds the setpoint or reference is above the High Deviation threshold while the associated gate is opened (Inp_HiDevGate = 1). The threshold, deadband, and gate delay are set in alarm configuration.
Sts_LoDev	Alm_LoDev	Raised when the amount by which the PV exceeds the setpoint or reference is below the Low Deviation threshold while the associated gate is opened (Inp_LoDevGate = 1). Since the threshold is a negative number, this is the amount the PV falls below the setpoint or reference. The threshold, deadband, and gate delay are set in alarm configuration.
Sts_HiRoC	Alm_HiRoC	Raised when the amount by which the absolute value of PV rate of change exceeds High Rate of Change limit while the associated gate is opened (Inp_HiRoCGate = 1). The threshold, deadband, and gate delay are set in alarm configuration.

Mark the alarm as used or unused and set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format:

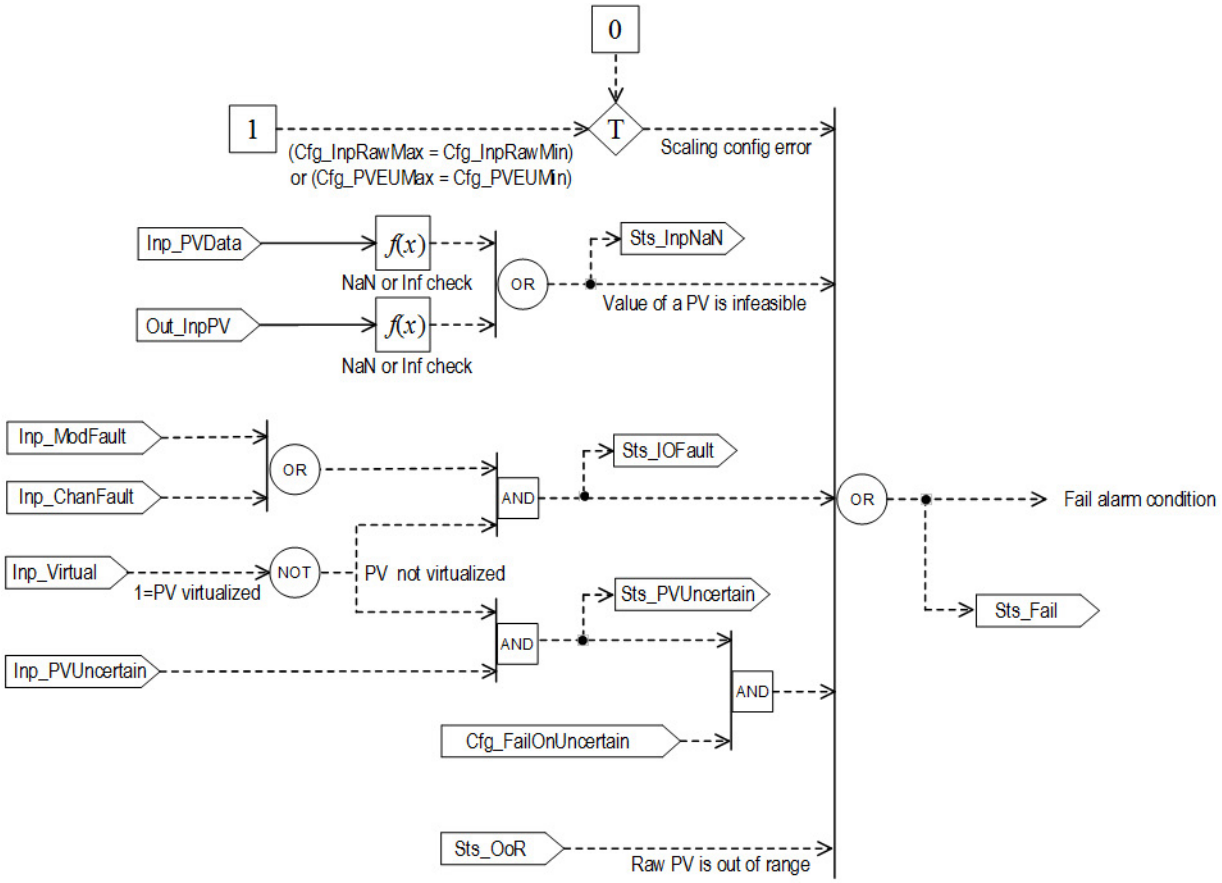
PAITag.@Alarms.AlarmName.AlarmElement

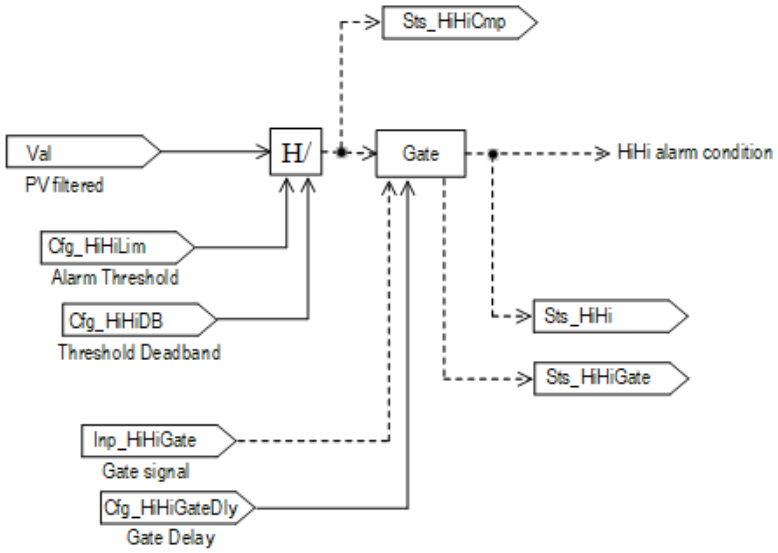
- There is a Program command that enables to Reset all alarms of the instruction (Alarm Set) at the same time.
- There are Operator commands that enable to Reset, and Reset&Acknowledge all alarms of the instruction (Alarm Set) at the same time.

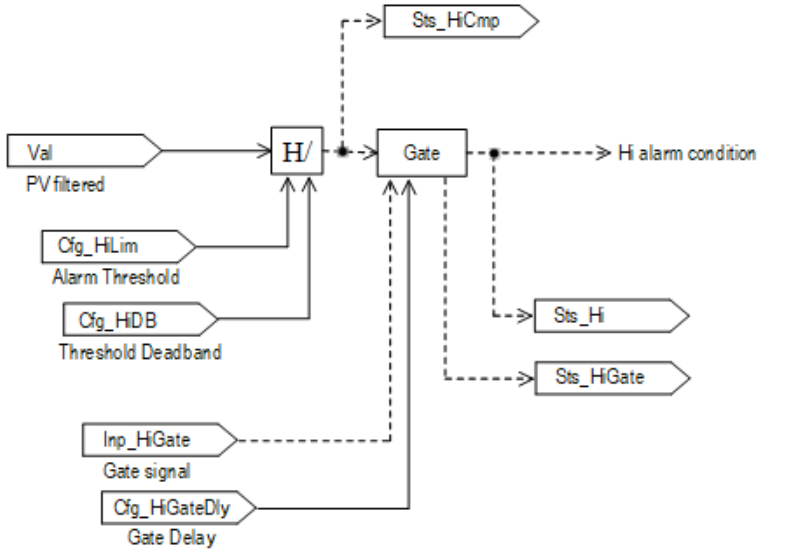
- There are External commands that enable to Reset, and Reset&Acknowledge all alarms of the instruction (Alarm Set) at the same time.

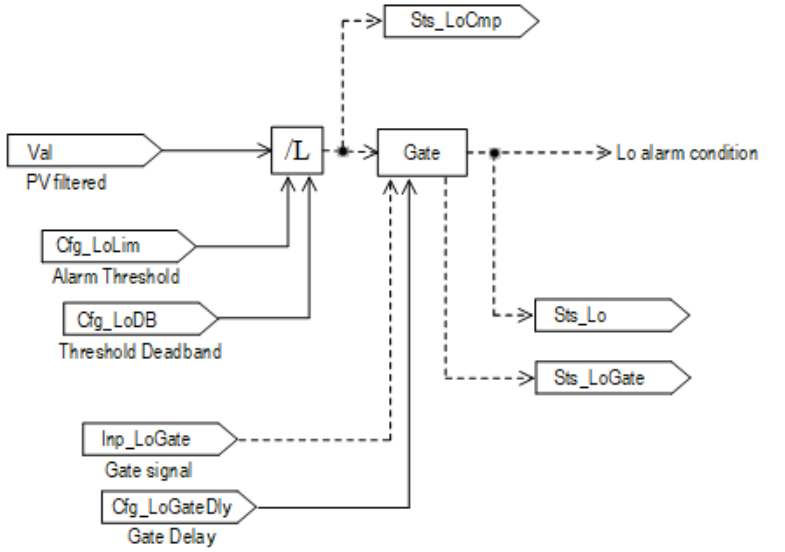


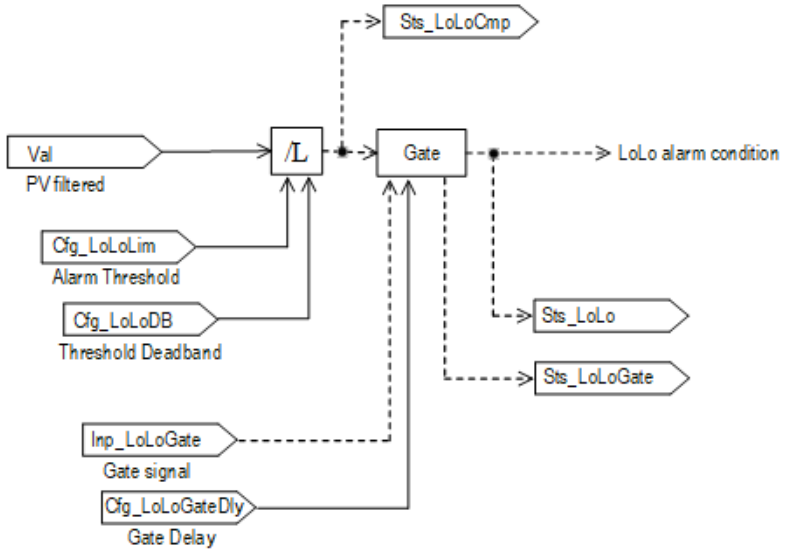
These diagrams show how Fail, High High, High, Low, Low Low, High Deviation, Low Deviation and High Rate of Change alarm conditions are calculated in the PAI instruction.

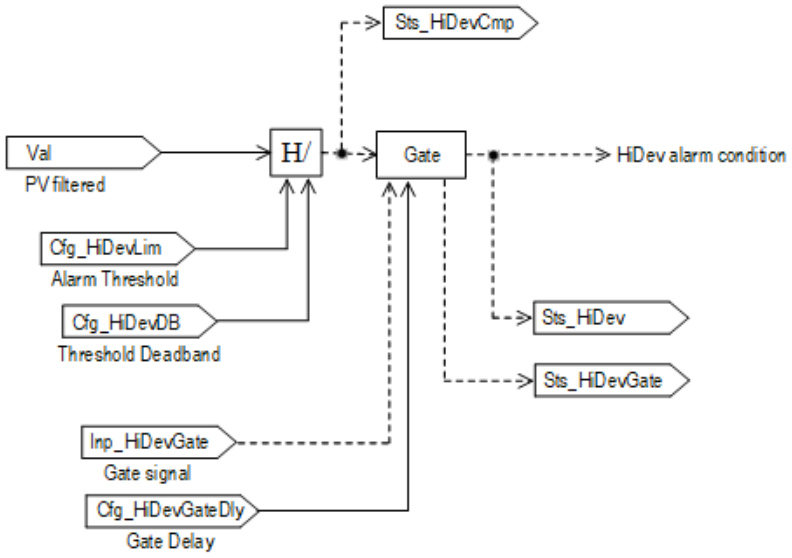

0
1
T
(Cfg_InpRawMax = Cfg_InpRawMin)
or (Cfg_PVEUMax = Cfg_PVEUMin)
Scaling config error
Inp_PVData
f(x)
NaN or Inf check
Out_InpPV
f(x)
NaN or Inf check
OR
Sts_InpNaN
Value of a PV is infeasible
Inp_ModFault
Inp_ChanFault
OR
AND
Sts_IOFault
Inp_Virtual
NOT
1=PV virtualized
PV not virtualized
AND
Sts_PVUncertain
Inp_PVUncertain
AND
Cfg_FailOnUncertain
OR
Fail alarm condition
Sts_Fail
Sts_OoR
Raw PV is out of range

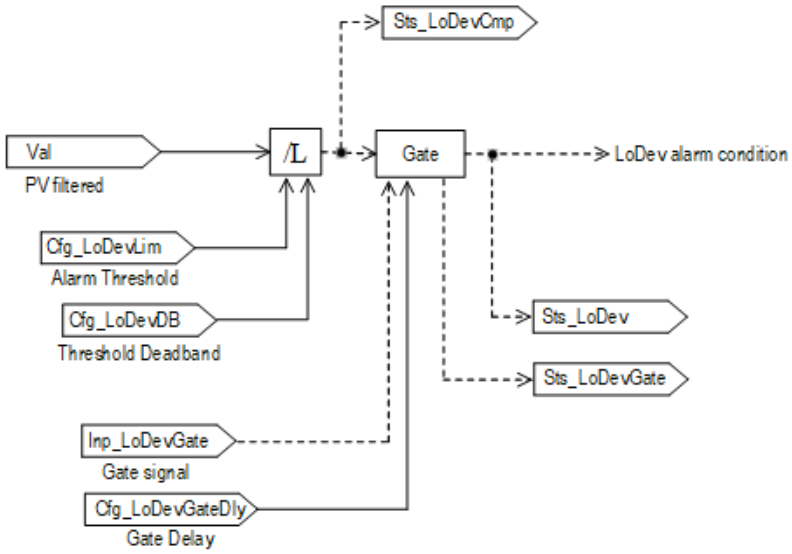

Sts_HiHiCmp
Val
PV filtered
H/
Gate
HiHi alarm condition
Cfg_HiHiLim
Alarm Threshold
Cfg_HiHiDB
Threshold Deadband
Sts_HiHi
Sts_HiHiGate
Inp_HiHiGate
Gate signal
Cfg_HiHiGateDly
Gate Delay

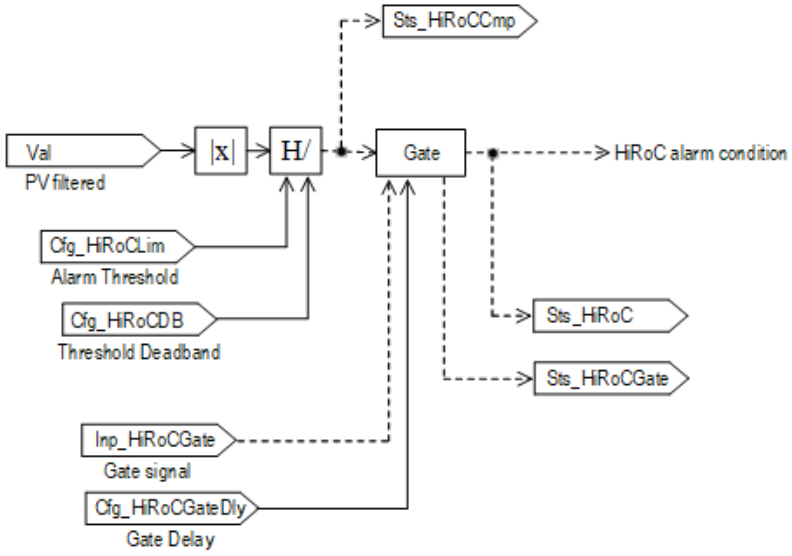

Sts_HiCmp
Val
PV filtered
H/
Gate
Hi alarm condition
Cfg_HiLim
Alarm Threshold
Cfg_HiDB
Threshold Deadband
Sts_Hi
Sts_HiGate
Inp_HiGate
Gate signal
Cfg_HiGateDly
Gate Delay


Sts_LoCmp
Val
PV filtered
/L
Gate
Lo alarm condition
Cfg_LoLim
Alarm Threshold
Cfg_LoDB
Threshold Deadband
Sts_Lo
Sts_LoGate
Inp_LoGate
Gate signal
Cfg_LoGateDly
Gate Delay


Sts_LoLoCmp
Val
PV filtered
/L
Gate
LoLo alarm condition
Cfg_LoLoLim
Alarm Threshold
Cfg_LoLoDB
Threshold Deadband
Sts_LoLo
Sts_LoLoGate
Inp_LoLoGate
Gate signal
Cfg_LoLoGateDly
Gate Delay


Sts_HiDevCmp
Val
PV filtered
H/
Gate
HiDev alarm condition
Cfg_HiDevLim
Alarm Threshold
Cfg_HiDevDB
Threshold Deadband
Sts_HiDev
Sts_HiDevGate
Inp_HiDevGate
Gate signal
Cfg_HiDevGateDly
Gate Delay


Sts_LoDevCmp
Val
PV filtered
/L
Gate
LoDev alarm condition
Cfg_LoDevLim
Alarm Threshold
Cfg_LoDevDB
Threshold Deadband
Sts_LoDev
Sts_LoDevGate
Inp_LoDevGate
Gate signal
Cfg_LoDevGateDly
Gate Delay



Operation

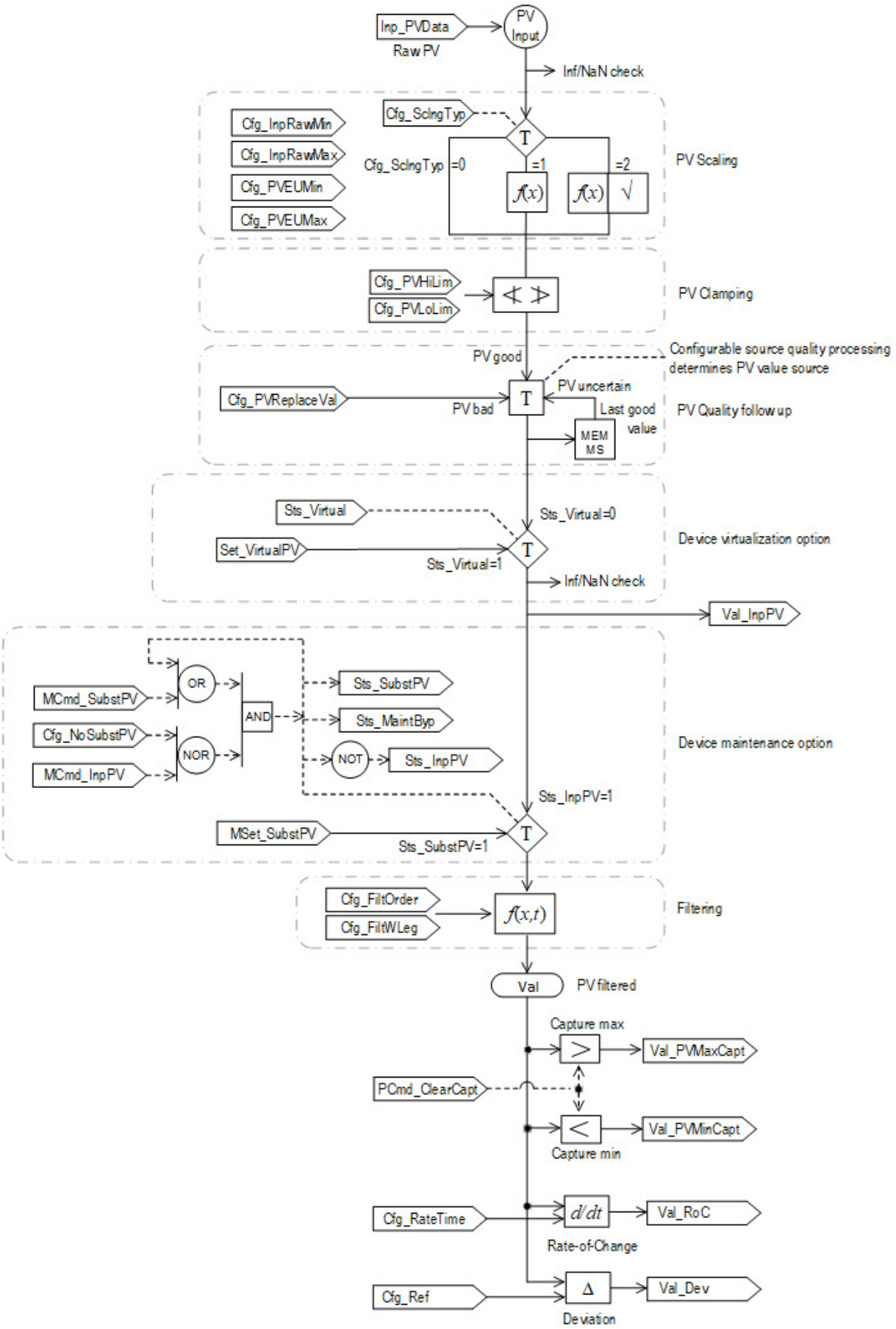
The PAI instruction:

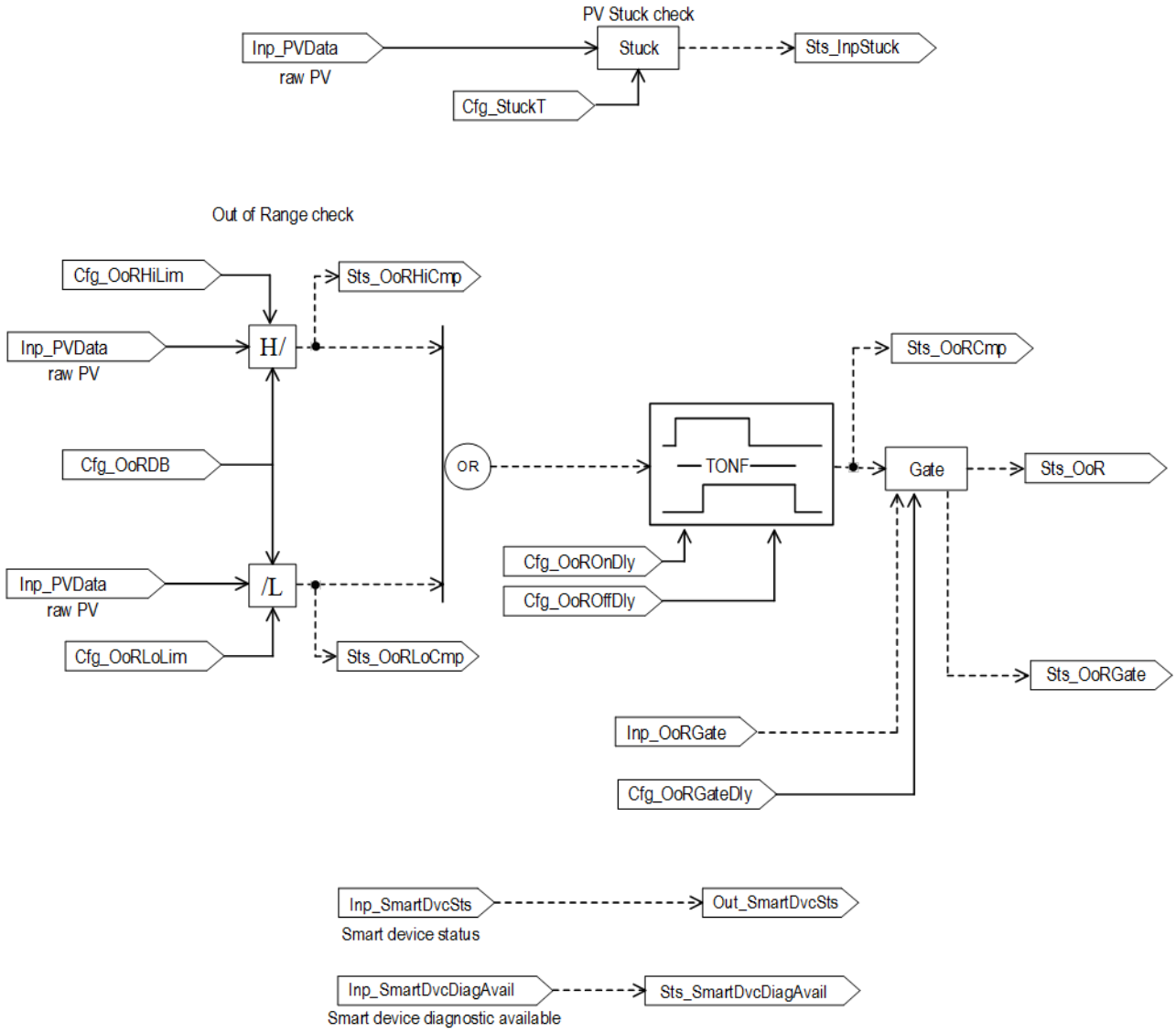
- Monitors one analog input channel for the following conditions:
  - Invalid configuration of the instruction (scaling configuration error)
  - I/O channel fault
  - I/O module fault
  - Input not-a-number (floating-point exception)
  - Raw input out of range
  - Input stuck (unchanging)
  - Out of specification (uncertain) – reported from the device
  - Function check (substitute PV entered manually) - reported from the device
  - Maintenance required - reported from the device
- For each condition, takes these actions:
  - Pass the PV through unchanged
  - Use the last good PV value
  - Apply a configured replacement PV value
- Scales the input value from raw (input card) units to engineering (display) units:
  - Linear scaling (optional)
  - Square root characterized scaling (optional).

    Tip: Square root characterized scaling is typically used with orifice plates or other pressure-differential elements for flow measurement when the transmitter does not provide square root characterization. The square root scaling in the instruction works with ± pressure differential to provide positive or negative flow values.
- Filters PV (optional) to reduce signal noise.
- Monitors PV Source, PV Quality and PV out-of-range condition.

- Supports maintenance selection of the substitute PV function to allow manual override of the input PV.
- Supports virtual PV for use in instruction testing, demonstration, or operator training.
- Provides entry of a reference (setpoint) value and calculates PV deviation from the reference value.
- Calculates the PV rate of change (RoC).
- Captures Min and Max PV excursion values.

These diagrams illustrate the functionality of the PAI instruction:


Inp_PVData
Raw PV
PV Input
Inf/NaN check
Cfg_SclngTyp
Cfg_InpRawMin
Cfg_InpRawMax
Cfg_PVEUMin
Cfg_PVEUMax
Cfg_SclngTyp =0
=1
=2
PV Scaling
Cfg_PVHiLim
Cfg_PVLoLim
PV Clamping
PV good
Configurable source quality processing determines PV value source
Cfg_PVReplaceVal
PV bad
PV uncertain
Last good value
PV Quality follow up
MEM MS
Sts_Virtual
Set_VirtualPV
Sts_Virtual=0
Sts_Virtual=1
Inf/NaN check
Device virtualization option
Val_InpPV
MCmd_SubstPV
Cfg_NoSubstPV
MCmd_InpPV
OR
NOR
AND
NOT
Sts_SubstPV
Sts_MaintByp
Sts_InpPV
Sts_InpPV=1
MSet_SubstPV
Sts_SubstPV=1
Device maintenance option
Cfg_FiltOrder
Cfg_FiltWLeg
Filtering
Val
PV filtered
Capture max
Val_PVMaxCapt
PCmd_ClearCapt
Val_PVMinCapt
Capture min
Cfg_RateTime
Val_RoC
Rate-of-Change
Cfg_Ref
Val_Dev
Deviation



Virtualization

Use virtualization for instruction testing and operator training. Set the Inp_Virtual operand to 1 to enable virtualization. After finishing virtualization, set the Inp_Virtual operand to 0 to return to normal operation.

Virtualization enables processing the virtual input instead of normal (scaled) input PV. The instruction has operand (Set_VirtualPV) for entering virtual PV in EU. When the instruction is not in Virtual, the virtual PV setting (Set_VirtualPV) tracks the selected PV for bumpless transfer into Virtual.

Initialization

The instruction is normally initialized in the instruction first run. Re-initialization can be requested any time by setting Inp_InitializeReq = 1. For proper initialization, when adding the instruction while performing an online edit of the code, make sure that Inp_InitializeReq = 1, the default value.

Configuration of Strings for HMI

Configure strings for HMI faceplates, as seen in FactoryTalk View, and for the Logix Designer configuration dialog box. The strings are set to extended properties of tag items. Configure the strings in the Logix Designer application only.

- Description
- Label for graphic symbol
- Display Library for HMI Faceplate call-up
- Instruction name
- Area name
- URL link
- PV raw units
- PV engineering units

Monitor the PAI Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Rung-condition-out is cleared to false. The state of using raw input or maintenance substitute PV is not modified and persists through a controller powerup or PROG-to-RUN transition.
Instruction first run	All commands that are automatically cleared each execution are cleared and ignored. Filter is initialized. Internal timers are reset. The instruction executes normally.
Rung-condition-in is false	Set rung-condition-out to rung-condition-in. The instruction shows a status of IO fault (Sts_IOFault). The calculation of the scaled input PV value (Val_InpPV) is executed to indicate to the operator the actual input value, even though the primary PV (Val) is not updated (holds last value).
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	EnableOut is cleared to false. The state of using raw input or maintenance substitute PV is not modified and persists through a controller powerup or PROG-to-RUN transition.
Instruction first run	All commands that are automatically cleared on each execution are cleared and ignored. Filter is initialized. Internal timers are reset. The instruction executes normally.
Instruction first scan	See Instruction first run in the Function Block Diagram table.
EnableIn is false	EnableOut is cleared to false. The instruction shows a status of IO fault (Sts_IOFault). The calculation of the scaled input PV value (Val_InpPV) is executed to indicate to the operator the actual input value, even though the primary PV (Val) is not updated (holds last value).
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

In Structured Text, EnableIn is always true during normal scan. The instruction executes when it is in the control path activated by the logic.

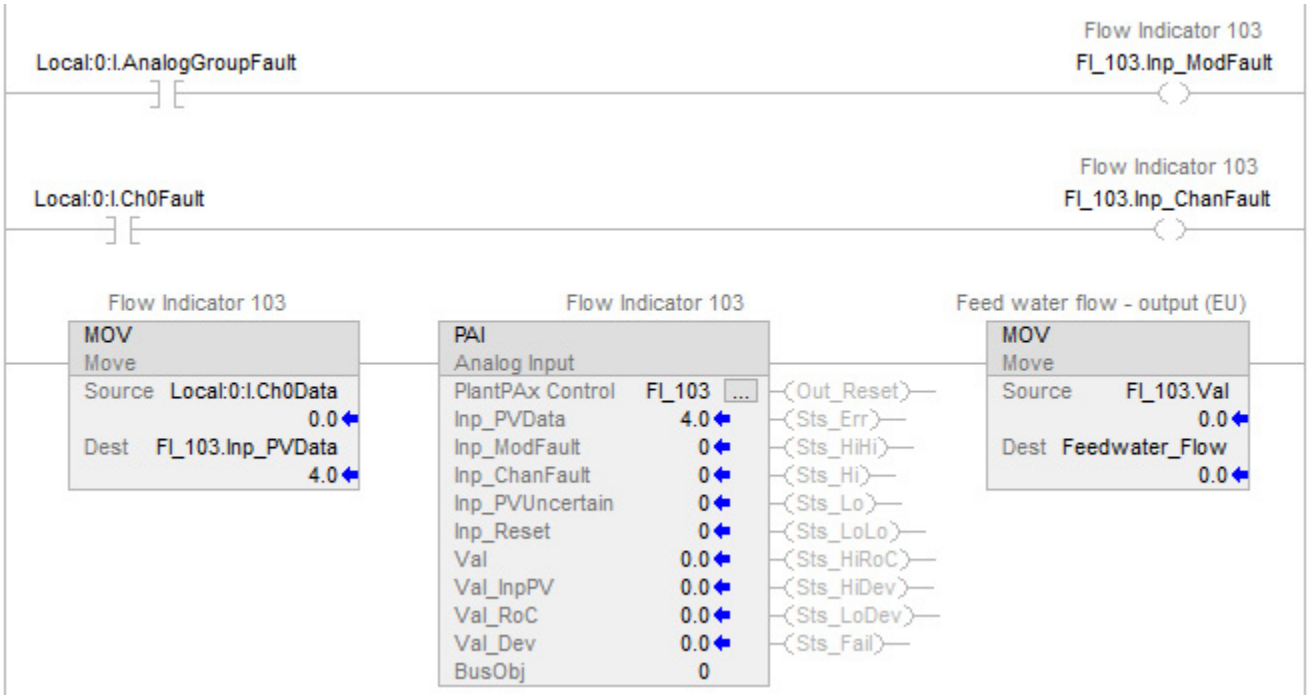
Condition/State	Action Taken
Prescan	See Prescan in the Function Block Diagram table.

Condition/State	Action Taken
Instruction first run	See Instruction first run in the Function Block Diagram table.
EnableIn is true	See EnableIn is true in the Function Block Diagram table.
Postscan	See Postscan in the Function Block Diagram table.

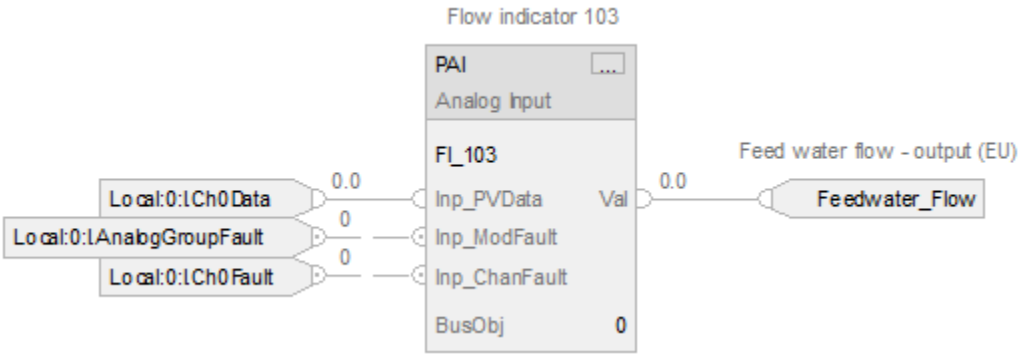
Example

The following example shows the processing of raw analog input by the PAI instruction. The raw input value (Local:1:I.ChoData) from the analog input card is used as the raw input value (Inp_PVData) for the PAI instruction. The final output process value (Feedwater_Flow) is the fully converted, scaled, and filtered analog value that is propagated through the system. The instruction also uses the Channel Fault and Module Fault parameters taken from the same analog input module as the process value. The Inp_ChanFault is the tag value for the channel (Local:1:I.ChoFault). The Inp_ModFault is the tag value for the Local:1:I.AnalogGroupFault tag, which is set when any bits in the Channel Fault word are set.

Ladder Diagram



Function Block Diagram



Structured Text

```
FI_103.Inp_PVData := Local:1:I.Ch0Data;

FI_103.Inp_ModFault := Local:1:I.AnalogGroupFault;

FI_103.Inp_ChanFault := Local:1:I.Ch0Fault;

PAI(FI_103);

Feedwater_Flow := FI_103.Out;
```

See also

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Process Dual Sensor Analog Input (PAID)

This information applies to the ControlLogix 5380P and 5580P controllers.

The Process Dual Sensor Analog Input (PAID) instruction monitors one analog Process Variable (PV) by using two analog input signals, from sources such as dual sensors, dual transmitters, and dual input channels. The PAID instruction monitors conditions of the channels and reports configured PV quality. The PAID instruction has functions for input selection, averaging, and failure detection. Additional functions, such as for filtering and alarming, are done by a downstream PAI block.

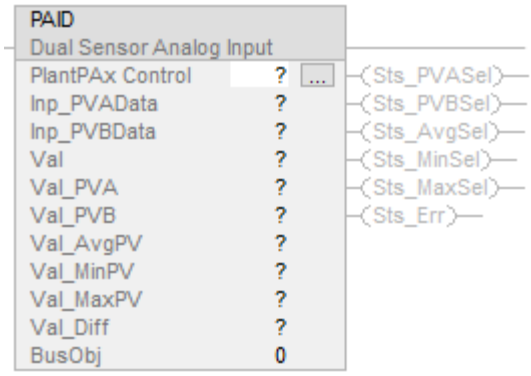
The PAID Instruction provides:

- Selection of the sensor or input A value, the sensor or input B value, the average of the two, the lesser of the two, or the greater of the two as the PV value.

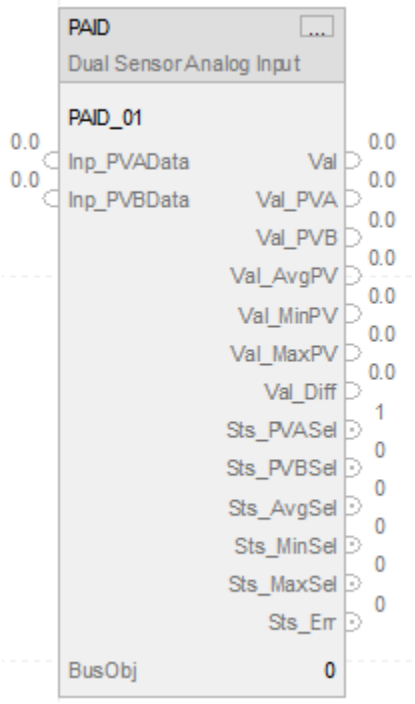
- Input Source and Quality monitoring for uncertain or bad input for each sensor, transmitter, or input, plus monitoring of each signal for out-of-range condition. If one PV is bad, failed, or out or range, the other PV is automatically selected.
- Warning alarm if the difference between the two sensor PVs exceeds a configured limit.
- Warning alarm if only one PV has good quality.
- Warning alarm if neither PV has good quality; for example, if both are uncertain.
- Failure alarm if both PVs are bad; for example, each PV has bad quality (Inp_PVABad or Inp_PVBBad) or is outside the configured failure range.

Available Languages

Ladder Diagram



Function Block Diagram



Structured Text

PAID (PAIDTag, o);

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PlantPAx Control	P_ANALOG_INPUT_DUAL	tag	Data structure required for proper operation of instruction.
BusObj	BUS_OBJ	tag	Bus component

P_ANALOG_INPUT_DUAL Structure

Public members are standard, visible tag members that are programmatically accessible. Private, hidden members are used in HMI faceplates and are not

programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	Description
EnableIn	BOOL	Enable Input - System Defined Parameter Default is true.
Inp_InitializeReq	BOOL	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request when reinitialization is needed. The instruction clears this operand automatically. Default is true.
Inp_PVAData	REAL	PV signal from sensor or input A (PV units). Valid = any float. Default is 0.0.
Inp_PVASrcQ	SINT	Input source and quality, from channel A object, if available (enumeration). Default is 0.
Inp_PVANotify	SINT	Related PV object alarm priority and acknowledgement status. 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged. Default is 0.
Inp_SmartDvcASts	DINT	Current code provided by SMART Device on Inp_PVAData. The code is copied to Out_SmartDvcSts allowing a user to monitor the device status on HMI for diagnostic lookup purposes. Valid = 0 to maximum positive number. Default is 0.
Inp_PVBData	REAL	PV signal from sensor or input B (PV units). Valid = any float. Default is 0.0.
Inp_PVBSrcQ	SINT	Input source and quality, from channel B object, if available (enumeration). Default is 0.
Inp_PVBNotify	SINT	Related PV object alarm priority and acknowledgement status. 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged. Default is 0.
Inp_SmartDvcBSts	DINT	Current code provided by SMART Device on Inp_PVBData. The code is copied to Out_SmartDvcSts allowing a user to monitor the device status on HMI for diagnostic lookup purposes. Valid = 0 to maximum positive number. Default is 0.
Inp_PVABad	BOOL	Signal quality or communication status for input A: 1 = Bad, 0 = OK. Default is false.

Public Input Members	Data Type	Description
Inp_PVAUncertain	BOOL	Signal quality for input A: 1 = Uncertain, 0 = OK. Default is false.
Inp_SmartDvcADiagAvailable	BOOL	1 = SMART Device on Inp_PVAData diagnostics available. Typically used to indicate device requires action to keep operating as expected. Default is false.
Inp_PVBBad	BOOL	Signal quality or communication status for input B: 1 = Bad, 0 = OK. Default is false.
Inp_PVBUncertain	BOOL	Signal quality for input B: 1 = Uncertain, 0 = OK. Default is false.
Inp_SmartDvcBDiagAvailable	BOOL	1 = SMART device on Inp_PVBData diagnostics available. Typically used to indicate device requires action to keep operating as expected. Default is false.
Inp_DiffGate	BOOL	The gate input used for status detection. 1 = The corresponding analog input threshold monitoring is enabled. 0 = detection is disabled and the corresponding status output is forced off. Default is false.
Cfg_AllowDisable	BOOL	1 = Allow maintenance to disable alarms. Default is true.
Cfg_AllowShelve	BOOL	1 = Allow operator to shelve alarms. Default is true.
Cfg_UseInpSrcQPVA	BOOL	1 = Use PVA SrcQ input for rejection decisions. Default is false.
Cfg_UseInpSrcQPVB	BOOL	1 = Use PVB SrcQ input for rejection decisions. Default is false.
Cfg_HasPVNav	BOOL	1 = Tells HMI to enable navigation to a connected PV (Val) object. Default is false.
Cfg_HasMoreObj	BOOL	1 = Tells HMI an object with more information is available for navigation. Default is false.
Cfg_HasNav	SINT	Set bits indicate which navigation buttons are enabled .0=PVA, .1=PVB Default is 0.
Cfg_PVEUMin	REAL	PV (Output) minimum for display PV units. Valid = any float. Default is 0.0.
Cfg_PVEUMax	REAL	PV (Output) maximum for display PV units. Valid = any float. Default is 100.0.
Cfg_DiffLim	REAL	Signal difference status limit for PV units, difference. Valid = any nonnegative float. Default is 1.50E+38.
Cfg_DiffDB	REAL	Signal difference status deadband for PV units, difference. Valid = any nonnegative float. Default is 1.0.
Cfg_DiffGateDly	REAL	The time (seconds) after the gate input activates before the threshold detection is enabled. Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_OoRHiLim	REAL	Out-of-range (fail) high limit for input units. Valid = any float. Default is 103.958336.
Cfg_OoRLoLim	REAL	Out-of-range (fail) low limit for input units. Valid = any float. Default is -2.0833333.
Cfg_OoRDB	REAL	Out-of-range (fail) high or low deadband for input units. Valid = any nonnegative float. Default is 0.41666666.
Cfg_AllowOper	BOOL	1 = Oper is allowed to control PV selection. Default is false.

Public Input Members	Data Type	Description
Cfg_AllowProg	BOOL	1 = Prog is allowed to control PV selection. Default is false.
Cfg_AllowExt	BOOL	1 = Ext is allowed to control PV selection. Default is false.
Cfg_PVDecPlcs	SINT	Number of decimal places for PV display. Valid = 0,1,2,3,4,5,6. Default is 2.
Cfg_CnfrmReqd	SINT	Operator command confirmation required. Represents the type of command confirmation required. 0 = None, 1 = Command confirmation required, 2 = Performer e-signature required, 3 = Performer and approver e-signature required. Default is 0.
PSet_Owner	DINT	Program owner request ID (non-zero) or release (zero) (Valid = any integer). Default is 0.
PCmd_SelA	BOOL	Program command to select sensor A PV. Default is false.
PCmd_SelB	BOOL	Program command to select sensor B PV. Default is false.
PCmd_SelAvg	BOOL	Program command to select average (A,B) PV. Default is false.
PCmd_SelMin	BOOL	Program command to select minimum (A,B) PV. Default is false.
PCmd_SelMax	BOOL	Program command to select maximum (A,B) PV. Default is false.
PCmd_Reset	BOOL	Program command to reset all alarms requiring reset. Default is false.
XCmd_SelA	BOOL	External command to select sensor A PV. Default is false.
XCmd_SelB	BOOL	External command to select sensor B PV. Default is false.
XCmd_SelAvg	BOOL	External command to select average (A,B) PV. Default is false.
XCmd_SelMin	BOOL	External command to select minimum (A,B) PV. Default is false.
XCmd_SelMax	BOOL	External command to select maximum (A,B) PV. Default is false.
XCmd_Reset	BOOL	External command to reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.
XCmd_ResetAckAll	BOOL	External command to acknowledge and reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.

Public Output Members	Data Type	Description
EnableOut	BOOL	Enable Output - System Defined Parameter
Val	REAL	Selected analog PV , including substitute PV, if used (PV units).
Val_PVA	REAL	Analog value (actual) from input A (PV units).
Val_PVB	REAL	Analog value (actual) from input B (PV units).
Val_AvgPV	REAL	Analog value average of input A and input B (PV units).

Public Output Members	Data Type	Description
Val_MinPV	REAL	Analog value minimum of input A and input B (PV units).
Val_MaxPV	REAL	Analog value maximum of input A and input B (PV units).
Val_InpPV	REAL	Selected PV, before substitution, for example (PV units).
Val_Diff	REAL	Difference between input A and input B PVs (PV units).
Val_PVEUMin	REAL	Minimum of PV range = minimum (Cfg_PVEUMin, Cfg_PVEUMax) (PV units).
Val_PVEUMax	REAL	Maximum of PV range = maximum (Cfg_PVEUMin, Cfg_PVEUMax) (PV units).
Out_SmartDvcSts	DINT	Status code of a SMART device provided by Inp_SmartDvcASts or Inp_SmartDvcBSts. Highest status code selected.
Sts_Initialized	BOOL	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.
Sts_SmartDvcDiagAvailable	BOOL	1 = Diagnostics of a SMART device is currently available. Typically used to indicate one of the devices requires action to keep operating as expected.
Sts_PVASel	BOOL	1 = Input A selected for PV.
Sts_PVBSel	BOOL	1 = Input B selected as PV.
Sts_AvgSel	BOOL	1 = Average (A,B) selected as PV.
Sts_MinSel	BOOL	1 = Minimum (A,B) selected as PV.
Sts_MaxSel	BOOL	1 = Maximum (A,B) selected as PV.
Sts_PVBad	BOOL	1 = PV bad quality or out of range.
Sts_PVUncertain	BOOL	1 = PV value is uncertain (quality).
SrcQ_IOA	SINT	Source and quality of primary I/O (enumeration).
SrcQ_IOB	SINT	Source and quality of primary I/O (enumeration).
SrcQ_IO	SINT	Source and quality of primary I/O (enumeration).
SrcQ	SINT	Source and quality of primary Val or Sts (enumeration).
Sts_eSts	SINT	Device confirmed status (enum): 0 = PV Good, 1: PV Uncertain, 2: PV Bad.
Sts_eFault	INT	Device fault status (enum): 0 = None, ... 32 = Fail, 34 = Bad Config.
Sts_eNotify	SINT	All alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyAll	SINT	All alarm status enumerated values including related objects: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.

Public Output Members	Data Type	Description
Sts_eNotifyOneGood	SINT	Only one good PV alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyNoneGood	SINT	No good PV alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyDiff	SINT	Input Difference alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyFail	SINT	Fail alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_UnackAlmCount	DINT	Count of unacknowledged alarms.
Sts_Err	BOOL	1 = Error in config, see detail bits for reason.
Sts_ErrEU	BOOL	1 = Error in config: Cfg_PVEUMax cannot equal Cfg_PVEUMin.
Sts_ErrDiffDB	BOOL	1 = Error in configuration: Cfg_LoDevDB deadband is < 0.0.
Sts_ErrDiffGateDly	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrOoRDB	BOOL	1 = Error in configuration: Cfg_OoRDB deadband is < 0.0.
Sts_ErrAlm	BOOL	1 = Error in logix tag-based alarm settings.

Public Output Members	Data Type	Description
Sts_Alm	BOOL	1 = An alarm is active.
Sts_AlmInh	BOOL	1 = An alarm is shelved or disabled.
Sts_OneGood	BOOL	1 = Only one good PV (other is bad or uncertain).
Sts_NoneGood	BOOL	1 = No good PV (both bad, or one bad or one uncertain).
Sts_DiffCmp	BOOL	Signal difference comparison result 1 = high difference.
Sts_DiffGate	BOOL	Signal difference gate delay status, 1 = done.
Sts_Diff	BOOL	1 = High signal difference detected.
Sts_Fail	BOOL	1 = Total signal failure (both bad or out of range).
Sts_RdyAck	BOOL	1 = An alarm is ready to be acknowledged.
Sts_RdyReset	BOOL	1 = A latched alarm or shed condition is ready to be reset.
XRdy_SelA	BOOL	1 = Ready for XCmd_SelA, enable HMI button.
XRdy_SelB	BOOL	1 = Ready for XCmd_SelB, enable HMI button.
XRdy_SelAvg	BOOL	1 = Ready for XCmd_SelAvg, enable HMI button.
XRdy_SelMin	BOOL	1 = Ready for XCmd_SelMin, enable HMI button.
XRdy_SelMax	BOOL	1 = Ready for XCmd_SelMax, enable HMI button.
XRdy_Reset	BOOL	1 = Ready for XCmd_Reset, enable HMI button.
XRdy_ResetAckAll	BOOL	1 = Ready for XCmd_ResetAckAll, enable HMI button.
Val_Owner	DINT	Current object owner ID (0 = not owned).

Private Input Members	Data Type	Description
HMI_BusObjIndex	DINT	HMI bus object index. Default is 0.
OCmd_Reset	BOOL	Operator command to reset all alarms requiring reset. Default is false.
OCmd_ResetAckAll	BOOL	Operator command to reset and acknowledge all alarms. Default is false.
OCmd_SelA	BOOL	Operator command to select sensor A PV. Default is false.
OCmd_SelAvg	BOOL	Operator command to select average(A,B) PV. Default is false.
OCmd_SelB	BOOL	Operator command to select sensor B PV. Default is false.
OCmd_SelMax	BOOL	Operator command to select maximum(A,B) PV. Default is false.
OCmd_SelMin	BOOL	Operator command to select minimum(A,B) PV. Default is false.

Private Output Members	Data Type	Description
ORdy_Reset	BOOL	1 = Ready for OCmd_Reset (enables HMI button).
ORdy_ResetAckAll	BOOL	1 = Ready for OCmd_ResetAckAll (enables HMI button).
ORdy_SelA	BOOL	1 = Ready for OCmd_SelA (enables HMI button).
ORdy_SelAvg	BOOL	1 = Ready for OCmd_SelAvg.
ORdy_SelB	BOOL	1 = Ready for OCmd_SelB (enables HMI button).
ORdy_SelMax	BOOL	1 = Ready for OCmd_SelMax (enables HMI button).
ORdy_SelMin	BOOL	1 = Ready for OCmd_SelMin (enables HMI button).

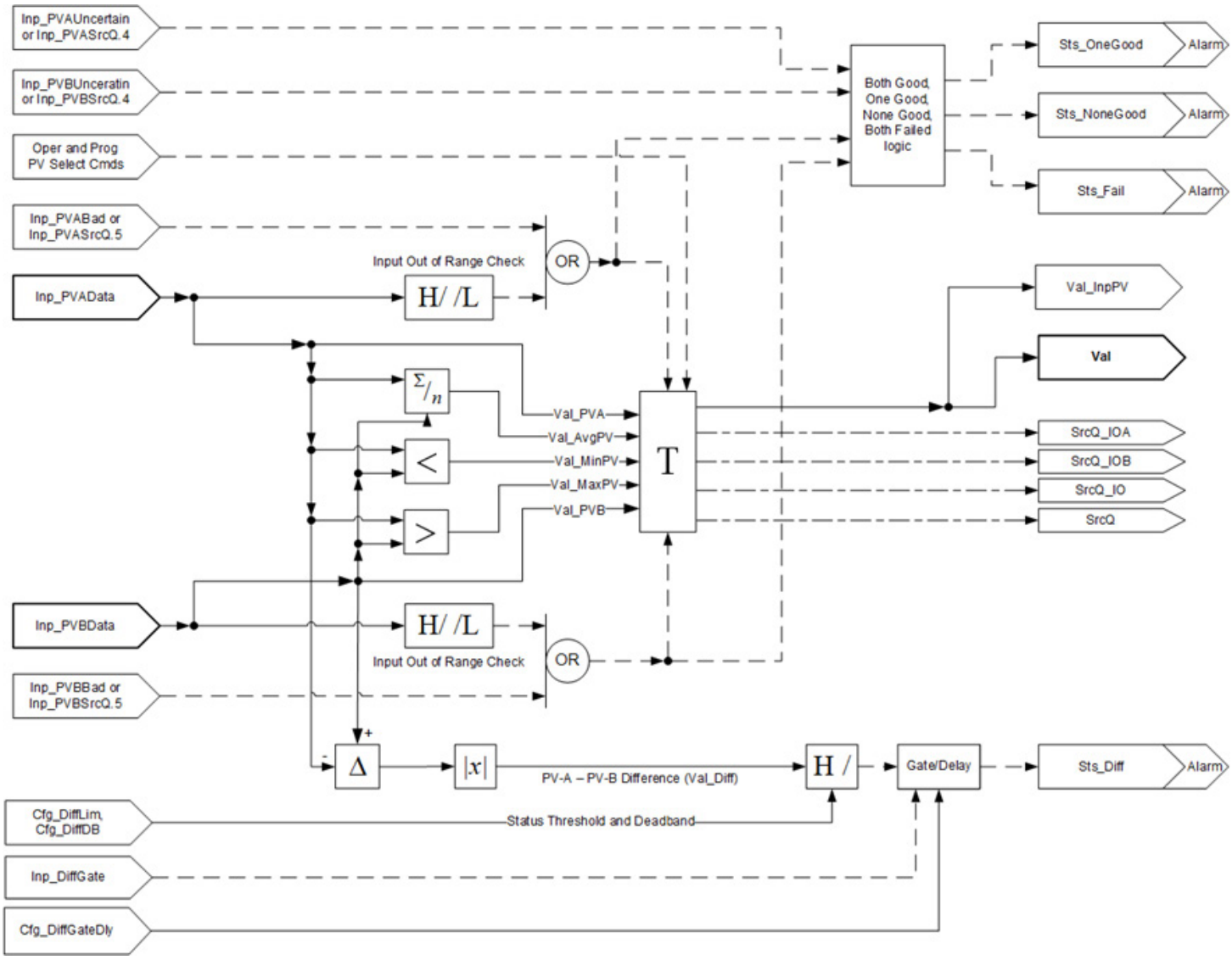
Public InOut Members	Data Type	Description
BusObj	BUS_OBJ	Bus component

BUS_OBJ Structure

Members	Data Type	Description
Inp_Cmd	DINT	Input to assert commands
Out_Cmd	DINT	Resultant commands
Inp_CmdLLH	DINT	Input for level normally high commands
Out_CmdLLH	DINT	Resultant line level high commands
Inp_Sts	DINT	Input to assert status
Out_Sts	DINT	Resultant status
Inp_CmdAck	DINT	Input to assert a command acknowledgement
Out_CmdAck	DINT	Resultant command acknowledgements
Inp_SeverityMax	DINT	Input: maximum alarm severity
Out_SeverityMax	DINT	Resultant of maximum alarm severity
Cfg_CmdMask	DINT	Propagation mask for commands
Cfg_CmdLLHMask	DINT	Propagation mask for line level high commands
Cfg_StsMask	DINT	Propagation mask for status
Ref_Index	DINT	Bus array index

Operation

This diagram illustrates the functionality of the PAID instruction:



Alarms

Discrete tag-based alarms are defined for these members.

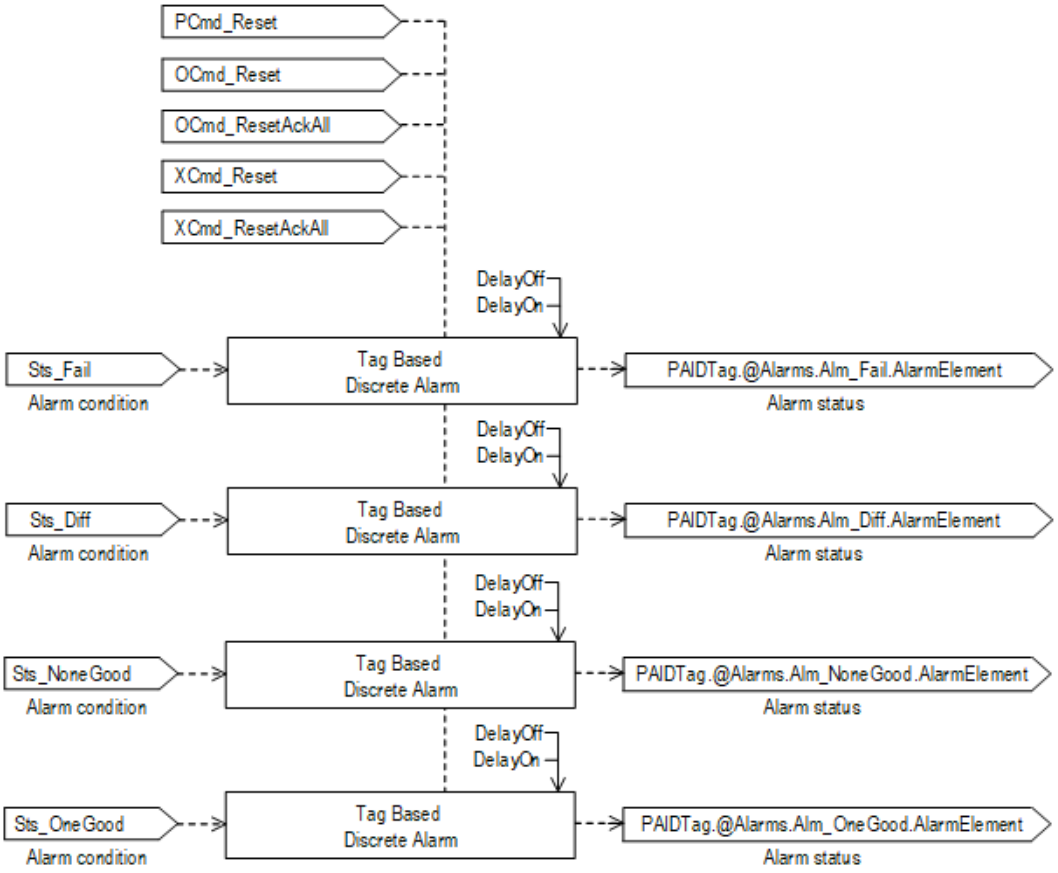
Member	Alarm Name	Description
Sts_Fail	Alm_Fail	Raised when the two sensor PVs are bad or out of range.
Sts_Diff	Alm_Diff	Raised when a high signal difference is detected between the two sensors PVs exceeds a configured limit.
Sts_NoneGood	Alm_NoneGood	Raised when neither PV has good quality (for example, if both are uncertain).
Sts_OneGood	Alm_OneGood	Raised when only one PV has a good quality.

Mark the alarm as used or unused and set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format:

PAIDTag.@Alarms.AlarmName.AlarmElement

There are Program, Operator, and External commands that enable the Reset and Reset & Acknowledge of all alarms of the instruction (Alarm Set) at the

same time. This diagram shows how the commands interact with the PDO instruction.



Initialization

The instruction is normally initialized in the instruction first run. Request re-initialization by setting Inp_InitializeReq = 1. For proper initialization, when adding the instruction while performing an online edit of the code, make sure that Inp_InitializeReq = 1, the default value.

Configuration of Strings for HMI

Configure strings for HMI faceplates, as seen in FactoryTalk View, and for the Logix Designer configuration dialog box.

- Description
- Label for graphic symbol
- Display Library for HMI Faceplate call-up
- Instruction name
- Area name
- URL link
- More Information

- Selected Analog PV engineering units – Units metadata of Val member, Val.@Units.
- Analog input A description – Label metadata of Inp_PVADataData member, Inp_PVADataData.@Label.
- Analog input B description – Label metadata of Inp_PVBDataData member, Inp_PVBDataData.@Label.
- Allow Navigation Object Tag Name Output – Navigation metadata of PAID member Val tag.
- Allow Navigation Object Tag Name Input A – Navigation metadata of PAID member Inp_PVADataData tag.
- Allow Navigation Object Tag Name Input B – Navigation metadata of PAID member Inp_PVBDataData tag

Monitor the PAID Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Rung-condition-out clears to false.
Instruction first run	All commands that are automatically cleared on each execution are cleared and ignored. PSet_Owner and Val_Owner are set to 0. The instruction executes normally.

Condition/State	Action Taken
Rung-condition-in is false	Rung-condition-out is cleared to false. All commands that are automatically cleared on each execution are cleared and ignored. PSet_Owner and Val_Owner are set to 0. Latched alarms are reset. Clear Bus Object commands and HMI Bus Object Index Execute Bus command on receipt for Disable, Enable, Suppress, Unsuppress all alarms. Execute Bus command status propagation. Internal timers are reset.
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	EnableOut is cleared to false.
Instruction first run	All commands that are automatically cleared on each execution are cleared and ignored. PSet_Owner and Val_Owner are set to 0. The instruction executes normally.
Instruction first scan	See Instruction first run in the Function Block Diagram table.
EnableIn is false	EnableOut is cleared to false. All commands that are automatically cleared on each execution are cleared and ignored. PSet_Owner and Val_Owner are set to 0. Latched alarms are reset. Clear Bus Object commands and HMI Bus Object Index Execute Bus command on receipt for Disable, Enable, Suppress, Unsuppress all alarms. Execute Bus command status propagation. Internal timers are reset.
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

In Structured Text, EnableIn is always true during normal scan. The instruction executes when it is in the control path activated by the logic.

Condition/State	Action Taken
Prescan	See Prescan in the Function Block Diagram table.
Instruction first run	See Instruction first run in the Function Block Diagram table.
EnableIn is true	See EnableIn is true in the Function Block Diagram table.
Postscan	See Postscan in the Function Block Diagram table.

Example

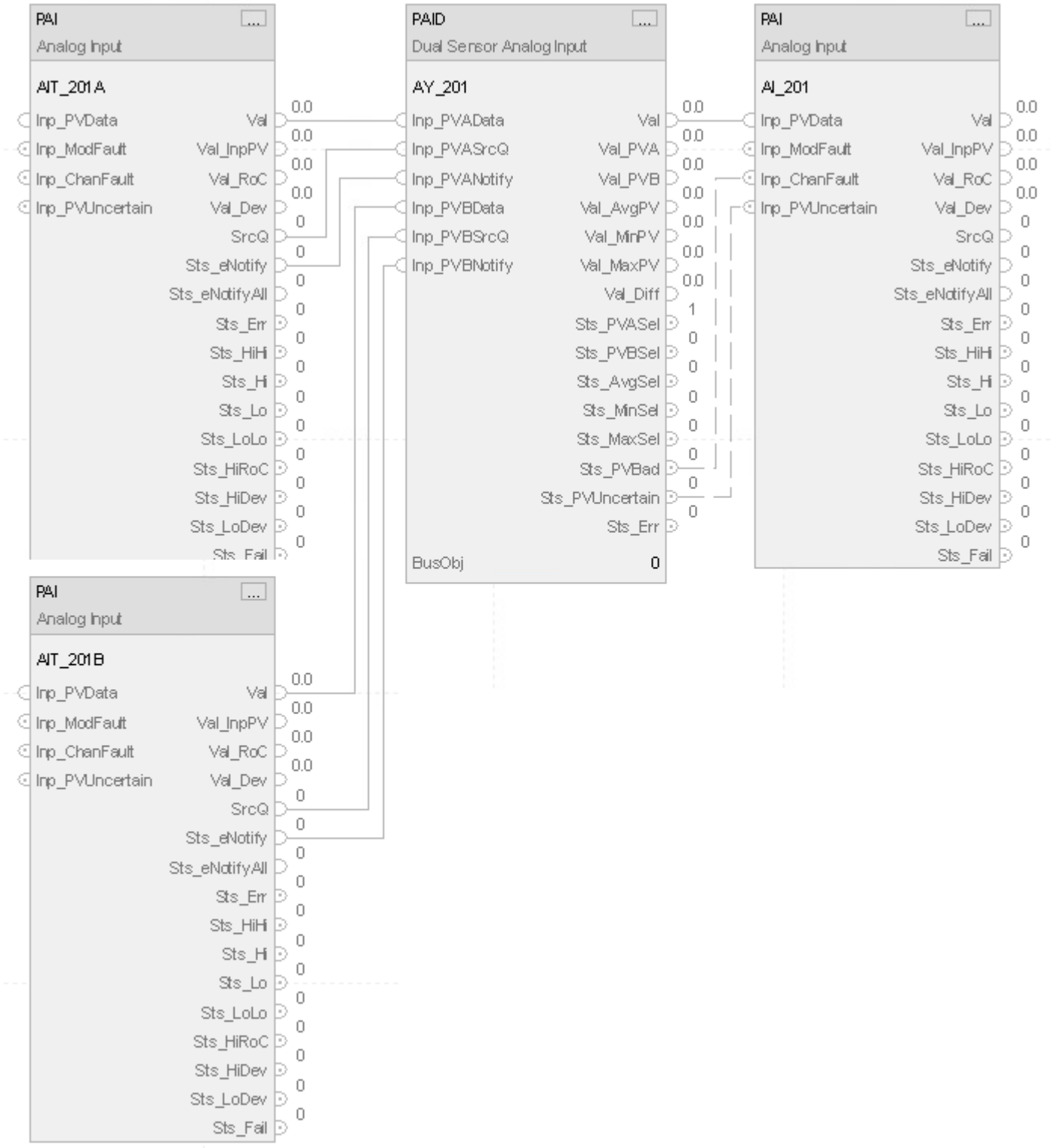
This example uses the PAID instruction to monitor one analog Process Variable (PV) using two analog input signals (dual sensors, dual transmitters). The PAID instruction allows you to select one sensor, the other sensor, or the average, minimum or maximum of either sensors. If difference between the two input signals exceeds a configured limit, an Alarm is generated.

In this example, an application uses two analog sensors (A, B). The average of these analogs is used elsewhere in logic to control a separate application element.

The Inp_PVAData and Inp_PVBData parameters are connected to the values from the two analog transmitters. The fault status of each of these sensors is tied to the bad input of the instruction (for example, Inp_PVABad). The output parameters Val, Sts_PVBad and Sts_PVUncertain, can then be connected to the Inp_PVData, Inp_PVUncertain and Inp_ChanFault for control.

Ladder Diagram

Function Block Diagram



Structured Text

```
PAI(AIT_201A);
```

```
PAI(AIT_201B);
AY_201.Inp_PVAData := AIT_201A.Val;
AY_201.Inp_PVASrcQ := AIT_201A.SrcQ;
AY_201.Inp_PVANotify := AIT_201A.Sts_eNotify;
AY_201.Inp_PVBData := AIT_201B.Val;
AY_201.Inp_PVBSrcQ := AIT_201B.SrcQ;
AY_201.Inp_PVBNotify := AIT_201B.Sts_eNotify;
PAID(AY_201,0);
AI_201.Inp_PVData := AY_201.Val;
AI_201.Inp_ChanFault := AY_201.Sts_PVBad;
AI_201.Inp_PVUncertain := AY_201.Sts_PVUncertain;
PAI(AI_201);
```

See also

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Process Multi Sensor Analog Input (PAIM)

This information applies to the ControlLogix 5380P and 5580P controllers.

The Process Multi Sensor Analog Input (PAIM) instruction monitors one analog process variable (PV) by using up to eight analog input signals from sources such as sensors, transmitters, and input channels. The PAIM instruction has functions for input selection, averaging, and failure detection. Additional functions, such as filtering and alarming, are done by a downstream PAI block.

The PAIM instruction provides:

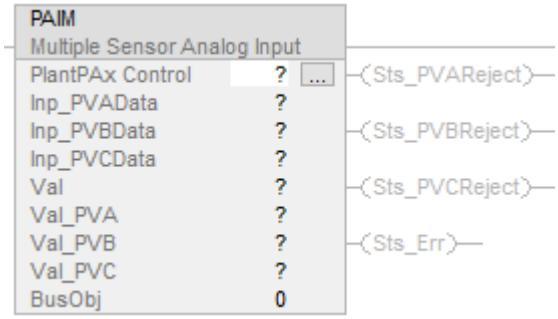
- Configuration to use between two and eight input signals.
- Input Source and Quality monitoring of inputs, plus monitoring of each signal for out of range condition. Rejection from the PV calculation of inputs that are out of range, flagged as bad, infinite, or not a number (floating-point exception values).
- Calculation of the average (mean) or median of the inputs in use as the PV value.
- Selectable rejection from the PV calculation of inputs that are outside tau standard deviations from the mean, with a minimum of four

required inputs, or inputs that are outside a user-defined deviation from the mean.

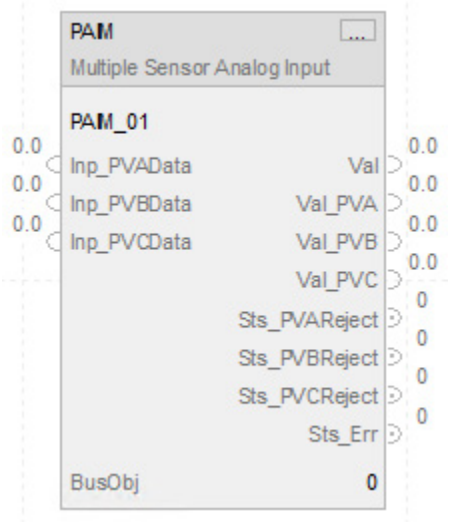
- Configuration of the minimum number of good, unrejected input signals required to have a good PV value, and an alarm if the required number of good inputs is not met.
- Configuration of which PV to use if there are only two unrejected signals remaining: the lesser, the greater, or the average of the two.
- An alarm if any inputs configured to be used are rejected.
- An alarm if the number of unrejected inputs is equal to the minimum number required to be good, meaning the next input failure results in a PV failure.
- Display elements, plus a faceplate with bar graph PV indication, mode selection, alarm limit entry and alarm display, configuration, acknowledgment, trending, and maintenance and engineering configuration and setup.

Available Languages

Ladder Diagram



Function Block Diagram



Structured Text

PAIM (PAIMTag, o);

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PlantPAx Control	P_ANALOG_INPUT_MULTI	tag	Data structure required for proper operation of instruction.
BusObj	BUS_OBJ	tag	Bus component

P_ANALOG_INPUT_MULTI Structure

Public members are standard, visible tag members that are programmatically accessible. Private, hidden members are used in HMI faceplates and are not programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
EnableIn	BOOL	Not Visible	Not Required	Input	Enable Input - System Defined Parameter Default is true.
Inp_InitializeReq	BOOL	Not Visible	Not Required	Input	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request when reinitialization is needed. The instruction clears this operand automatically. Default is true.
Inp_PVAData	REAL	Visible	Not Required	Input	PV signal from sensor or input A (PV units). Valid = any float. Default is 0.0.
Inp_PVASrcQ	SINT	Not Visible	Not Required	Input	Input source and quality, from channel A object, if available (enumeration). Default is 0.
Inp_PVANotify	SINT	Not Visible	Not Required	Input	Related PV object alarm priority and acknowledgement status. 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged. Default is 0.
Inp_SmartDvcASts	DINT	Not Visible	Not Required	Input	Current code provided by SMART device on Inp_PVAData. The code is copied to Out_SmartDvcSts allowing a user to monitor the device status on HMI for diagnostic lookup purposes. Valid = 0 to maximum positive number. Default is 0.
Inp_PVBData	REAL	Visible	Not Required	Input	PV signal from sensor or input B (PV units). Valid = any float. Default is 0.0.
Inp_PVBSrcQ	SINT	Not Visible	Not Required	Input	Input source and quality, from channel B object, if available (enumeration). Default is 0.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Inp_PVBNotify	SINT	Not Visible	Not Required	Input	Related PV object alarm priority and acknowledgement status. 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged. Default is 0.
Inp_SmartDvcBSts	DINT	Not Visible	Not Required	Input	Current code provided by SMART device on Inp_PVBData. The code is copied to Out_SmartDvcSts allowing a user to monitor the device status on HMI for diagnostic lookup purposes. Valid = 0 to maximum positive number. Default is 0.
Inp_PVCData	REAL	Visible	Not Required	Input	PV signal from sensor or input C (PV units). Valid = any float. Default is 0.0.
Inp_PVCSrcQ	SINT	Not Visible	Not Required	Input	Input source and quality, from channel C object, if available (enumeration). Default is 0.
Inp_PVCNotify	SINT	Not Visible	Not Required	Input	Related PV object alarm priority and acknowledgement status. 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged. Default is 0.
Inp_SmartDvcCSts	DINT	Not Visible	Not Required	Input	Current code provided by SMART device on Inp_PVCData. The code is copied to Out_SmartDvcSts allowing a user to monitor the device status on HMI for diagnostic lookup purposes. Valid = 0 to maximum positive number. Default is 0.
Inp_PVDData	REAL	Not Visible	Not Required	Input	PV signal from sensor or input D (PV units). Valid = any float. Default is 0.0.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Inp_PVDSrcQ	SINT	Not Visible	Not Required	Input	Input source and quality, from channel D object, if available (enumeration). Default is 0.
Inp_PVDNotify	SINT	Not Visible	Not Required	Input	Related PV object alarm priority and acknowledgement status. 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged. Default is 0.
Inp_SmartDvcDSts	DINT	Not Visible	Not Required	Input	Current code provided by SMART device on Inp_PVDData. The code is copied to Out_SmartDvcSts allowing a user to monitor the device status on HMI for diagnostic lookup purposes. Valid = 0 to maximum positive number. Default is 0.
Inp_PVEData	REAL	Not Visible	Not Required	Input	PV signal from sensor or input E (PV units). Valid = any float. Default is 0.0.
Inp_PVESrcQ	SINT	Not Visible	Not Required	Input	Input source and quality, from channel E object, if available (enumeration). Default is 0.
Inp_PVENotify	SINT	Not Visible	Not Required	Input	Related PV object alarm priority and acknowledgement status. 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged. Default is 0.
Inp_SmartDvcESts	DINT	Not Visible	Not Required	Input	Current code provided by SMART device on Inp_PVEData. The code is copied to Out_SmartDvcSts allowing a user to monitor the device status on HMI for diagnostic lookup purposes. Valid = 0 to maximum positive number. Default is 0.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Inp_PVFData	REAL	Not Visible	Not Required	Input	PV signal from sensor or input F (PV units). Valid = any float. Default is 0.0.
Inp_PVFSrcQ	SINT	Not Visible	Not Required	Input	Input source and quality, from channel F object, if available (enumeration). Default is 0.
Inp_PVFNotify	SINT	Not Visible	Not Required	Input	Related PV object alarm priority and acknowledgement status. 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged. Default is 0.
Inp_SmartDvcFSts	DINT	Not Visible	Not Required	Input	Current code provided by SMART device on Inp_PVFData. The code is copied to Out_SmartDvcSts allowing a user to monitor the device status on HMI for diagnostic lookup purposes. Valid = 0 to maximum positive number. Default is 0.
Inp_PVGData	REAL	Not Visible	Not Required	Input	PV signal from sensor or input G (PV units). Valid = any float. Default is 0.0.
Inp_PVGSrcQ	SINT	Not Visible	Not Required	Input	Input source and quality, from channel G object, if available (enumeration). Default is 0.
Inp_PVGNotify	SINT	Not Visible	Not Required	Input	Related PV object alarm priority and acknowledgement status. 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged. Default is 0.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Inp_SmartDvcGSts	DINT	Not Visible	Not Required	Input	Current code provided by SMART device on Inp_PVGData. The code is copied to Out_SmartDvcSts allowing a user to monitor the device status on HMI for diagnostic lookup purposes. Valid = 0 to maximum positive number. Default is 0.
Inp_PVHData	REAL	Not Visible	Not Required	Input	PV signal from sensor or input H (PV units). Valid = any float. Default is 0.0.
Inp_PVHSrcQ	SINT	Not Visible	Not Required	Input	Input source and quality, from channel H object, if available (enumeration). Default is 0.
Inp_PVHNotify	SINT	Not Visible	Not Required	Input	Related PV object alarm priority and acknowledgement status. 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged. Default is 0.
Inp_SmartDvcHSts	DINT	Not Visible	Not Required	Input	Current code provided by SMART device on Inp_PVHData. The code is copied to Out_SmartDvcSts allowing a user to monitor the device status on HMI for diagnostic lookup purposes. Valid = 0 to maximum positive number. Default is 0.
Inp_PVABad	BOOL	Not Visible	Not Required	Input	Signal quality or communication status for input A 1 = Bad, 0 = OK. Default is false.
Inp_PVAUncertain	BOOL	Not Visible	Not Required	Input	Signal quality for input A, 1 = Uncertain, 0 = OK. Default is false.
Inp_SmartDvcADiagAvailable	BOOL	Not Visible	Not Required	Input	1 = SMART device on Inp_PVAData diagnostics available. Typically used to indicate device requires action to keep operating as expected. Default is false.
Inp_PVBBad	BOOL	Not Visible	Not Required	Input	Signal quality or communication status for input B 1 = Bad, 0 = OK. Default is false.
Inp_PVBUncertain	BOOL	Not Visible	Not Required	Input	Signal quality for input B, 1 = Uncertain, 0 = OK. Default is false.
Inp_SmartDvcBDiagAvailable	BOOL	Not Visible	Not Required	Input	1 = SMART device on Inp_PVBData diagnostics available. Typically used to indicate device requires action to keep operating as expected. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Inp_PVCBad	BOOL	Not Visible	Not Required	Input	Signal quality or communication status for input C 1 = Bad, 0 = OK. Default is false.
Inp_PVCUncertain	BOOL	Not Visible	Not Required	Input	Signal quality for input C, 1 = Uncertain, 0 = OK. Default is false.
Inp_SmartDvcCDiagAvailable	BOOL	Not Visible	Not Required	Input	1 = SMART device on Inp_PVCData diagnostics available. Typically used to indicate device requires action to keep operating as expected. Default is false.
Inp_PVDBad	BOOL	Not Visible	Not Required	Input	Signal quality or communication status for input D 1 = Bad, 0 = OK. Default is false.
Inp_PVDUncertain	BOOL	Not Visible	Not Required	Input	Signal quality for input D, 1 = Uncertain, 0 = OK. Default is false.
Inp_SmartDvcDDiagAvailable	BOOL	Not Visible	Not Required	Input	1 = SMART device on Inp_PVDData diagnostics available. Typically used to indicate device requires action to keep operating as expected. Default is false.
Inp_PVEBad	BOOL	Not Visible	Not Required	Input	Signal quality or communication status for input E 1 = Bad, 0 = OK. Default is false.
Inp_PVEUncertain	BOOL	Not Visible	Not Required	Input	Signal quality for input E, 1 = Uncertain, 0 = OK. Default is false.
Inp_SmartDvcEDiagAvailable	BOOL	Not Visible	Not Required	Input	1 = SMART device on Inp_PVEData diagnostics available. Typically used to indicate device requires action to keep operating as expected. Default is false.
Inp_PVFBad	BOOL	Not Visible	Not Required	Input	Signal quality or communication status for input F 1 = Bad, 0 = OK. Default is false.
Inp_PVFUncertain	BOOL	Not Visible	Not Required	Input	Signal quality for input F, 1 = Uncertain, 0 = OK. Default is false.
Inp_SmartDvcFDiagAvailable	BOOL	Not Visible	Not Required	Input	1 = SMART device on Inp_PVFData diagnostics available. Typically used to indicate device requires action to keep operating as expected. Default is false.
Inp_PVGBad	BOOL	Not Visible	Not Required	Input	Signal quality or communication status for input G 1 = Bad, 0 = OK. Default is false.
Inp_PVGUncertain	BOOL	Not Visible	Not Required	Input	Signal quality for input G, 1 = Uncertain, 0 = OK. Default is false.
Inp_SmartDvcGDiagAvailable	BOOL	Not Visible	Not Required	Input	1 = SMART device on Inp_PVGData diagnostics available. Typically used to indicate device requires action to keep operating as expected. Default is false.
Inp_PVHBad	BOOL	Not Visible	Not Required	Input	Signal quality or communication status for input H 1 = Bad, 0 = OK. Default is false.
Inp_PVHUncertain	BOOL	Not Visible	Not Required	Input	Signal quality for input H, 1 = Uncertain, 0 = OK. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Inp_SmartDvcHDiagAvailable	BOOL	Not Visible	Not Required	Input	1 = SMART device on Inp_PVHData diagnostics available. Typically used to indicate device requires action to keep operating as expected. Default is false.
Cfg_AllowDisable	BOOL	Not Visible	Not Required	Input	1 = Allow maintenance to disable alarms. Default is true.
Cfg_AllowShelve	BOOL	Not Visible	Not Required	Input	1 = Allow operator to shelve alarms. Default is true.
Cfg_HasPVA	BOOL	Not Visible	Not Required	Input	1 = Inp_PVAData is connected in logic. Default is true.
Cfg_HasPVB	BOOL	Not Visible	Not Required	Input	1 = Inp_PVBData is connected in logic. Default is true.
Cfg_HasPVC	BOOL	Not Visible	Not Required	Input	1 = Inp_PVCData is connected in logic. Default is true.
Cfg_HasPVD	BOOL	Not Visible	Not Required	Input	1 = Inp_PVDData is connected in logic. Default is false.
Cfg_HasPVE	BOOL	Not Visible	Not Required	Input	1 = Inp_PVEData is connected in logic. Default is false.
Cfg_HasPVF	BOOL	Not Visible	Not Required	Input	1 = Inp_PVFData is connected in logic. Default is false.
Cfg_HasPVG	BOOL	Not Visible	Not Required	Input	1 = Inp_PVGData is connected in logic. Default is false.
Cfg_HasPVH	BOOL	Not Visible	Not Required	Input	1 = Inp_PVHData is connected in logic. Default is false.
Cfg_UsePVA	BOOL	Not Visible	Not Required	Input	1 = Inp_PVAData should be used in PV calculation if good. Default is true.
Cfg_UsePVB	BOOL	Not Visible	Not Required	Input	1 = Inp_PVBData should be used in PV calculation if good. Default is true.
Cfg_UsePVC	BOOL	Not Visible	Not Required	Input	1 = Inp_PVCData should be used in PV calculation if good. Default is true.
Cfg_UsePVD	BOOL	Not Visible	Not Required	Input	1 = Inp_PVDData should be used in PV calculation if good. Default is false.
Cfg_UsePVE	BOOL	Not Visible	Not Required	Input	1 = Inp_PVEData should be used in PV calculation if good. Default is false.
Cfg_UsePVF	BOOL	Not Visible	Not Required	Input	1 = Inp_PVFData should be used in PV calculation if good. Default is false.
Cfg_UsePVG	BOOL	Not Visible	Not Required	Input	1 = Inp_PVGData should be used in PV calculation if good. Default is false.
Cfg_UsePVH	BOOL	Not Visible	Not Required	Input	1 = Inp_PVHData should be used in PV calculation if good. Default is false.
Cfg_RejectUncertain	BOOL	Not Visible	Not Required	Input	1 = Reject an input if its quality is uncertain. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_UseStdDev	BOOL	Not Visible	Not Required	Input	1 = Reject outside tau standard deviations; 0 = Reject ouside Cfg_AbsDev from mean. Default is false.
Cfg_CalcAvg	BOOL	Not Visible	Not Required	Input	1 = Calculate average of good inputs; 0 = calculate median of good inputs. Default is false.
Cfg_UseInpSrcQPVA	BOOL	Not Visible	Not Required	Input	1 = Use PVA SrcQ input for rejection decisions. Default is false.
Cfg_UseInpSrcQPVB	BOOL	Not Visible	Not Required	Input	1 = Use PVB SrcQ input for rejection decisions. Default is false.
Cfg_UseInpSrcQPVC	BOOL	Not Visible	Not Required	Input	1 = Use PVC SrcQ input for rejection decisions. Default is false.
Cfg_UseInpSrcQPVD	BOOL	Not Visible	Not Required	Input	1 = Use PVD SrcQ input for rejection decisions. Default is false.
Cfg_UseInpSrcQPVE	BOOL	Not Visible	Not Required	Input	1 = Use PVE SrcQ input for rejection decisions. Default is false.
Cfg_UseInpSrcQPVF	BOOL	Not Visible	Not Required	Input	1 = Use PVF SrcQ input for rejection decisions. Default is false.
Cfg_UseInpSrcQPVG	BOOL	Not Visible	Not Required	Input	1 = Use PVG SrcQ input for rejection decisions. Default is false.
Cfg_UseInpSrcQPVH	BOOL	Not Visible	Not Required	Input	1 = Use PVH SrcQ input for rejection decisions. Default is false.
Cfg_HasPVNav	BOOL	Not Visible	Not Required	Input	1 = Tells HMI to enable navigation to a connected PV (Val) object. Default is false.
Cfg_HasMoreObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI an object with more information is available. Default is false.
Cfg_HasNav	SINT	Not Visible	Not Required	Input	Set bits indicate which navigation buttons are enabled .0=PVA, .1=PVB, ..., .7=PVH Default is 0.
Cfg_MinGood	DINT	Not Visible	Not Required	Input	Minimum good inputs for good PV [1..number of "Cfg_Has" inputs]. Default is 2.
Cfg_CalcWhen2	DINT	Not Visible	Not Required	Input	PV calculation when only 2 good inputs: 0 = average, 1 = minimum, 2 = maximum. Default is 0.
Cfg_PVEUMin	REAL	Not Visible	Not Required	Input	PV (Output) minimum for display PV units. Valid = any float. Default is 0.0.
Cfg_PVEUMax	REAL	Not Visible	Not Required	Input	PV (Output) maximum for display PV units. Valid = any float. Default is 100.0.
Cfg_AbsDevLim	REAL	Not Visible	Not Required	Input	Absolute deviation threshold for PV units. Valid = any nonnegative float: reject ouside this deviation from mean. Default is 10.0.
Cfg_OoRHiLim	REAL	Not Visible	Not Required	Input	Out-of-range (fail) high limit for PV units. Valid = any float). Default is 103.958336.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_OoRLoLim	REAL	Not Visible	Not Required	Input	Out-of-range (fail) low limit for PV units. Valid = any float. Default is -2.0833333.
Cfg_OoRDB	REAL	Not Visible	Not Required	Input	Out-of-range (fail) deadbandfor PV units. Valid = any nonnegative float. Default is 0.41666666.
Cfg_PVDecPlcs	SINT	Not Visible	Not Required	Input	Number of decimal places for PV display. Valid = 0,1,2,3,4,5,6. Default is 2.
Cfg_CnfrmReqd	SINT	Not Visible	Not Required	Input	Operator Command Confirmation Required. Represents the type of command confirmation required. 0 = None, 1 = Command confirmation required, 2 = Performer e-signature required, 3 = Performer and approver e-signature required. Default is 0.
PSet_Owner	DINT	Not Visible	Not Required	Input	Program owner request ID (non-zero) or release (zero) (Valid = any integer). Default is 0.
PCmd_Reset	BOOL	Not Visible	Not Required	Input	Program command to reset all alarms requiring Reset. Default is false.
XCmd_Reset	BOOL	Not Visible	Not Required	Input	External command to reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.
XCmd_ResetAckAll	BOOL	Not Visible	Not Required	Input	External command to acknowledge and reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
EnableOut	BOOL	Not Visible	Not Required	Output	Enable Output - System Defined Parameter
Val	REAL	Visible	Not Required	Output	Selected analog PV (including substitute PV, if used) (PV units).
Val_PVA	REAL	Visible	Not Required	Output	Analog value (actual) from input A (PV units).
Val_PVB	REAL	Visible	Not Required	Output	Analog value (actual) from input B (PV units).
Val_PVC	REAL	Visible	Not Required	Output	Analog value (actual) from input C (PV units).
Val_PVD	REAL	Not Visible	Not Required	Output	Analog value (actual) from input D (PV units).
Val_PVE	REAL	Not Visible	Not Required	Output	Analog value (actual) from input E (PV units).
Val_PVF	REAL	Not Visible	Not Required	Output	Analog value (actual) from input F (PV units).
Val_PVG	REAL	Not Visible	Not Required	Output	Analog value (actual) from input G (PV units).
Val_PVH	REAL	Not Visible	Not Required	Output	Analog value (actual) from input H (PV units).
Val_InpPV	REAL	Not Visible	Not Required	Output	Selected PV (PV units).
Val_PVEUMin	REAL	Not Visible	Not Required	Output	Minimum of PV range = Min (Cfg_PVEUMin, Cfg_PVEUMax) (PV units).
Val_PVEUMax	REAL	Not Visible	Not Required	Output	Maximum of PV range = Max (Cfg_PVEUMin, Cfg_PVEUMax) (PV units).

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Out_SmartDvcSts	DINT	Not Visible	Not Required	Output	Status code of a SMART device provided by Inp_SmartDvcASts or Inp_SmartDvcBSts or Inp_SmartDvcCSts or Inp_SmartDvcDSts or Inp_SmartDvcESts or Inp_SmartDvcFSts or Inp_SmartDvcGSts or Inp_SmartDvcHSts. Highest status code selected.
Val_NumPVs	DINT	Not Visible	Not Required	Output	Number of PVs that are currently used in calculating Val_CalcPV.
Sts_Initialized	BOOL	Not Visible	Not Required	Output	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.
Sts_SmartDvcDiagAvailable	BOOL	Not Visible	Not Required	Output	1 = Diagnostics of a SMART device is currently available. Typically used to indicate one of the devices requires action to keep operating as expected.
Sts_PVBad	BOOL	Not Visible	Not Required	Output	1 = At least one input channel PV is bad quality or out of range.
Sts_PVUncertain	BOOL	Not Visible	Not Required	Output	1 = At least one input channel PV value is uncertain quality.
Sts_PVAReject	BOOL	Visible	Not Required	Output	1 = Input A rejected, not used to calculate PV.
Sts_PVBReject	BOOL	Visible	Not Required	Output	1 = Input B rejected, not used to calculate PV.
Sts_PVCReject	BOOL	Visible	Not Required	Output	1 = Input C rejected, not used to calculate PV.
Sts_PVDReject	BOOL	Not Visible	Not Required	Output	1 = Input D rejected, not used to calculate PV.
Sts_PVEReject	BOOL	Not Visible	Not Required	Output	1 = Input E rejected, not used to calculate PV.
Sts_PVFReject	BOOL	Not Visible	Not Required	Output	1 = Input F rejected, not used to calculate PV.
Sts_PVGReject	BOOL	Not Visible	Not Required	Output	1 = Input G rejected, not used to calculate PV.
Sts_PVHReject	BOOL	Not Visible	Not Required	Output	1 = Input H rejected, not used to calculate PV.
SrcQ_IOA	SINT	Not Visible	Not Required	Output	Source and quality of Input A (enumeration).
SrcQ_IOB	SINT	Not Visible	Not Required	Output	Source and quality of Input B (enumeration).
SrcQ_IOC	SINT	Not Visible	Not Required	Output	Source and quality of Input C (enumeration).
SrcQ_IOD	SINT	Not Visible	Not Required	Output	Source and quality of Input D (enumeration).
SrcQ_IOE	SINT	Not Visible	Not Required	Output	Source and quality of Input E (enumeration).
SrcQ_IOF	SINT	Not Visible	Not Required	Output	Source and quality of Input F (enumeration).
SrcQ_IOG	SINT	Not Visible	Not Required	Output	Source and quality of Input G (enumeration).
SrcQ_IOH	SINT	Not Visible	Not Required	Output	Source and quality of Input H (enumeration).
SrcQ_IO	SINT	Not Visible	Not Required	Output	Source and quality of calculated PV (enumeration).
SrcQ	SINT	Not Visible	Not Required	Output	Source and quality of primary Val/Sts (enumeration).
Sts_eSts	SINT	Not Visible	Not Required	Output	Device confirmed status (enum): 0 = PV Good, 1: PV Uncertain, 2: PV Bad.
Sts_eFault	INT	Not Visible	Not Required	Output	Device fault status (enum): 0 = None, 17 = Any Reject, 18 = Min Good, 32 = Fail, 34 = Bad Config.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_eNotify	SINT	Not Visible	Not Required	Output	All alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyAll	SINT	Not Visible	Not Required	Output	All alarm status enumerated values including related objects: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyAnyReject	SINT	Not Visible	Not Required	Output	Any Reject alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyMinGood	SINT	Not Visible	Not Required	Output	Min Good alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_eNotifyFail	SINT	Not Visible	Not Required	Output	Fail alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_UnackAlmCount	DINT	Not Visible	Not Required	Output	Count of unacknowledged alarms.
Sts_MaintByp	BOOL	Not Visible	Not Required	Output	1 = Device has a maintenance bypass function active.
Sts_Err	BOOL	Visible	Not Required	Output	1 = Error in config, see detail bits for reason.
Sts_ErrEU	BOOL	Not Visible	Not Required	Output	1 = Error in config: Cfg_PVEUMax cannot equal Cfg_PVEUMin.
Sts_ErrHas	BOOL	Not Visible	Not Required	Output	1 = Error in config: at least one Cfg_HasPVx must be 1.
Sts_ErrUse	BOOL	Not Visible	Not Required	Output	1 = Error in config: at least one Cfg_UsePVx must be 1.
Sts_ErrMinGood	BOOL	Not Visible	Not Required	Output	1 = Error in config: Cfg_MinGood must be in the range [1..8].
Sts_ErrOoRDB	BOOL	Not Visible	Not Required	Output	1 = Error in configuration: Cfg_OoRDB deadband is < 0.0.
Sts_ErrAlm	BOOL	Not Visible	Not Required	Output	1 = Error in logix tag-based alarm settings.
Sts_Alm	BOOL	Not Visible	Not Required	Output	1 = An alarm is active.
Sts_AlmInh	BOOL	Not Visible	Not Required	Output	1 = An alarm is shelved, disabled or suppressed: display icon.
Sts_AnyReject	BOOL	Not Visible	Not Required	Output	1 = At least one input has been rejected.
Sts_MinGood	BOOL	Not Visible	Not Required	Output	1 = At minimum required number of good inputs, next reject/fail will result in bad PV.
Sts_Fail	BOOL	Not Visible	Not Required	Output	1 = Total signal failure (too many inputs rejected).
Sts_RdyAck	BOOL	Not Visible	Not Required	Output	1 = An alarm is ready to be acknowledged.
Sts_RdyReset	BOOL	Not Visible	Not Required	Output	1 = A latched alarm or shed condition is ready to be reset.
XRdy_Reset	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Reset, enable HMI button.
XRdy_ResetAckAll	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_ResetAckAll, enable HMI button.
Val_Owner	DINT	Not Visible	Not Required	Output	Current object owner ID (0 = not owned).

Private Input Members	Data Type	Description
HMI_BusObjIndex	DINT	HMI bus object index. Default is 0.
OCmd_Reset	BOOL	Operator command to reset all alarms requiring reset. Default is false.

Private Input Members	Data Type	Description
OCmd_ResetAckAll	BOOL	Operator command to reset and acknowledge all alarms. Default is false.

Private Output Members	Data Type	Description
ORdy_Reset	BOOL	1 = Ready for OCmd_Reset (enables HMI button).
ORdy_ResetAckAll	BOOL	1 = Ready for OCmd_ResetAckAll (enables HMI button).

Public InOut Members	Data Type	Description
BusObj	BUS_OBJ	Bus component

Public InOut Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
BusObj	BUS_OBJ	Visible	Required	InOut	Bus component

BUS_OBJ Structure

Members	Data Type	Description
Inp_Cmd	DINT	Input to assert commands
Out_Cmd	DINT	Resultant commands
Inp_CmdLLH	DINT	Input for level normally high commands
Out_CmdLLH	DINT	Resultant line level high commands
Inp_Sts	DINT	Input to assert status
Out_Sts	DINT	Resultant status
Inp_CmdAck	DINT	Input to assert a command acknowledgment
Out_CmdAck	DINT	Resultant command acknowledgments
Inp_SeverityMax	DINT	Input: maximum alarm severity
Out_SeverityMax	DINT	Resultant of maximum alarm severity
Cfg_CmdMask	DINT	Propagation mask for commands
Cfg_CmdLLHMask	DINT	Propagation mask for line level high commands
Cfg_StsMask	DINT	Propagation mask for status
Ref_Index	DINT	Bus array index

Alarms

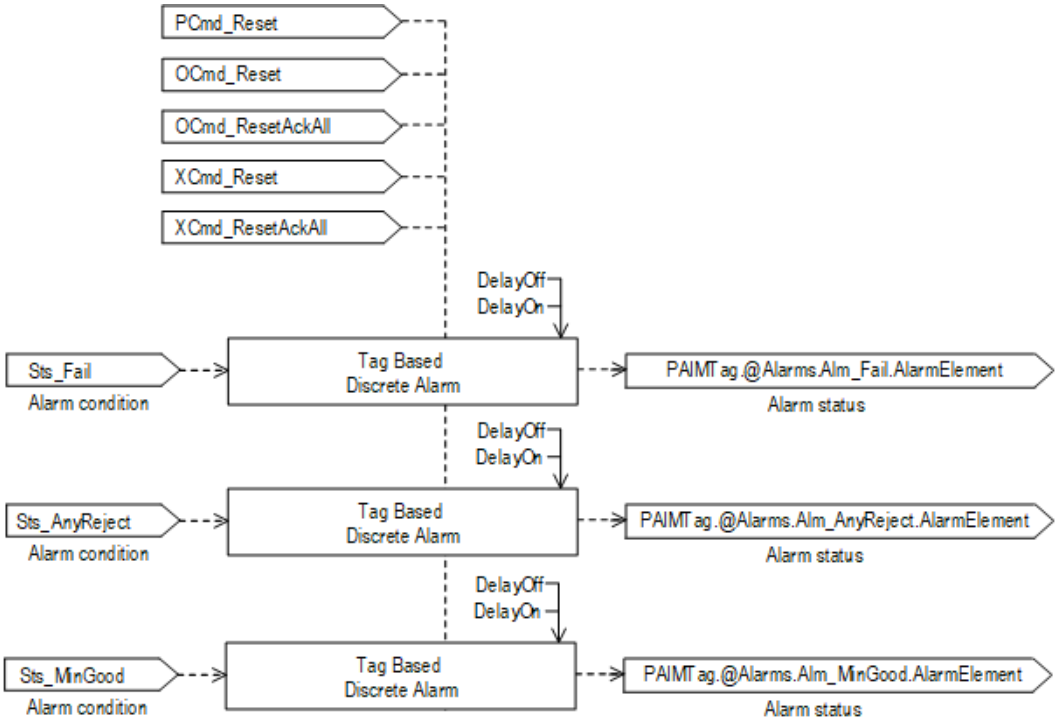
Discrete tag-based alarms are defined for these members.

Member	Alarm name	Description
Sts_Fail	Alm_Fail	Raised when the two sensor PVs are bad or out of range.
Sts_AnyReject	Alm_AnyReject	Raised when any inputs configured to be used are rejected.
Sts_MinGood	Alm_MinGood	Raised when the number of unrejected input is equal to the minimum number required to be good, meaning the next input failure results in a PV failure.

Mark the alarm as used or unused and set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format:

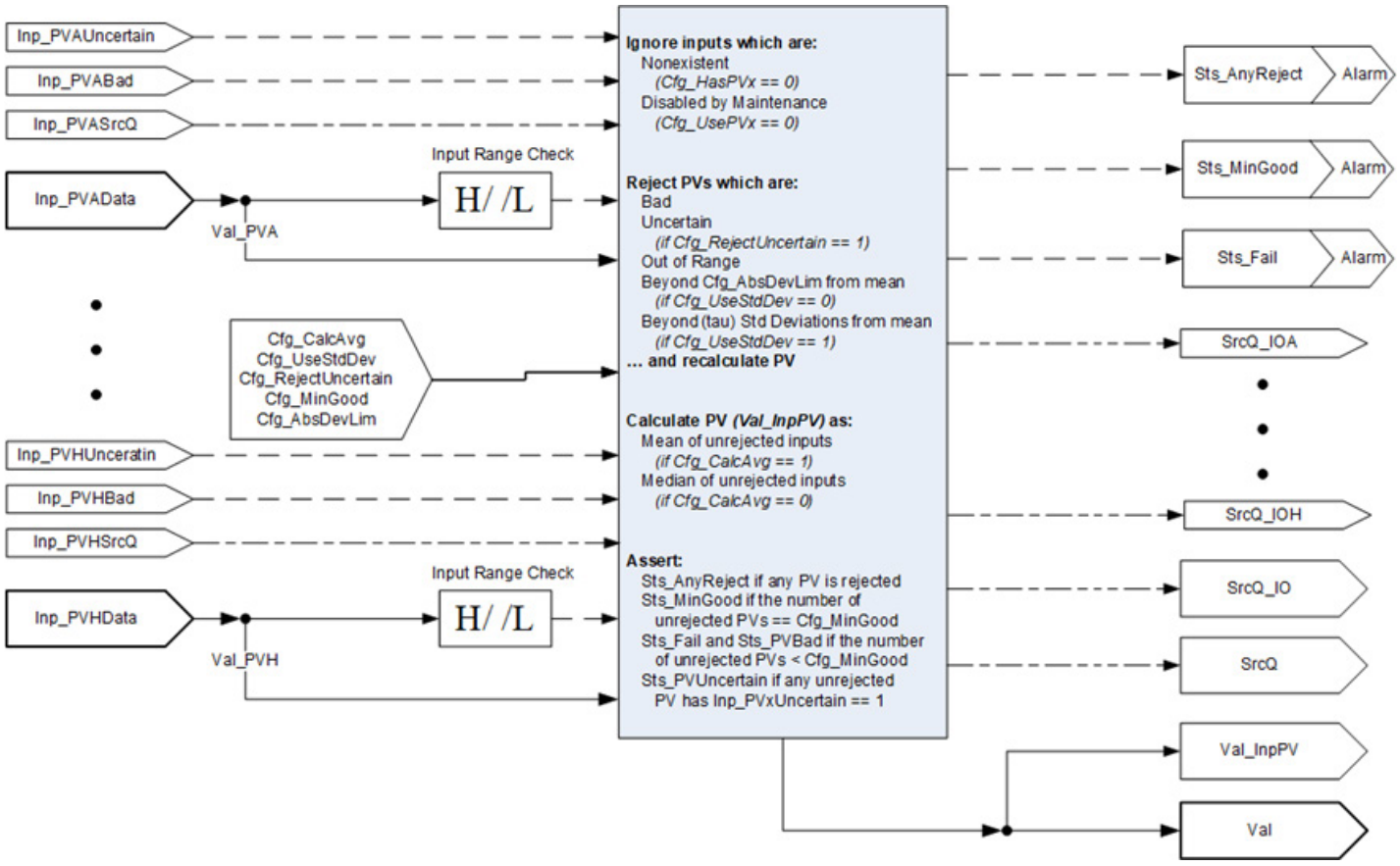
PAIMTag.@Alarms.AlarmName.AlarmElement

There are Program, Operator, and External commands that enable the Reset and Reset & Acknowledge of all alarms of the instruction (Alarm Set) at the same time. This diagram shows how the commands interact with the PAIM instruction.



Operation

This diagram illustrates the functionality of the PAIM instruction:



Initialization

The instruction is normally initialized in the instruction first run. Request re-initialization by setting Inp_InitializeReq = 1. For proper initialization, when adding the instruction while performing an online edit of the code, make sure that Inp_InitializeReq = 1, the default value.

Configuration of Strings for HMI

Configure strings for HMI faceplates, as seen in FactoryTalk View, and for the Logix Designer configuration dialog box. The strings are set to extended properties of tag items. Configure the strings in Logix Designer only.

- Description
- Label for graphic symbol
- Display Library for HMI Faceplate call-up
- Instruction name
- Area name
- URL link
- More Information
- Selected Analog PV engineering units – Units of Val member, Val.@Units.

- Analog input A description – Description of Inp_PVAData member, Inp_PVAData.@Label.
- Analog input B description – Description of Inp_PVBData member, Inp_PVBData.@Label
- Analog input C description – Description of Inp_PVCData member, Inp_PVCData.@Label.
- Analog input D description – Description of Inp_PVDData member, Inp_PVDData.@Label.
- Analog input E description – Description of Inp_PVEData member, Inp_PVEData.@Label.
- Analog input F description – Description of Inp_PVFData member, Inp_PVFData.@Label.
- Analog input G description – Description of Inp_PVGData member, Inp_PVGData.@Label.
- Analog input H description – Description of Inp_PVHData member, Inp_PVHData.@Label.

Monitor the PAIM Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Rung-condition-out clears to false.
Instruction first run	All commands that are automatically cleared on each execution are cleared and ignored. PSet_Owner and Val_Owner are set to 0. Ensure the constants for the Thompson Tau test logic have not been zeroed out. The instruction executes normally.

Condition/State	Action Taken
Rung-condition-in is false	Rung-condition-out is cleared to false. All commands that are automatically cleared on each execution are cleared and ignored. PSet_Owner and Val_Owner are set to 0. Latched alarms are reset. Clear Bus Object commands and HMI Bus Object Index Execute Bus command on receipt for Disable, Enable, Suppress, Unsuppress all alarms. Execute Bus command status propagation. Internal timers are reset.
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	EnableOut is cleared to false.
Instruction first run	All commands that are automatically cleared on each execution are cleared and ignored. PSet_Owner and Val_Owner are set to 0. Ensure the constants for the Thompson Tau test logic have not been zeroed out. The instruction executes normally.
Instruction first scan	See Instruction first run in the Function Block Diagram table.
EnableIn is false	EnableOut is cleared to false. All commands that are automatically cleared on each execution are cleared and ignored. PSet_Owner and Val_Owner are set to 0. Latched alarms are reset. Clear Bus Object commands and HMI Bus Object Index Execute Bus command on receipt for Disable, Enable, Suppress, Unsuppress all alarms. Execute Bus command status propagation. Internal timers are reset.
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

In Structured Text, EnableIn is always true during normal scan. The instruction executes when it is in the control path activated by the logic.

Condition/State	Action Taken
Prescan	See Prescan in the Function Block Diagram table.
Instruction first run	See Instruction first run in the Function Block Diagram table.
EnableIn is true	See EnableIn is true in the Function Block Diagram table.
Postscan	See Postscan in the Function Block Diagram table.

Example

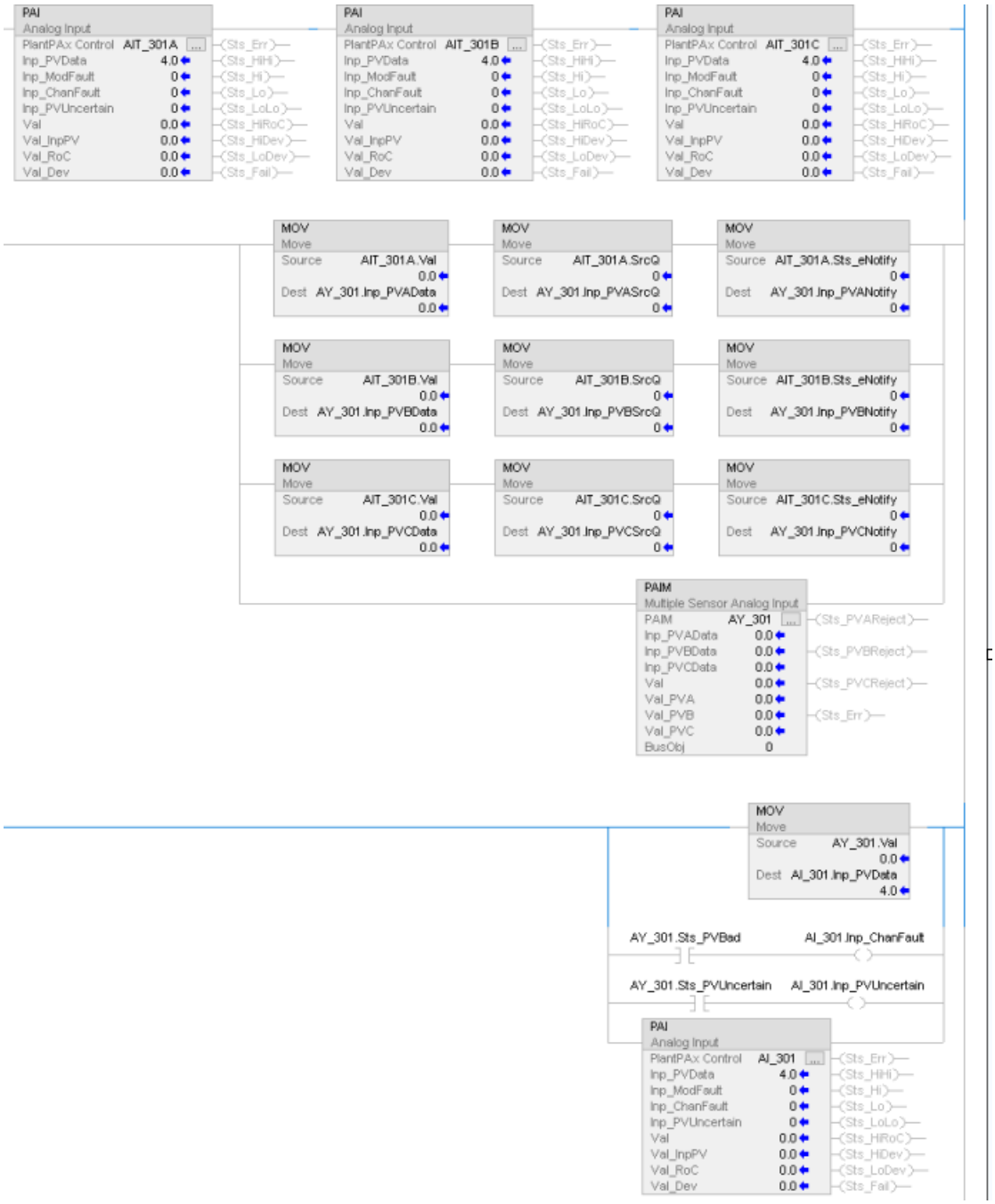
The example uses the PAIM instruction to monitor one analog Process Variable (PV) by using up to eight analog input signals (sensors, transmitters). The PAIM instruction allows you to display a temperature, pressure, level, or other PV on a user interface or use the PV in control logic, and the following apply:

- Have three or more sensors for that PV, for example, six thermocouples.
- Calculate a PV with the mean or median of the sensor input values.

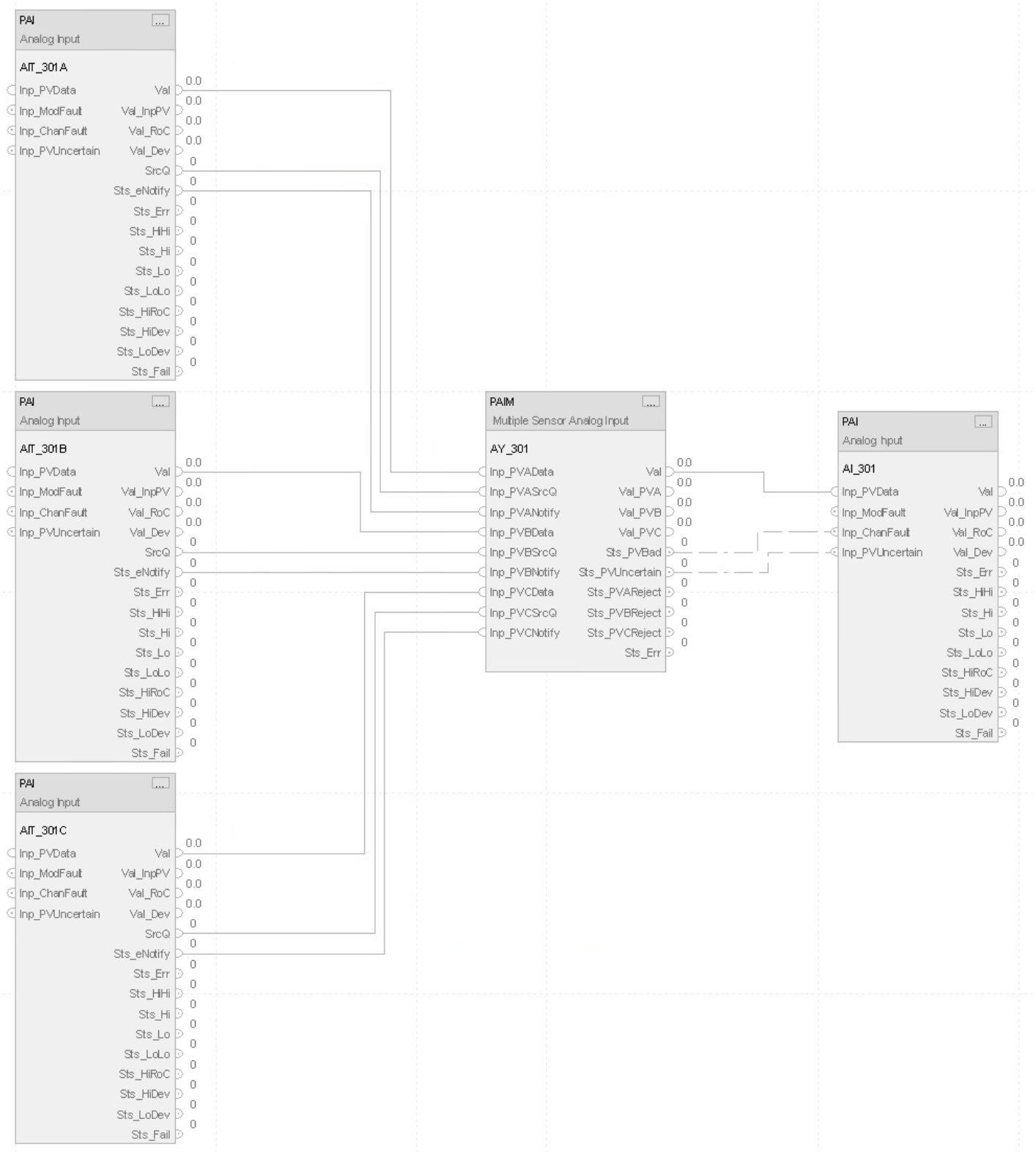
The example uses the PAIM instruction to average multiple sensors for a single PV. In this example, an application uses three analog sensors (A, B, C). The average of these analogs is used elsewhere in logic to control a separate application element.

The Inp_PVAData, Inp_PVBData, and Inp_PVCData parameters are connected to the values from the three analog transmitters. The fault status of each sensor is tied to the bad input of the PAIM (for example, Inp_PVABad). The output parameters Val, Sts_PVBad and Sts_PVUncertain, can then be connected to the Inp_PVData, Inp_PVUncertain and Inp_ChanFault for control.

Ladder Diagram



Function Block Diagram



Structured Text

```
PAI(AIT_301A);
```

```
PAI(AIT_301B);
PAI(AIT_301C);
AY_301.Inp_PVAData := AIT_301A.Val;
AY_301.Inp_PVASrcQ := AIT_301A.SrcQ;
AY_301.Inp_PVANotify := AIT_301A.Sts_eNotify;
AY_301.Inp_PVBData := AIT_301B.Val;
AY_301.Inp_PVBSrcQ := AIT_301B.SrcQ;
AY_301.Inp_PVBNotify := AIT_301B.Sts_eNotify;
AY_301.Inp_PVCData := AIT_301C.Val;
AY_301.Inp_PVCSrcQ := AIT_301C.SrcQ;
AY_301.Inp_PVCNotify := AIT_301C.Sts_eNotify;
PAIM(AY_301);
AI_301.Inp_PVData := AY_301.Val;
AI_301.Inp_ChanFault := AY_301.Sts_PVBad;
AI_301.Inp_PVUncertain := AY_301.Sts_PVUncertain;
PAI(AI_301);
```

See also

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Process Analog Output (PAO)

This information applies to the ControlLogix 5380P and 5580P controllers.

The Process Analog Output (PAO) instruction drives an analog output and checks for alarm conditions. Use the PAO instruction for a channel of an analog output module. Use the PAO instruction with any analog (REAL) signal.

The PAO instruction:

- Monitors one analog output channel for I/O fault input and raises alarm on an I/O fault.
- Operates in Hand, Out of Service, Maintenance, Override, Program, and Operator modes.

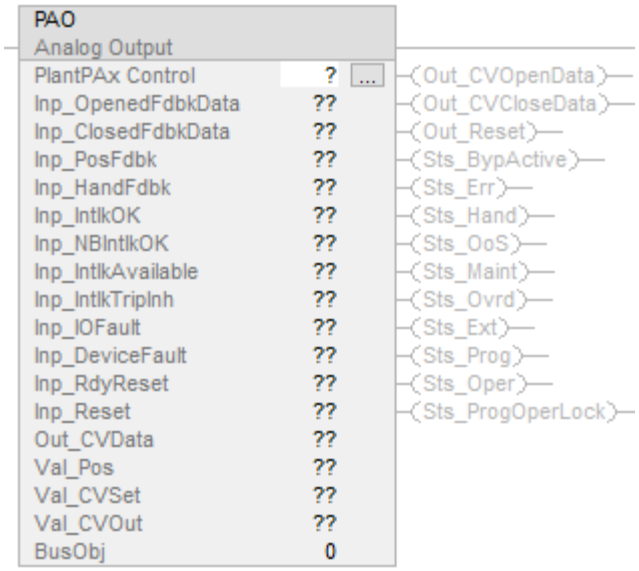
- Provides Operator and Program commands to set an Analog Control Variable (CV, or output) to a specific value. The entered CV is scaled from engineering units to raw (output module) units.
- Monitors bypassable and non-bypassable Interlocks that force the analog output to a specific configured (safe) value or to maintain the current value (configurable).

  Tip: An alarm initiates when an interlock causes the Analog Output CV to change. The PAO instruction enables bypassing interlocks.

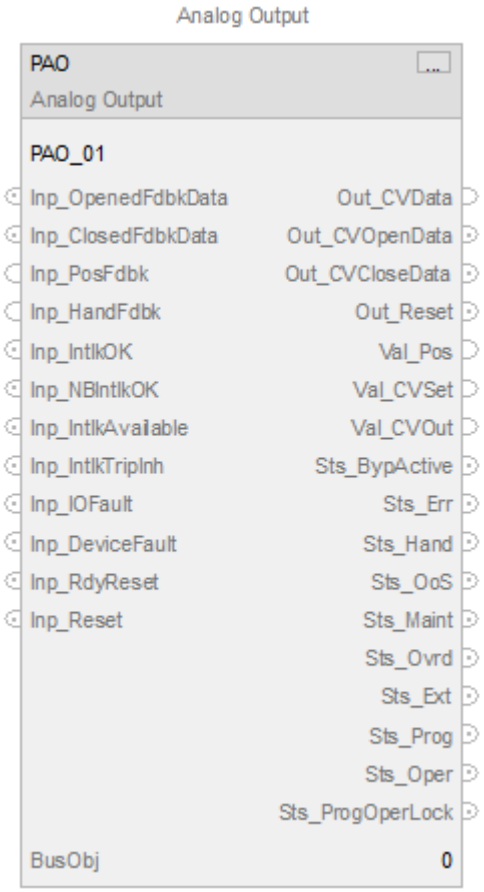
- Allows overriding CV in Override mode.
- Allows analog output ramping with configurable rate of change limits.
- Reads Tieback input (REAL) and a Hand mode request input (BOOL). When Hand mode is asserted, the CV is forced to follow the Tieback value.
- Provides an Available status, when in Program mode and operating normally, for use by higher-level automation logic to determine
- if the logic is able to manipulate the analog output.

Available Languages

Ladder Diagram



Function Block Diagram



Structured Text

PAO(PAO tag, BusObj);

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PlantPAx Control	P_ANALOG_OUTPUT	tag	Data structure required for proper operation of instruction.
BusObj	BUS_OBJ	tag	Bus component

P_ANALOG_OUTPUT Structure

Public members are standard, visible tag members that are programmatically accessible. Private, or hidden, members are used in HMI faceplates and are not programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	Description
EnableIn	BOOL	Enable input. Ladder Diagram: Corresponds to the rung condition. Default is true.
Inp_InitializeReq	BOOL	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request to re-initialize. The instruction clears this operand automatically. Default is true.
Inp_OwnerCmd	DINT	Owner device command. 0 = None, Inp_OwnerCmd.10 = Operator Lock, Inp_OwnerCmd.11 = Operator Unlock, Inp_OwnerCmd.12 = Program Lock, Inp_OwnerCmd.13 = Program Unlock, Inp_OwnerCmd.14 = Acquire Maintenance, Inp_OwnerCmd.15 = Release Maintenance, Inp_OwnerCmd.16 = Acquire External, Inp_OwnerCmd.17 = Release External. Default is 0.
Inp_OpenedFdbkData	BOOL	Feedback from opened limit switch of the device. 1 = Device confirmed opened. Default is false.
Inp_ClosedFdbkData	BOOL	Feedback from closed limit switch of the device. 1 = Device confirmed closed. Default is false.
Inp_PosFdbk	REAL	Feedback from actual device position PV (CV engineering units). Valid any float. Default is 0.0.
Inp_HandFdbk	REAL	CV feedback used in Hand source (CV engineering units). Valid any float. Default is 0.0.
Inp_IntlkOK	BOOL	1 = Bypassable and non-bypassable interlocks OK, analog output can be set. Default is true.
Inp_NBIntlkOK	BOOL	1 = Non-bypassable interlocks OK, analog output can be set if bypassable interlocks are bypassed. Default is true.
Inp_IntlkAvailable	BOOL	1 = Interlock availability OK. Default is false.
Inp_IntlkTripInh	BOOL	1 = Inhibit interlock trip status. Default is false.
Inp_SmartDvcSts	DINT	Current code provided by SMART device on Inp_PosFdbk. The code is copied to Out_SmartDvcSts allowing a user to monitor the device status on HMI for diagnostic lookup purposes. Valid = 0 to maximum positive number. Default is 0.
Inp_SmartDvcDiagAvailable	BOOL	1 = SMART Device diagnostics is available. Typically used to indicate device requires action to keep operating as expected. Default is false.

Public Input Members	Data Type	Description
Inp_IOFault	BOOL	Indicates the IO data is inaccurate. 0 = The IO data is good, 1 = The IO data is bad, causing fault. This input sets Sts_IOFault, if the device is not virtual, which raises IOFault Alarm. Default is false.
Inp_DeviceFault	BOOL	Indicates the device fault (overload, etc.). 0 = The device is good, 1 = The device is bad, causing fault. This input sets Sts_DeviceFault (if the device is not virtual) which raises Device alarm (if the device is not virtual). Default is false.
Inp_Hand	BOOL	1 = Acquire Hand (typically permanently set to local), 0 = Release Hand. Default is false.
Inp_Ovrd	BOOL	1 = Acquire Override (higher priority program logic), 0 = Release Override. Default is false.
Inp_OvrdCV	REAL	CV target in Override (engineering units). Valid any float. Default is 0.0.
Inp_ExtInh	BOOL	1 = Inhibit External acquisition, 0 = Allow External acquisition. Default is false.
Inp_RdyReset	BOOL	1 = Related object, reset by this object, is ready to be reset. Default is false.
Inp_Reset	BOOL	1 = Reset shed latches and cleared alarms. Default is false.
Cfg_AllowDisable	BOOL	1 = Allow Maintenance to disable alarms. Default is true.
Cfg_AllowShelve	BOOL	1 = Allow Operator to shelve alarms. Default is true.
Cfg_StuckTime	REAL	Time with no change in input position while neither feedback from limit switch Closed nor Opened is on to raise stuck status (second). Valid = 0.0 to 2147483.0 seconds. Default is 60.0.
Cfg_HasSmartDvc	BOOL	1 = Enable a button on the HMI that could be used to call up a SMART Device faceplate (Diagnostics). Default is false.
Cfg_SetTrack	BOOL	1 = When the owner is Program the operator settings track the program settings. When the owner is Operator the program settings track the operator settings; and the virtual inputs match the output values (transitions are bumpless), 0 = No tracking. Default is true.
Cfg_ShedHold	BOOL	1 = Hold output on interlock. 0 = Go to Cfg_CVIntlk on interlock. Default is false.
Cfg_SkipRoCLim	BOOL	1 = Skip rate of change limiting in Maintenance or Override and on interlock. Default is false.
Cfg_SetTrackOvrdHand	BOOL	1 = Program/Operator settings track Override/Hand CV. Default is false.
Cfg_FdbkFail	BOOL	1 = Feedback from limit switches is invalid if both feedback inputs are set. 0 = Feedback from limit switches is invalid if both feedback inputs are cleared. Default is true.
Cfg_HasOpenedFdbk	BOOL	1 = Device provides opened feedback signal. Default is false.

Public Input Members	Data Type	Description
Cfg_HasClosedFdbk	BOOL	1 = Device provides closed feedback signal. Default is false.
Cfg_HasPosFdbk	BOOL	1 = Device provides position PV feedback signal. Default is true.
Cfg_UseOpenedFdbk	BOOL	1 = Use device opened feedback for failure checking. Default is false.
Cfg_UseClosedFdbk	BOOL	1 = Use device closed feedback for failure checking. Default is false.
Cfg_UsePosFdbk	BOOL	1 = Use device position PV feedback signal. Default is true.
Cfg_HasCombinedFdbk	BOOL	1 = Device provides opened, closed and position feedback signals to be used. Default is false.
Cfg_UseCombinedFdbk	BOOL	1 = Use device opened, closed and position feedback signals to determine the opened and closed status. The combined signals will be used for the position status. Default is false.
Cfg_HasPulseOut	BOOL	1 = Device provides pulse output (Open, Close). Default is false.
Cfg_HasOutNav	BOOL	1 = Tells HMI to enable navigation to a connected output object. Default is false.
Cfg_OvrdIntlk	BOOL	1 = Override bypasses (ignores) bypassable interlocks. 0 = Override abides by all interlock conditions. Default is false.
Cfg_ShedOnDeviceFault	BOOL	1 = Set output to interlock CV and alarm on Device fault. 0 = Alarm only on Device fault. Default is true.
Cfg_ShedOnIOFault	BOOL	1 = Set output to interlock CV and alarm on I/O fault. 0 = Alarm only on I/O fault. Default is true.
Cfg_CVLoLim	REAL	Minimum CV for limiting (engineering units). Valid any float less than or equal to Cfg_CVHiLim and greater than or equal to Cfg_CVEUMin. Default is 0.0.
Cfg_CVHiLim	REAL	Maximum CV for limiting (engineering units). Valid any float greater than or equal to Cfg_CVLoLim and less than or equal to Cfg_CVEUMax. Default is 100.0.
Cfg_CVRoCIncrLim	REAL	Maximum allowed CV rate of change increasing value (CVEU/second). The CV rate of change is unlimited when increasing if Cfg_CVRoCIncrLim = 0.0. Valid = 0.0 to maximum positive float. Default is 100.0.
Cfg_CVRoCDecrLim	REAL	Maximum allowed CV rate of change decreasing value (engineering units/second). The CV rate of change is unlimited when decreasing if Cfg_CVRoCDecrLim = 0.0. Valid = 0.0 to maximum positive float. Default is 100.0.
Cfg_CVIntlk	REAL	CV target when interlocked, if not Cfg_ShedHold (engineering units). Valid any float. Default is 0.0.
Cfg_CVEUMin	REAL	CV minimum for scaling (engineering units). Valid any float not equal to Cfg_CVEUMax. Default is 0.0.
Cfg_CVEUMax	REAL	CV maximum for scaling (engineering units). Valid any float not equal to Cfg_CVEUMin. Default is 100.0.
Cfg_CVRawMin	REAL	CV minimum for scaling (I/O raw units). Valid any float not equal to Cfg_CVRawMax. Default is 0.0.
Cfg_CVRawMax	REAL	CV maximum for scaling (I/O raw units). Valid any float not equal to Cfg_CVRawMin. Default is 20.0.

Public Input Members	Data Type	Description
Cfg_MaxInactiveCV	REAL	When Val_CVOut is greater than this value (CV engineering units) set Sts_Active (for HMI). Valid any float. Default is 0.0.
Cfg_HiDevLim	REAL	High deviation (CV-PV) deadband limit (engineering units). Valid = 0.0 to maximum positive float. Default is 1.50E+38.
Cfg_LoDevLim	REAL	Low deviation (CV-PV) deadband limit (engineering units). Valid = -maximum positive float to 0.0. Default is -1.50E+38.
Cfg_DevDly	REAL	The minimum time (seconds) the deviation must remain above the upper (Cfg_HiDevLim) or below the lower (Cfg_LoDevLim) limit for the status Sts_Dev to be set. On-delay time is used to avoid unnecessary alarm when the deviation only briefly overshoots Cfg_HiDevLim or undershoots Cfg_LoDevLim. Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_CycleTime	REAL	Open and Close pulse output overall period (seconds). Valid = 0.0 to 2147483.0 seconds. Default is 10.0.
Cfg_OpenRate	REAL	Rate at which device moves when opening (engineering units/second). Valid = 0.0 to maximum positive float. Default is 1.0.
Cfg_CloseRate	REAL	Rate at which device moves when closing (engineering units/second). Valid = 0.0 to maximum positive float. Default is 1.0.
Cfg_MaxOnTime	REAL	Open and Close pulse output maximum On time (seconds). Valid = 0.0 to 2147483.0 seconds. Default is 5.0.
Cfg_MinOnTime	REAL	Open and Close pulse output minimum On time (seconds). Valid = 0.0 to 2147483.0 seconds. Default is 1.0.
Cfg_BumpTime	REAL	Time to bump device open or close (used when device position (PV) feedback is not available) (seconds). Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_DeadTime	REAL	Additional time on first pulse after stop or direction change. Additional pulse time in seconds to overcome friction in the device. Deadtime is added to the open time or close time when the device changes direction or is stopped. Valid = 0.0 to Cfg_MaxOnTime seconds. Default is 0.0.
Cfg_MaxClosedPos	REAL	Position (PV value) above which device (valve) is assumed open if feedback from Opened limit switch is not used. Default is 0.0.
Cfg_HasIntlkObj	BOOL	1 = Tells HMI an interlock object (for example, P_Intlk) is used for Inp_IntlkOK and navigation to the interlock objects faceplate is enabled. **Important:** The name of the interlock object in the controller must be this PAO object's name with the suffix _Intlk. For example, if the PAO instruction has the name PAOut123, then its interlock object must be named PAOut123_Intlk. Default is false.
Cfg_HasOper	BOOL	1 = Operator (unlocked) exists, can be selected. Default is true.
Cfg_HasOperLocked	BOOL	1 = Operator Locked exists, can be selected. Default is true.
Cfg_HasProg	BOOL	1 = Program (unlocked) exists, can be selected. Default is true.
Cfg_HasProgLocked	BOOL	1 = Program Locked exists, can be selected. Default is true.
Cfg_HasExt	BOOL	1 = External exists, can be selected. Default is false.
Cfg_HasMaint	BOOL	1 = Maintenance exists, can be selected. Default is true.

Public Input Members	Data Type	Description
Cfg_HasMaintOoS	BOOL	1 = Maintenance Out of Service exists, can be selected. Default is true.
Cfg_OvrdOverLock	BOOL	1 = Override supersedes Program/Operator Lock, 0 = Do not override Lock. Default is true.
Cfg_ExtOverLock	BOOL	1 = External supersedes Program/Operator Lock, 0 = Do not override Lock. Default is false.
Cfg_ProgPwrUp	BOOL	1 = Power up to Program, 0 = Power up to Operator. Default is false.
Cfg_ProgNormal	BOOL	Normal Source: 1 = Program if no requests; 0 = Operator if no requests. Default is false.
Cfg_PCmdPriority	BOOL	Command priority. 1 = Program commands win, 0 = Operator commands win. Default is false.
Cfg_PCmdProgAsLevel	BOOL	1 = PCmd_Prog used as a Level. Default is false.
Cfg_PCmdLockAsLevel	BOOL	1 = PCmd_Lock used as a Level (1 = Lock, 0 = Unlock). Default is false.
Cfg_ExtAcqAsLevel	BOOL	1 = XCmd_Acq used as Level (1 = Acquire, 0 = Release). Default is false.
Cfg_CVDecPlcs	SINT	Number of decimal places for control variable display. Valid = 0 to 6. Default is 2.
Cfg_CnfrmReqd	SINT	Operator command confirmation required. Represents the type of command confirmation required. 0 = None, 1 = Command confirmation required, 2 = Performer e-signature required, 3 = Performer and approver e-signature required. Default is 0.
Cfg_CVPwrUpSel	SINT	Selection of power up CV. 0 = Use Cfg_CVPwrUp, 1 = No change (from last power down), 2 = Use Inp_PosFdbk if available (Cfg_CVPwrUp otherwise). Default is 0.
Cfg_CVPwrUp	REAL	CV initial value used on power up (engineering units). Valid any float. Default is 0.0.
Cfg_HasMoreObj	BOOL	1 = Tells HMI an object with more info is available. Default is false.
Cfg_HasPosFdbkNav	BOOL	1 = Tells HMI to enable navigation to a connected positive feedback object. Default is false.
Cfg_HasHistTrend	SINT	Has historical trend. This enables navigation to the device historical trend faceplate from the HMI. 0 = No external historical trend, 1 = Datalog historical trend, 2 = Historian historical trend. Default is 0.
PSet_CV	REAL	Program setting of controlled variable, output (engineering units). Valid any float. Default is 0.0.

Public Input Members	Data Type	Description
PSet_Owner	DINT	Program owner request ID (non-zero) or release (zero). Default is 0.
PCmd_Oper	BOOL	Program command to select Operator (Program to Operator). The instruction clears this operand automatically. Default is false.
PCmd_Prog	BOOL	Program command to select Program (Operator to Program). The instruction clears this operand automatically. Default is false.
PCmd_Lock	BOOL	Program command to lock Program (disallow Operator). The instruction clears this operand automatically if Cfg_PCmdLockAsLevel = 0. Default is false.
PCmd_Unlock	BOOL	Program command to unlock Program (allow Operator to acquire). The instruction clears this operand automatically. Default is false.
PCmd_Normal	BOOL	Program command to select Normal command source (Operator or Program). The instruction clears this operand automatically. Default is false.
PCmd_Reset	BOOL	Program command to reset all alarms and latched shed conditions requiring reset. The instruction clears this operand automatically. Default is false.
PCmd_Physical	BOOL	Program command to select Physical device operation (not simulated). The instruction clears this operand automatically. Default is false.
PCmd_Virtual	BOOL	Program command to select Virtual (simulated) device operation. The instruction clears this operand automatically. Default is false.
XSet_CV	REAL	External setting of controlled variable, output (engineering units). Default is 0.0.
XCmd_Acq	BOOL	External command to acquire ownership (Operator/Program/Override/Maintenance to External). The instruction clears this operand automatically if Cfg_ExtAcqAsLevel = 0. Default is false.
XCmd_BumpClose	BOOL	External command to bump device closed (used when device position feedback is not available). Default is false.
XCmd_BumpOpen	BOOL	External command to bump device open (used when device position feedback is not available). Default is false.
XCmd_Rel	BOOL	External command to release ownership if Cfg_ExtAcqAsLevel = 0 (External to Operator/Program/Override/Maintenance). The instruction clears this operand automatically. Default is false.
XCmd_Reset	BOOL	External command to reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.
XCmd_ResetAckAll	BOOL	External command to acknowledge and reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.

Public Output Members	Data Type	Description
EnableOut	BOOL	Enable Output. This output state always reflects EnableIn input state.
Out_CVData	REAL	CV output in raw (I/O Card) units. Extended properties of this member: Engineering Unit - Raw units (text) used for the analog output.

Public Output Members	Data Type	Description
Out_CVOpenData	BOOL	1 = Pulse output to drive device open.
Out_CVCloseData	BOOL	1 = Pulse output to drive device closed.
Val_Dev	REAL	Calculated deviation value (CV-PV) (in engineering units).
Val_Pos	REAL	Device actual position (PV) from feedback (in engineering units).
Val_CVSet	REAL	Value of selected CV setting before rate limiting, in engineering units.
Val_CVOut	REAL	Value of CV Output after optional rate limiting, in engineering units. Extended Properties of this member: Engineering Unit - Engineering units (text) used for the analog output.
Val_CVEUMin	REAL	Minimum of scaled range in engineering units = MIN (Cfg_CVEUMin, Cfg_CVEUMax).
Val_CVEUMax	REAL	Maximum of scaled range in engineering units = MAX (Cfg_CVEUMin, Cfg_CVEUMax).
Out_SmartDvcSts	DINT	Status code of a SMART Device provided by Inp_SmartDvcSts. Out_SmartDevSts is a copy of Inp_SmartDvcSts.
Out_OwnerSts	DINT	Status of command source, owner command handshake and ready status. 0 = None, Out_OwnerSts.10 = Operator Lock, Out_OwnerSts.11 = Operator Unlock, Out_OwnerSts.12 = Program Lock, Out_OwnerSts.13 = Program Unlock, Out_OwnerSts.14 = Acquire Maintenance, Out_OwnerSts.15 = Release Maintenance, Out_OwnerSts.16 = Acquire External, Out_OwnerSts.17 = Release External, Out_OwnerSts.18 = Has Maintenance, Out_OwnerSts.19 = External Override Lock, Out_OwnerSts.20 = Has External, Out_OwnerSts.21 = Has Operator, Out_OwnerSts.22 = Has Program, Out_OwnerSts.30 = Not Ready.
Sts_Initialized	BOOL	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.
Sts_SmartDvcDiagAvailable	BOOL	1 = Diagnostics of a SMART Device is currently available. Typically used to indicate device requires action to keep operating as expected. Sts_SmartDvcDiagAvailable is a copy of Inp_SmartDvcDiagAvailable.
Sts_CVInfNaN	BOOL	1 = Selected CV is infinite or not a number (1.$, 1.#NaN).
Sts_PosInfNaN	BOOL	1 = Inp_PosFdbk is infinite or not a number (1.$, 1.#NaN).
Sts_BumpOpen	BOOL	1 = Bump Open requested or active.
Sts_BumpClose	BOOL	1 = Bump Close requested or active.
Sts_PosStuck	BOOL	1 = Position is stuck (unchanging) while neither feedback from limit switch Closed nor Opened is on.
Sts_Ramping	BOOL	1 = CV is ramping to target.
Sts_Clamped	BOOL	1 = CV set being clamped at Low or High Limit.
Sts_WindupHi	BOOL	1 = Analog output winding up High, to Inp_WindupHi of the master controller.
Sts_WindupLo	BOOL	1 = Analog output winding up Low, to Inp_WindupLo of the master controller.
Sts_SkipRoCLim	BOOL	1 = Rate of change limiting was skipped this scan (Maintenance, Override, Interlock, Hand).
Sts_Active	BOOL	1 = CV is greater than Cfg_MaxInactiveCV, show graphic symbol as Active.
Sts_FdbkFail	BOOL	1 = Feedbacks are in an invalid state.
Sts_Virtual	BOOL	1 = The instruction treats the device as virtual. The instruction acts as normal but the output is kept de-energized (Out_CVData=0). 0 = The instruction operates the device normally. Sts_Virtual is a copy of Inp_Virtual.

Public Output Members	Data Type	Description
SrcQ_IO	SINT	Source and quality of primary input or output (enumerated): 0 = Good, live, confirmed good, 1 = Good, live, assumed good, 2 = Good, no feedback, assumed good, 8 = Test, virtualized, 9 = Test, loopback, 10 = Test, manually entered, 16 = Uncertain, live, off-spec, 17 = Uncertain, substituted at device or bus, 18 = Uncertain, substituted at instruction, 19 = Uncertain, using last known good, 20 = Uncertain, using replacement value, 32 = Bad, signal failure, 33 = Bad, channel fault, 34 = Bad, module or communication fault, 35 = Bad, invalid configuration.
SrcQ	SINT	Source and quality of primary value or status (enumerated): 0 = Good, live, confirmed good, 1 = Good, live, assumed good, 2 = Good, no feedback, assumed good, 8 = Test, virtualized, 9 = Test, loopback, 10 = Test, manually entered, 16 = Uncertain, live, off-spec, 17 = Uncertain, substituted at device or bus, 18 = Uncertain, substituted at instruction, 19 = Uncertain, using last known good, 20 = Uncertain, using replacement value, 32 = Bad, signal failure, 33 = Bad, channel fault, 34 = Bad, module or communication fault, 35 = Bad, invalid configuration.
Sts_bFdbk	SINT	Device feedback: 0 = None, Sts_bFdbk.0: Moving, Sts_bFdbk.1: Closed, Sts_bFdbk.2: Opened, Sts_bFdbk.3: Failure, Sts_bFdbk.4: Stuck.
Sts_bSts	SINT	Device status: 0 = At target, Sts_bSts.0: Ramping down, Sts_bSts.1: Ramping up, Sts_bSts.2: Clamped at minimum, Sts_bSts.3: Clamped at maximum, Sts_bSts.4: Out of Service, Sts_bSts.5: Bump open, Sts_bSts.6: Bump close.

Public Output Members	Data Type	Description
Sts_bFault	SINT	Device fault status: 0 = None, Sts_bFault.0: Feedback fault, Sts_bFault.1: IO fault, Sts_bFault.2: Device fault, Sts_bFault.3: Configuration error.
Sts_eNotify	SINT	Alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyAll	SINT	Alarm status enumerated values including related objects: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyIOFault	SINT	IOFault alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyDeviceFault	SINT	DeviceFault alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.

Public Output Members	Data Type	Description
Sts_eNotifyDev	SINT	Deviation alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyIntlkTrip	SINT	IntlkTrip alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_UnackAlmCount	DINT	Count of unacknowledged alarms.
Sts_eSrc	INT	The current command source enumerated values: 0 = Logic not in use, 4 = Hand, 8 = Maintenance, 16 = Override, 32 = Program, 33 = Program locked, 34 = Program by default (Normal), 64 = Operator, 65 = Operator locked, 66 = Operator by default (Normal), 128 = Maintenance Out of Service, 129 = Programmed Out of Service (rung false), 256 = External.
Sts_bSrc	INT	Active selection bitmap for HMI totem pole with command source request selection: Sts_bSrc.0: Hand, Sts_bSrc.1: Programmed Out of Service (rung false), Sts_bSrc.2: Maintenance Out of Service, Sts_bSrc.3: Maintenance, Sts_bSrc.4: Override, Sts_bSrc.5: External, Sts_bSrc.6: Program locked, Sts_bSrc.7: Program, Sts_bSrc.8: Operator locked, Sts_bSrc.9: Operator.
Sts_Available	BOOL	1 = Analog output available for control by automation (Program).
Sts_Bypass	BOOL	1 = Bypassable interlocks are bypassed.
Sts_BypActive	BOOL	1 = Interlock bypassing active (bypassed or maintenance).

Public Output Members	Data Type	Description
Sts_MaintByp	BOOL	1 = Device has a maintenance bypass function active.
Sts_NotRdy	BOOL	1 = Device is not ready, see detail bits (Sts_Nrdyxxx) for reason.
Sts_NrdyOoS	BOOL	1 = Device is not ready: Device disabled by Maintenance.
Sts_NrdyCfgErr	BOOL	1 = Device is not ready: Configuration Error.
Sts_NrdyIntlk	BOOL	1 = Device is not ready: Interlock Not OK.
Sts_NrdyIOFault	BOOL	1 = Device is not ready: IO Fault (Shed requires Reset).
Sts_Err	BOOL	1 = Error in configuration: see detail bits (Sts_Errxxx) for reason.
Sts_ErrCVRaw	BOOL	1 = Error in configuration: Raw output scaling Min = Max.
Sts_ErrCVEU	BOOL	1 = Error in configuration: Scaled CV EU Min = Max.
Sts_ErrCVRoCDecrLim	BOOL	1 = Error in configuration: Invalid decreasing rate of change.
Sts_ErrCVRoCIncrLim	BOOL	1 = Error in configuration: Invalid increasing rate of change.
Sts_ErrLimit	BOOL	1 = Error in configuration: CV High Limit < CV Low Limit.
Sts_ErrHiDevLim	BOOL	1 = Error in configuration: Cfg_HiDevLim.
Sts_ErrLoDevLim	BOOL	1 = Error in configuration: Cfg_LoDevLim.
Sts_ErrDevDly	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrCycleTime	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrOpenRate	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrCloseRate	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrStuckTime	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrMaxOnTime	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrMinOnTime	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrBumpTime	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrDeadTime	BOOL	1 = Invalid timer preset (use 0.0 to 2147483.0).
Sts_ErrCmdCnfrmTimeOutTime	BOOL	1 = Error in configuration: Command confirmation timer preset (use 0.0 to 2147483.0).
Sts_ErrAlm	BOOL	1 = Error in Logix Tag-based alarm settings.
Sts_Hand	BOOL	1 = Hand is selected (supersedes OoS, Maintenance, Override, External, Program, Operator).
Sts_OoS	BOOL	1 = Out of Service is selected (supersedes Maintenance, Override, External, Program, Operator).
Sts_Maint	BOOL	1 = Maintenance is selected (supersedes Override, External, Program, Operator).
Sts_Ovrd	BOOL	1 = Override is selected (supersedes External, Program, Operator).
Sts_Ext	BOOL	1 = External is selected (supersedes Program, Operator).
Sts_Prog	BOOL	1 = Program is selected.
Sts_ProgLocked	BOOL	1 = Program is selected and locked.
Sts_Oper	BOOL	1 = Operator is selected.
Sts_OperLocked	BOOL	1 = Operator is selected and locked.
Sts_ProgOperSel	BOOL	Program/Operator selection (latch) state: 1 = Program, 0 = Operator.
Sts_ProgOperLock	BOOL	Program/Operator lock (latch) state, 1 = Locked, 0 = Unlocked.
Sts_Normal	BOOL	1 = Selection equals the Normal (Program or Operator).
Sts_ExtReqInh	BOOL	1 = External request inhibited, cannot get to External from current state.
Sts_ProgReqInh	BOOL	1 = Program request inhibited, cannot get to Program from current state.
Sts_MAcqRcvd	BOOL	1 = Maintenance acquire command received this scan.
Sts_Alm	BOOL	1 = An alarm is active.
Sts_AlmInh	BOOL	1 = An alarm is shelved or disabled.
Sts_IOFault	BOOL	1 = IO Fault Status Bad. 0 = OK. There is a predefined default discrete Logix tag-based alarm for the status. Set standard configuration members of the discrete Logix tag-based alarm. Access alarm elements using this format: PAOTag.@Alarms.Alm_IOFault.AlarmElement

Public Output Members	Data Type	Description
Sts_DeviceFault	BOOL	Device Fault status: 1 = Bad, 0 = OK. There is a predefined default discrete Logix tag-based alarm for the status. Set standard configuration members of the discrete Logix tag-based alarm. Access alarm elements using this format: PAOTag.@Alarms.Alm_DeviceFault.AlarmElement
Sts_Dev	BOOL	1 = Deviation (CV-PV) out of limits, 0 = Deviation within limits. There is a predefined default discrete Logix tag-based alarm for the status. Set standard configuration members of the discrete Logix tag-based alarm. Access alarm elements using this format: PAOTag.@Alarms.Alm_Dev.AlarmElement
Sts_IntlkTrip	BOOL	1 = Status: CV held or forced by interlock NOT OK. There is a predefined default discrete Logix Tag based alarm for the status. Set standard configuration members of the discrete Logix Tag based alarm. Access alarm elements using this format: PAOTag.@Alarms.Alm_IntlkTrip.AlarmElement
Sts_CnfrmOperCmdReq	BOOL	1 = Operator command request is awaiting confirmation.
Sts_CnfrmOperSPReq	BOOL	1 = Operator set point request is awaiting confirmation.
Sts_RdyAck	BOOL	1 = An alarm is ready to be acknowledged.
Sts_RdyReset	BOOL	1 = A latched alarm or shed condition is ready to be reset.
XRdy_Acq	BOOL	1 = Ready for XCmd_Acq, enable HMI button.
XRdy_Rel	BOOL	1 = Ready for XCmd_Rel, enable HMI button.
XRdy_Reset	BOOL	1 = Ready for XCmd_Reset, enable HMI button.
XRdy_ResetAckAll	BOOL	1 = Ready for XCmd_ResetAckAll, enable HMI button.
Val_Owner	DINT	Current Object Owner ID (0 = not owned).

Private Input Members	Data Type	Description
OSet_CV	REAL	Operator setting of controlled variable (output) in engineering units. Default = 0.
MCmd_Bypass	BOOL	Maintenance command to bypass all bypassable interlocks. The instruction clears this operand automatically. Default is false.
MCmd_Check	BOOL	Maintenance command to check (not bypass) all interlocks. The instruction clears this operand automatically. Default is false.
MCmd_OoS	BOOL	Maintenance command to select Out of Service. The instruction clears this operand automatically. Default is false.
MCmd_IS	BOOL	Maintenance command to select In Service. The instruction clears this operand automatically. Default is false.
MCmd_Acq	BOOL	Maintenance command to acquire ownership (Operator/Program/External/Override to Maintenance). The instruction clears this operand automatically. Default is false.
MCmd_Rel	BOOL	Maintenance command to release ownership (Maintenance to Operator/Program/External/Override). The instruction clears this operand automatically. Default is false.
OCmd_BumpClose	BOOL	Operator Command to bump device closed (used when device position feedback is not available).

Private Input Members	Data Type	Description
OCmd_BumpOpen	BOOL	Operator Command to bump device open (used when device position feedback is not available).
OCmd_Oper	BOOL	Operator command to select Operator (Program to Operator). The instruction clears this operand automatically. Default is false.
OCmd_Prog	BOOL	Operator command to select Program (Operator to Program). The instruction clears this operand automatically. Default is false.
OCmd_Lock	BOOL	Operator command to lock Operator (disallow Program). The instruction clears this operand automatically. Default is false.
OCmd_Unlock	BOOL	Operator command to unlock or release (allow Program to acquire) ownership. The instruction clears this operand automatically. Default is false.
OCmd_Normal	BOOL	Operator command to select Normal (Operator or Program). The instruction clears this operand automatically. Default is false.
OCmd_Reset	BOOL	Operator command to reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.
OCmd_ResetAckAll	BOOL	Operator command to acknowledge and reset all alarms and latched shed conditions. The use of OCmd_ResetAckAll is restricted to HMI. The instruction clears this operand automatically. Default is false.
OCmd_CmdCncl	BOOL	Operator command to cancel command request. The instruction clears this operand automatically. Default is false.
OCmd_CmdCnfrm	BOOL	Operator command to confirm command request. The instruction clears this operand automatically. Default is false.

Private Output Members	Data Type	Description
MRdy_Bypass	BOOL	1 = Ready to receive MCmd_Bypass, enable data entry field.
MRdy_Check	BOOL	1 = Ready to receive MCmd_Check, enable data entry field.
MRdy_OoS	BOOL	1 = Ready for MCmd_OoS, enable HMI button.
MRdy_IS	BOOL	1 = Ready for MCmd_IS, enable HMI button.
MRdy_Acq	BOOL	1 = Ready for MCmd_Acq, enable HMI button.
MRdy_Rel	BOOL	1 = Ready for MCmd_Rel, enable HMI button.
ORdy_Oper	BOOL	1 = Ready for OCmd_Oper, enable HMI button.
ORdy_Prog	BOOL	1 = Ready for OCmd_Prog, enable HMI button.
ORdy_Lock	BOOL	1 = Ready for OCmd_Lock, enable HMI button.
ORdy_Unlock	BOOL	1 = Ready for OCmd_Unlock, enable HMI button.
ORdy_Normal	BOOL	1 = Ready for OCmd_Normal, enable HMI button.
ORdy_BumpClose	BOOL	1 = Ready for OCmd_BumpClose, enable HMI button.
ORdy_BumpOpen	BOOL	1 = Ready for OCmd_BumpOpen, enable HMI button.
ORdy_CV	BOOL	1 = Ready to receive OSet_CV (enables data entry field).
ORdy_Reset	BOOL	1 = At least one alarm or shed condition requires reset.
ORdy_ResetAckAll	BOOL	1 = At least one alarm or latched shed condition requires reset or acknowledgement.

Alarms

Discrete Logix Tag based alarms are defined for the following members.

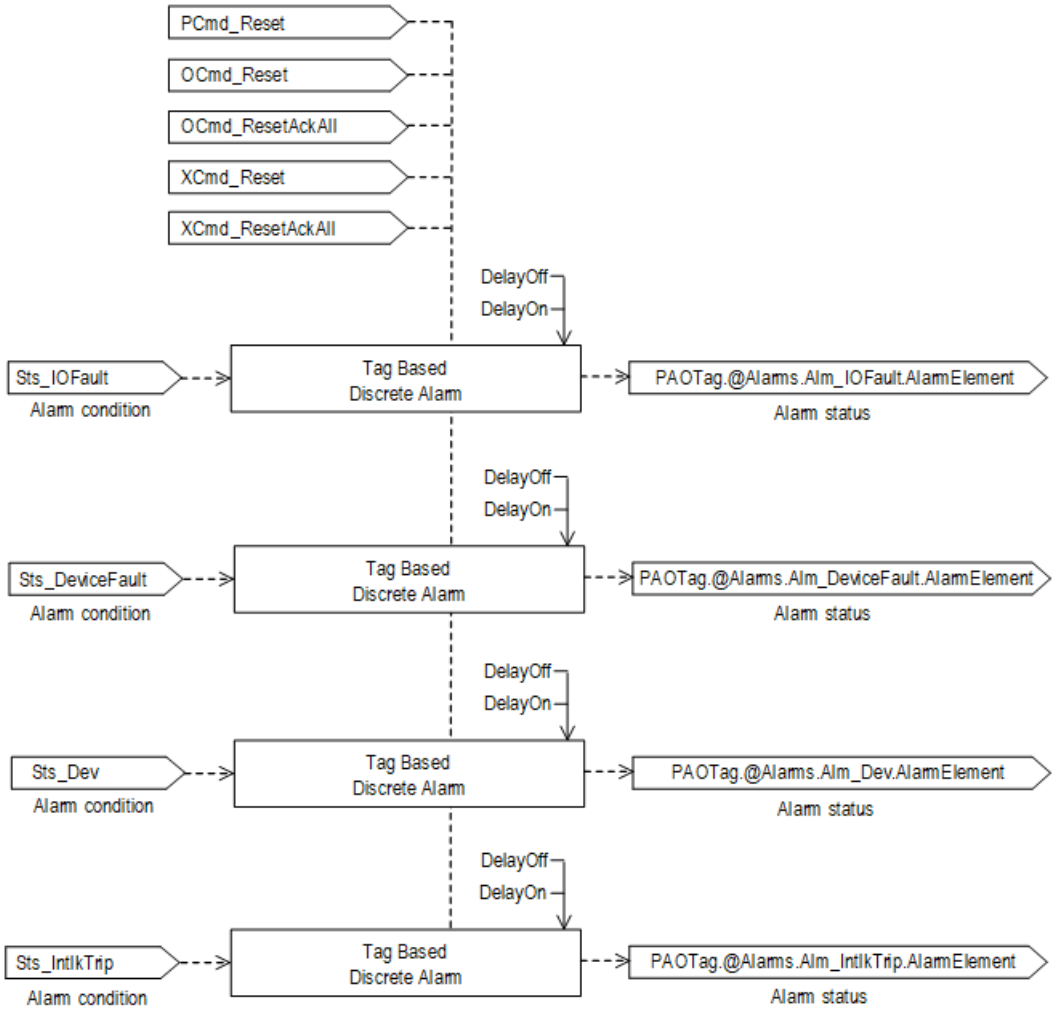
Member	Alarm Name	Description
Sts_IOFault	Alm_IOFault	IO Failure. Raised when the Inp_IOFault input is true. Use this input to indicate to the instruction that a connection with the module is in fault. This input also indicates if a module reports field power loss/no load/short circuit is occurring for its I/O. If the I/O Fault is configured as a shed fault, the device is commanded Off and cannot be commanded to another state until reset. The alarm condition is not raised when in Virtual.
Sts_DeviceFault	Alm_DeviceFault	Device Confirmed Failure. Raised when the Inp_DeviceFault input is true. The Device fault alarm condition is not raised when in Virtual.
Sts_Dev	Alm_Dev	Deviation alarm. Raised when the difference between desired device position and actual device position is greater than High limit or lower than Low limit., i.e. if the following is true: ((Val_CVOut-Val_Pos) > Cfg_HiDevLim) OR ((Val_CVOut-Val_Pos) < Cfg_LoDevLim)
Sts_IntlkTrip	Alm_IntlkTrip	Interlock Trip alarm. Raised when an interlock not-OK condition causes the device to transition from the On state or a pulsing state to the Off state. If interlocks are not bypassed, a bypassable interlock or a non-bypassable interlock not-OK condition initiates an interlock trip. If interlocks are bypassed, only a non-bypassable interlock not-OK condition initiates an interlock trip.

Mark the alarm as used or unused and set standard configuration members of the discrete Logix Tag based alarm. Access alarm elements using this format:

PAOTag.@Alarms.AlarmName.AlarmElement

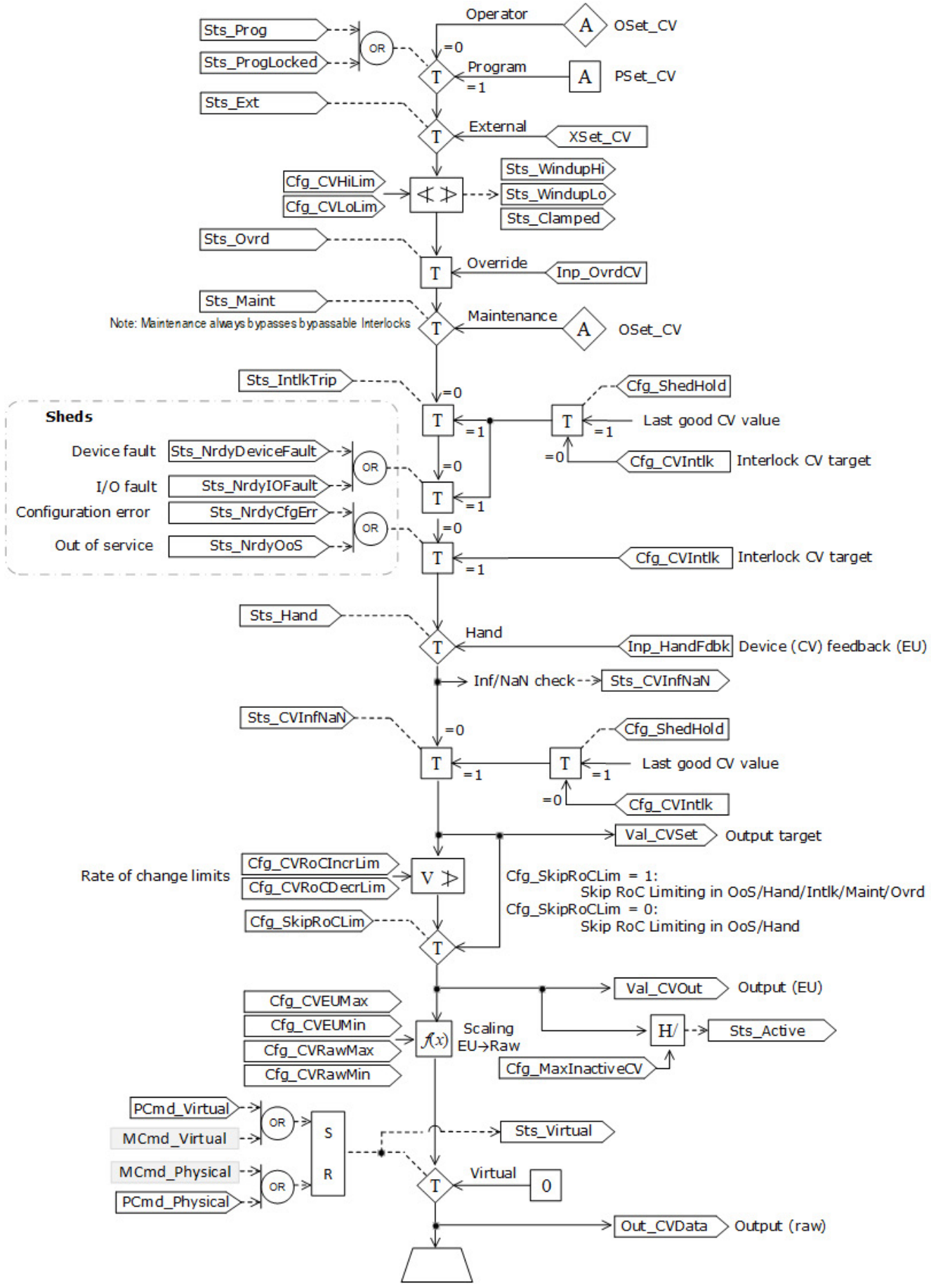
The Program commands for each alarm enable users to Acknowledge, Suppress, Unsupress and Unshelve the Alarm. These commands are propagated to corresponding commands (ProgAck, ProgSuppress, ProgUnsupress, ProgUnshelve) of the tag-based alarm.

There are also Program commands that enable users to Acknowledge, Reset, Suppress and Unsuppress all alarms of the instruction or an alarm set at the same time.

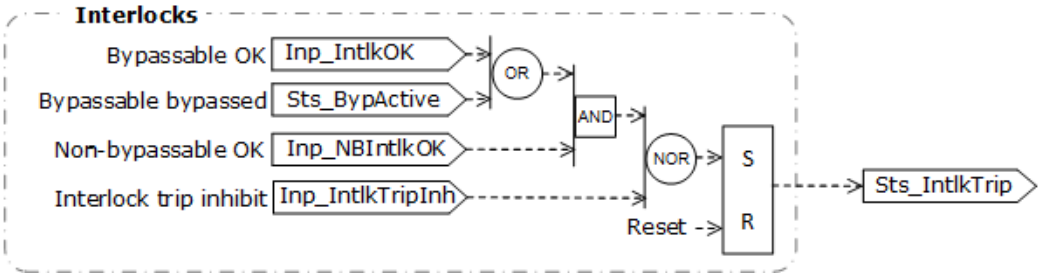


Operation

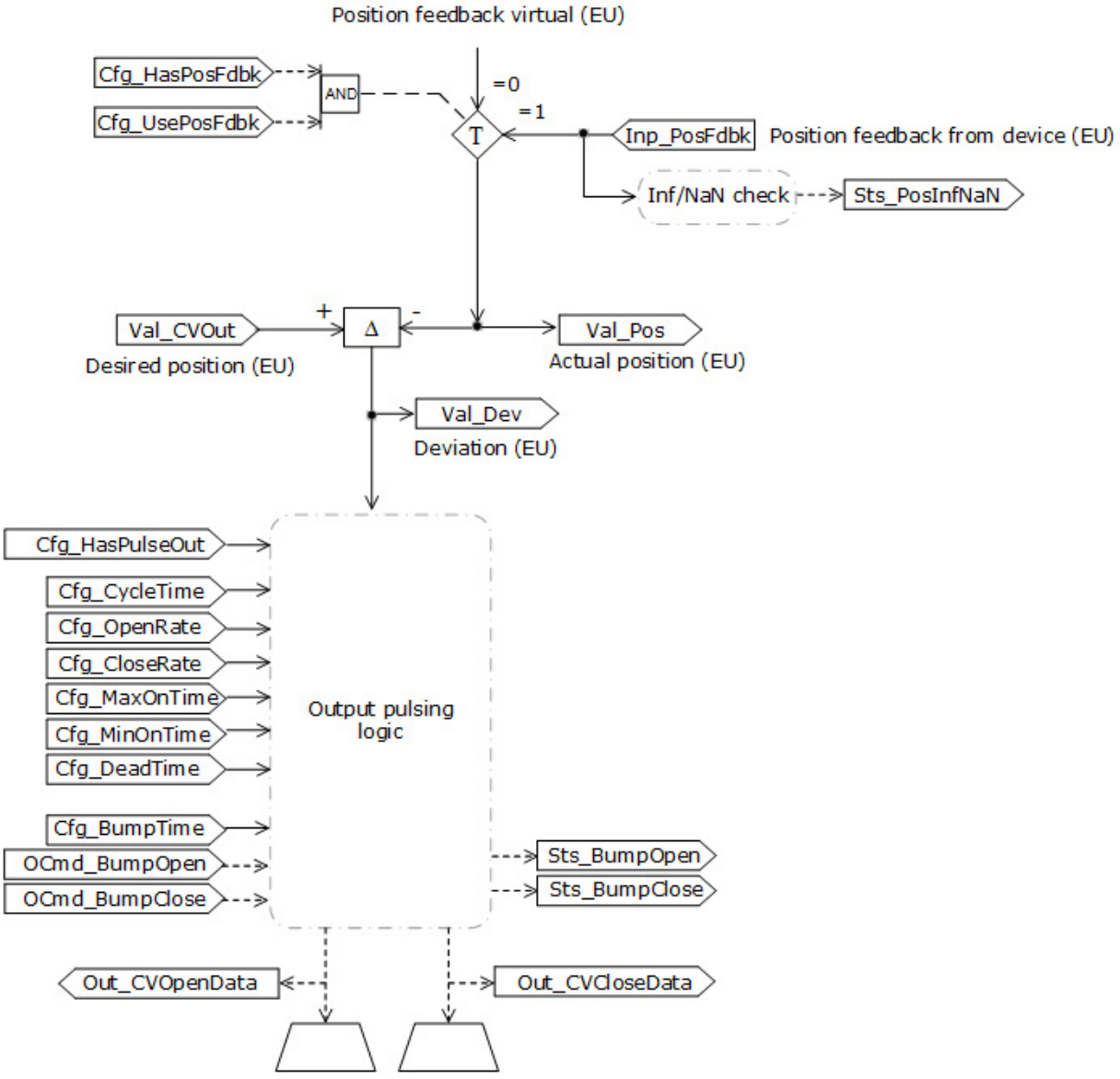
The following diagram illustrates the functionality of the PAO instruction:


Sts_Prog
Sts_ProgLocked
OR
Operator
A
OSet_CV
=0
T
Program
=1
A
PSet_CV
Sts_Ext
T
External
XSet_CV
Cfg_CVHiLim
Cfg_CVLoLim
Sts_WindupHi
Sts_WindupLo
Sts_Clamped
Sts_Ovrd
T
Override
Inp_OvrdCV
Sts_Maint
Note: Maintenance always bypasses bypassable Interlocks
T
Maintenance
A
OSet_CV
Sts_IntlkTrip
=0
Cfg_ShedHold
Sheds
Device fault
Sts_NrdyDeviceFault
I/O fault
Sts_NrdyIOFault
Configuration error
Sts_NrdyCfgErr
Out of service
Sts_NrdyOoS
OR
T
=1
Last good CV value
=0
Cfg_CVIntlk
Interlock CV target
Sts_Hand
Hand
Inp_HandFdbk
Device (CV) feedback (EU)
Inf/NaN check
Sts_CVInfNaN
Val_CVSet
Output target
Rate of change limits
Cfg_CVRoCIncrLim
Cfg_CVRoCDecrLim
V
Cfg_SkipRoCLim
Cfg_SkipRoCLim = 1:
Skip RoC Limiting in OoS/Hand/Intlk/Maint/Ovrd
Cfg_SkipRoCLim = 0:
Skip RoC Limiting in OoS/Hand
Val_CVOut
Output (EU)
Cfg_CVEUMax
Cfg_CVEUMin
Cfg_CVRawMax
Cfg_CVRawMin
f(x)
Scaling
EU→Raw
H/
Sts_Active
Cfg_MaxInactiveCV
PCmd_Virtual
MCmd_Virtual
MCmd_Physical
PCmd_Physical
S
R
Sts_Virtual
Virtual
0
Out_CVData
Output (raw)

The following diagram explains logic for Interlock trip status:



The following diagram illustrates the functionality of the PAO instruction for a pulsed device:



Virtualization

When Virtualization is active, the output of the analog output holds at zero and I/O faults are ignored. Manipulate the instruction to operate as if a

working analog output is present. Do this for instruction testing and operator training. Set the Inp_Virtual operand to 1 to enable virtualization. After finishing virtualization, set the Inp_Virtual operand to 0 to return to normal operation.

Configuration of Strings for HMI

Configure strings for HMI faceplates, as seen in FactoryTalk View, and for the Logix Designer configuration dialog box. The strings are set to extended properties of tag items. Configure the strings in the Logix Designer application only.

- Description
- Label for graphic symbol
- Tag name
- Display Library for HMI Faceplate call-up
- Instruction name
- Area name
- URL link
- CV raw units – EngineeringUnit information from Out_CVData tag
- CV engineering units – EngineeringUnit information from Val_CVOut tag
- Path to an object with more information – Navigation information from Cfg_HasMoreObj member of P_ANALOG_INPUT structure, Cfg_HasMoreObj.@Navigation
- Path to an object with output CV information – Navigation information from Val_CVOut member of P_ANALOG_INPUT structure, Val_CVOut.@Navigation
- Path to an object with output data information - Navigation information from Inp_PosFdbk member of P_ANALOG_INPUT structure, Inp_PosFdbk.@Navigation

Command Source

The instruction uses the following command sources. The command sources are prioritized in order from highest to lowest in this table.

Command Source	Description
Hand	Logic outside the instruction owns control of the device. The instruction tracks the state of the device for bumpless transfer back to one of the other command sources. (Highest Priority command source)
Out-of-Service	The instruction is disabled and has no owner.
Maintenance	Maintenance owns control of the device and supersedes Operator, Program, and Override control. Operator commands and settings from the HMI are accepted. Bypassable interlocks and permissives are bypassed, and device timeout checks are not processed.

Command Source	Description
Override	Priority logic owns control of the device and supersedes Operator and Program control. Override Input (Inp_OvrdCmd) is accepted. If so configured, bypassable interlocks and permissives are bypassed.
Program locked	Program logic owns control of the device. Program commands (PCmd_) are accepted. Operator cannot take control from the Program. Override cannot take control from the Program unless Cfg_OvrdOverLock = 1.
Program	Program logic owns control of the device. Program commands (PCmd_) are accepted.
Operator locked	The Operator owns control of the device. Operator commands (OCmd_) from the HMI are accepted. Program cannot take control from Operator. Override cannot take control from Operator unless Cfg_OvrdOverLock = 1.
Operator	The Operator owns control of the device. Operator commands (OCmd_) from the HMI are accepted. (Lowest Priority command source)

The instruction is able to enable/disable the following operations:

- Maintenance Out-of-Service exists
- Maintenance exists
- External exists
- Program (unlocked) exists
- Program locked exists
- Operator (unlocked) exists
- Operator locked exists
- PCmd_Lock used as a Level (1 = Lock, 0 = Unlock)

The instruction checks for invalid configurations on control forcing the nearest valid configuration.

The core control model deals specifically with the arbitration of the source of the commands and parameters currently being accepted by the receiving function. More specifically, whether the source is:

- A programmatic entity, one which resides entirely within the processing environment, or
- An external interface entity, one which issues commands and parameters external and asynchronously to the processing environment.

These sources are known as Prog (Program) and Oper (Operator) control, respectively.

The optional ability to lock into one control source or the other is required to ensure that the other control source cannot acquire privilege when the designer wants to prevent it.

Core Command Source Model

The core control model consists of the following control sources: Oper, OperLocked, Prog, ProgLocked. The control model defaults to this configuration. Other control sources may be present in the model but act as overriding control sources acting independent of the base Operator/Program state machine.

Enabling control sources as Configuration

The individual control sources may be enabled or disabled by the user. The default configuration utilizes the entire base model. The differentiation being that upon power-up of the processing environment the control source will be the designated default. Some combinations of enabled control sources are disallowed as they are either unnecessary or could create unintended changes.

Prog Power Up

Configuration allows the user to specify whether Operator or Program will be the power-up default.

Prog Priority

Configuration allows the user to specify whether Operator or Program commands will win when simultaneously asserted.

Automatic reset of commands

All commands are treated as 'one-shot-latched.' This means that all commands are automatically cleared when the instruction executes and processes them.

Change Destination States

Under certain configurations the destination command source for some commands may change. This is in keeping with the intent of the command. Example: If the Prog is disabled, the destination of the OCmd_Prog command is directed to the ProgLocked state instead of the Prog state. This is done to maintain the intent of the OCmd_Prog command: the operator entity wishes to place the function in control of the program. If the command was eliminated then there would be no way to accomplish this. This is only done in configurations where it would cause no conflict or race condition. But serves to preserve as much user functionality as is practical.

Higher Priority Command Sources

Higher priority command sources which operate independently within the model: Override, Maintenance, Out-of-Service, In-Service and Hand.

Monitoring the PAO Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See *Index Through Arrays* for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Rung-condition-out is cleared to false. Sts_eSrc is set to 0. Sts_bSrc is set to 1 (never scanned).
Instruction first run	All commands that are automatically cleared each execution are cleared and ignored. The Program/Operator selection is set based on the configuration (Cfg_ProgPwrUp). The Program or Operator lock selection is set to unlocked. The Maintenance acquired/released state is not modified and persists through a controller powerup or PROG-to-RUN transition. PSet_Owner and Sts_Owner are set to 0.
Rung-condition-in is false	The instruction is put Out of Service if Inp_Hand=0. The output is de-energized. All alarms are cleared. Command source selection processing proceeds as normal except that all ownership status bits (Sts_Maint, Sts_Ovrd, Sts_Prog and Sts_Oper) are cleared to 0. Commands are still received for Maintenance, Operator, and Program and are processed behind the scenes, just as they are in Hand mode.
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Instruction first run	See Instruction first run in the Ladder Diagram table.
Instruction first scan	See Instruction first run in the Ladder Diagram table.
EnableIn is false	See Rung-condition-in is false in the Ladder Diagram table.
EnableIn is true	See Rung-condition-in is true in the Ladder Diagram table.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

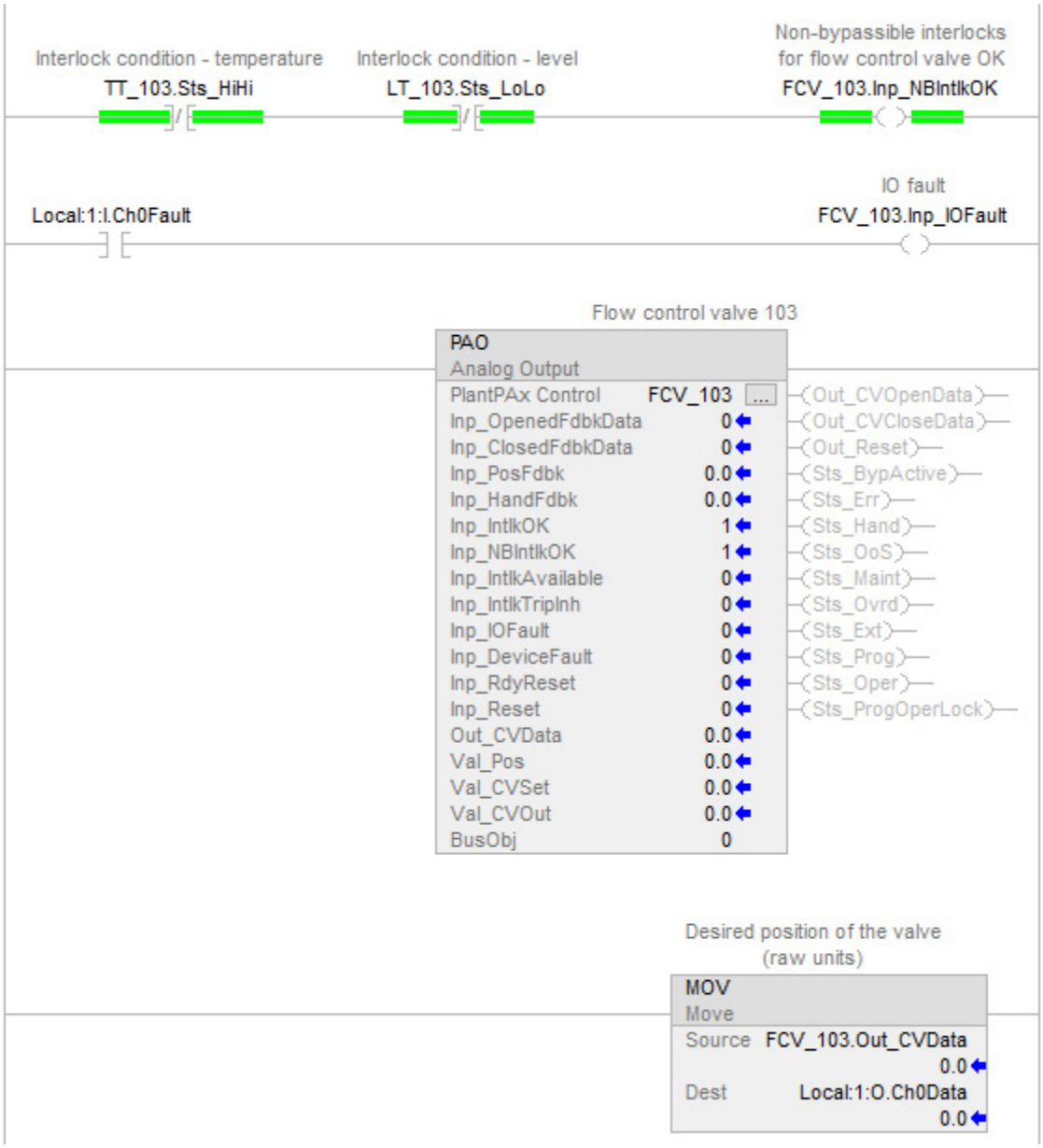
In Structured Text, EnableIn is always true during normal scan. Therefore, if the instruction is in the control path activated by the logic it will execute.

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Instruction first run	See Instruction first run in the Ladder Diagram table.
EnableIn is true	See Rung-condition-in is true in the Ladder Diagram table.
Postscan	EnableIn and EnableOut bits are cleared to false.

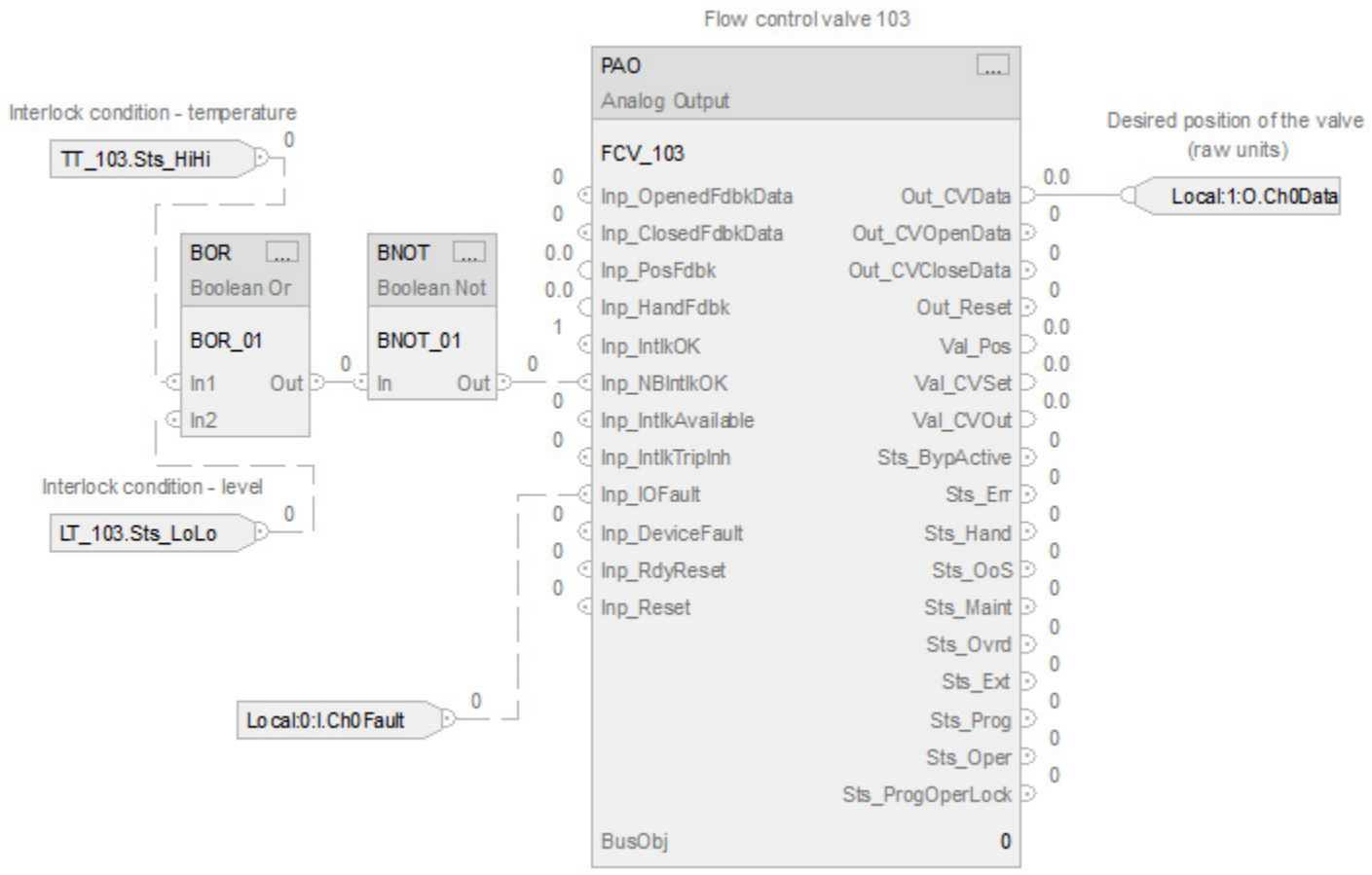
Example 1: Analog valve

The following example provides a demonstration of using a PAO instruction to control a valve. During normal operation, the operator sets the valve position using PAO through the HMI faceplate. This example also includes interlock conditions low-low level and high-high temperature provided by outputs of PAI instructions. The interlock conditions are used as inputs by PAO to set the valve (Local:0:O.Ch0Data = Out_CVData) to an interlock position (for example, closed). This is done by setting the PAO configuration parameter Cfg_Intlk to 0.

Ladder Diagram



Function Block Diagram



Structured Text

```
FCV_103.Inp_NBIntlkOK := NOT(TT_103.Sts_HiHi OR LT_103.Sts_LoLo);

FCV_103.Inp_IOFault := Local:0:I.Ch0Fault;

PAO(FCV_103, 0);

Local:1:O.Ch0Data := FCV_103.Out_CVData;
```

Example 2: Manual loading station

This example uses the PAO instruction to implement a manual loading station for a pressure control valve that is used to regulate gas supply to a process. The control valve in our example has opened and closed limit switches and a position feedback. The desired valve position is provided by the operator through the HMI faceplate.

The field inputs for position feedback, opened limit switch, and closed limit switch are connected to the instruction inputs Inp_PosFdbk,

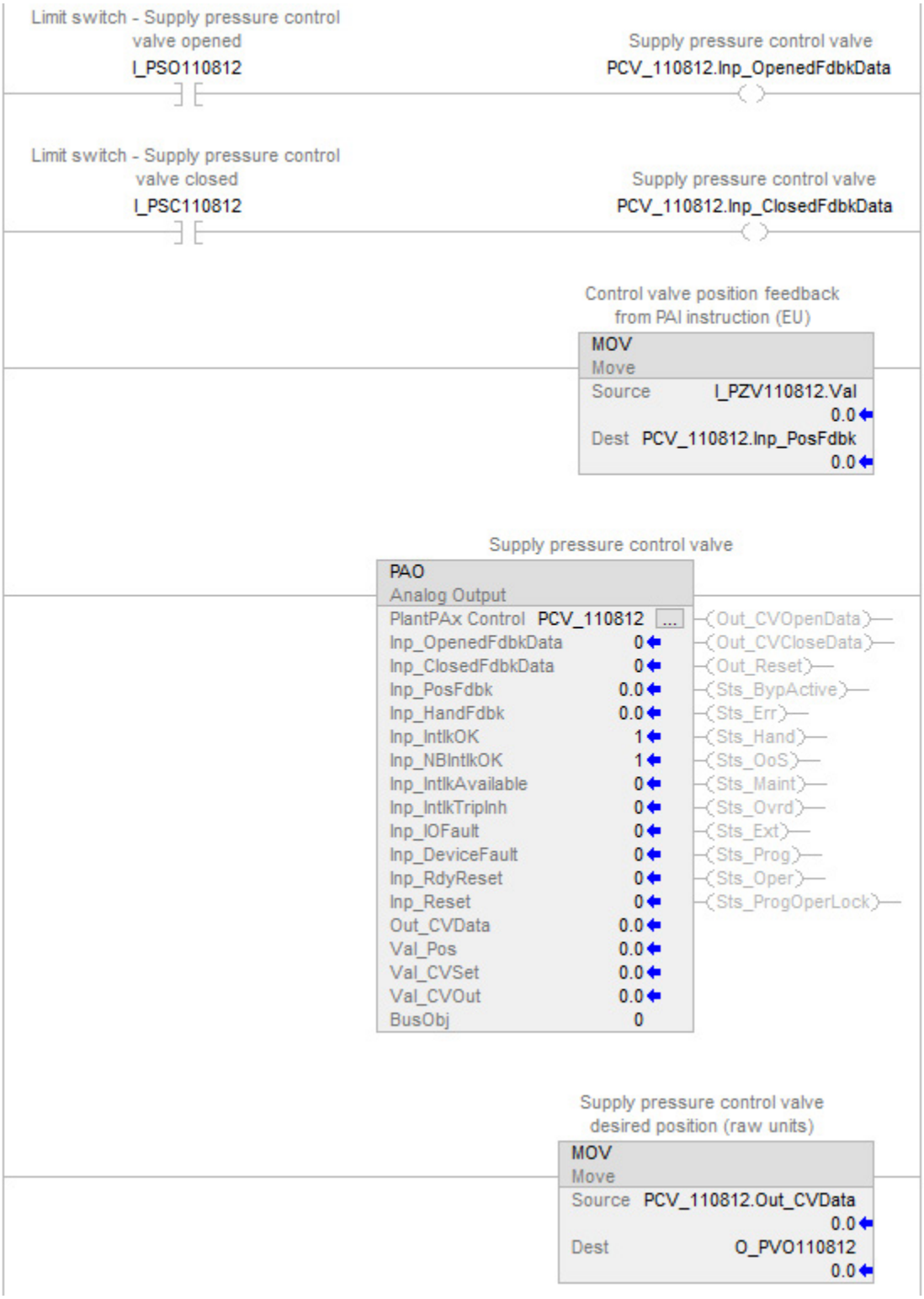
Inp_OpenedFdbkData, and Inp_ClosedFdbkData. The Out_CVData is connected to the field output going to the valve.

The parameters Cfg_HasOpenedFdbk and Cfg_HasClosedFdbk are both set to 1 so the instruction knows the field is providing opened and closed limit switches. The parameters Cfg_UseOpenedFdbk and Cfg_UseClosedFdbk are set to 1 so that these limit switches are used to determine device status.

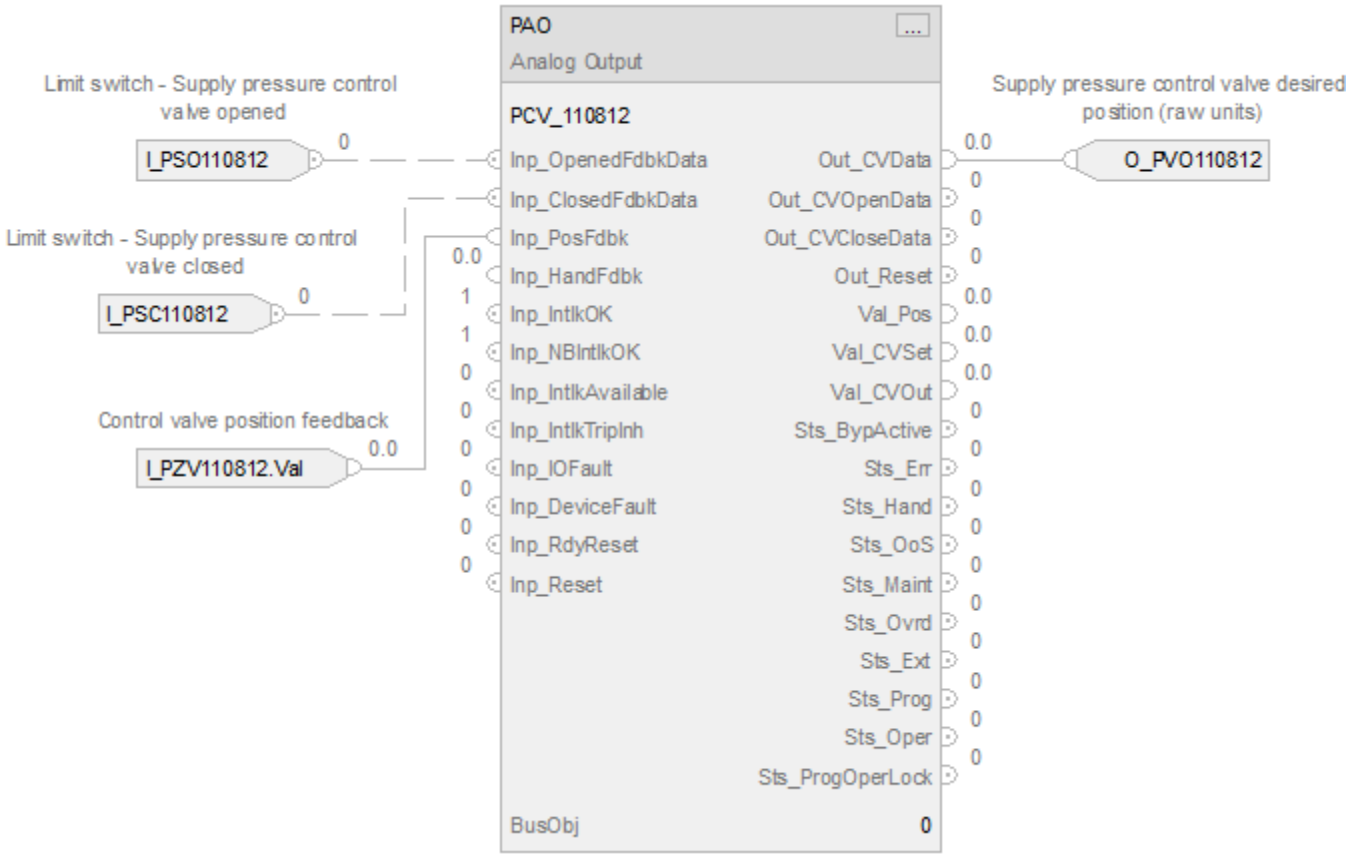
The analog output card is expecting an output in units of 4…20 mA; however, the faceplate shows the value in terms of 0…100% open. Therefore, the scaling parameters are set as follows. Cfg_CVEUMin=0, Cfg_CVEUMax=100, Cfg_CVRawMin=4, Cfg_CVRawMax=20.

The valve position feedback is provided in CV engineering units, percent in our particular case, via PAI instruction.

Ladder Diagram



Function Block Diagram



Structured Text

```
PCV_110812.Inp_OpenedFdbkData := I_PSO110812;

PCV_110812.Inp_ClosedFdbkData := I_PSC110812;

PCV_110812.Inp_PosFdbk := I_PZV110812.Val;

PAO(PCV_110812, 0);

O_PV110812 := PCV_103.Out_CVData;
```

Example 3: Ratchet valve

This example uses the PAO instruction to automate a ratcheting valve that is driven open or closed by using two discrete outputs to control flow. The flow valve in our example has a position feedback. The desired valve position is provided by an output of a control algorithm that is elsewhere in logic.

In this example, the field inputs for position feedback are wired (or connected) to the instruction input Inp_PosFdbk. Out_CVOpenData and Out_CVCloseData are connected to the field outputs going to the valve. The

input to the instruction to set valve position is wired to PSet_CV. Cfg_ProgNormal is set to 1 so the instruction defaults to Program mode.

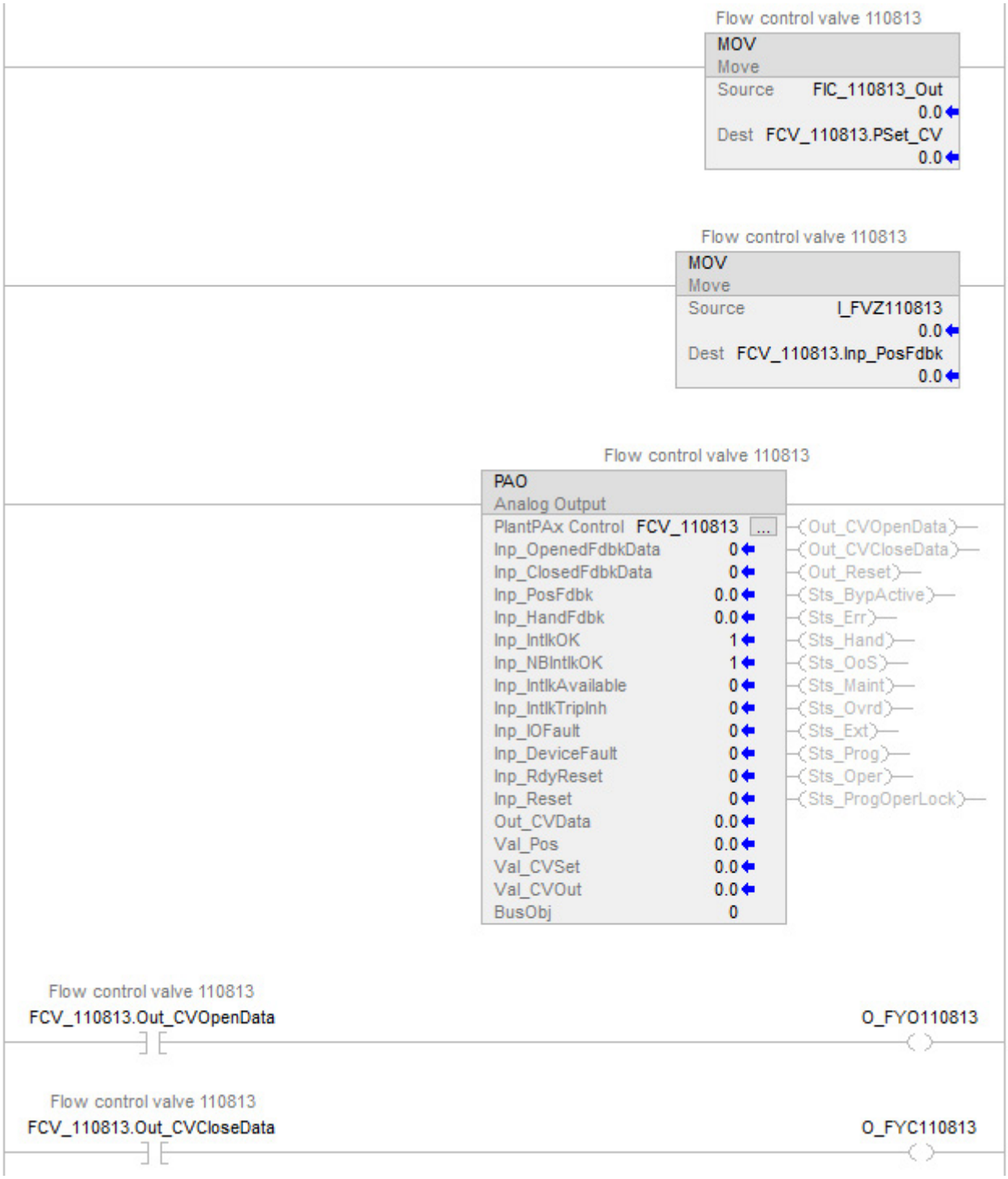
The analog output is not used; however, the faceplate shows the value of Val_CVOut in terms of 0...100% open, CV in engineering units. Therefore, the scaling parameters are set as follows. Cfg_CVEUMin=0, Cfg_CVEUMax=100, Cfg_CVRawMin=0 (default), Cfg_CVRawMax=100 (default)

The feedback signal is provided in CV engineering units via PAI instruction which scales the signal from feedback raw units to CV engineering units.

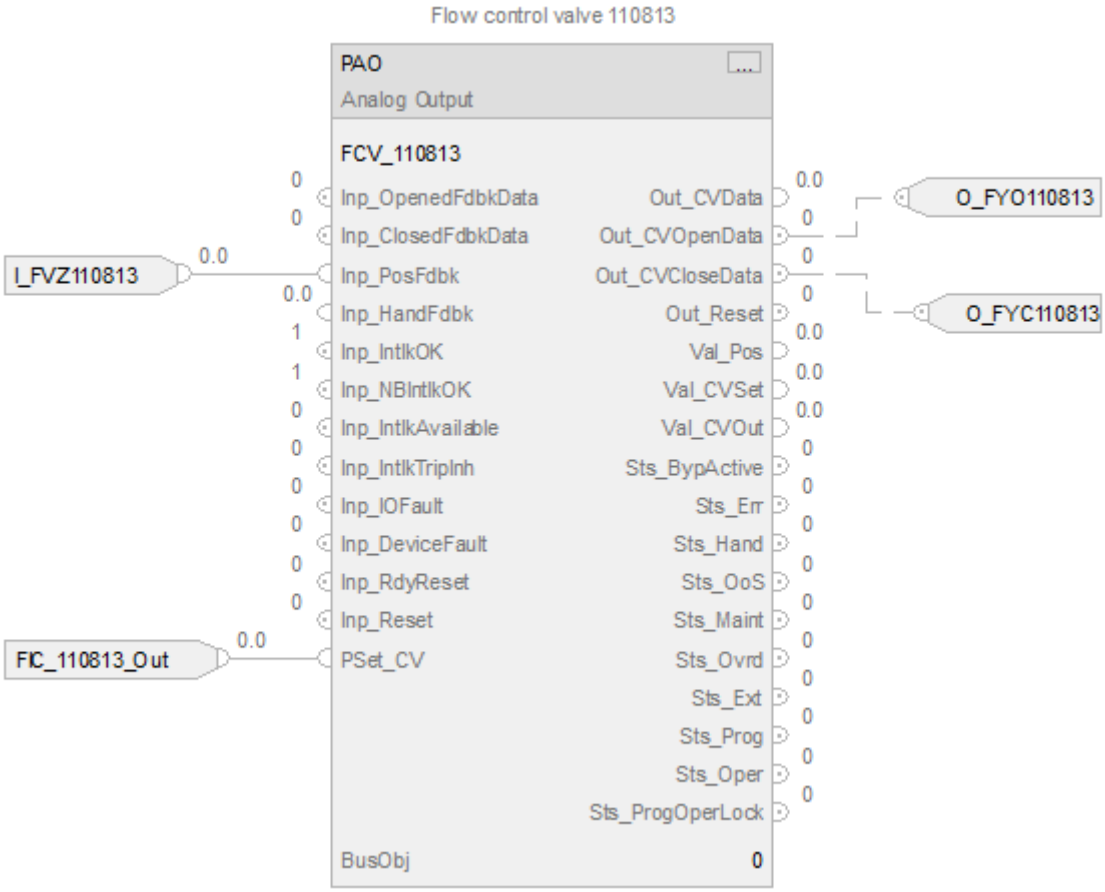
In this example, the ratcheting control valve is to be adjusted by cycling the open or close valve command for a period of time proportional to the amount the valve is to be moved. Cfg_CycleTime is set to 10, to define the overall period of the cycle to cycle on and off the open or close output. Cfg_OpenRate and Cfg_CloseRate are both set to 1, which means the required valve output is energized 1 second for every 1% difference between the target position and the actual position provided by feedback.

Cfg_MaxOnTime is set to 5 so that the output is energized for no more than 5 seconds of the 10-second cycle time to allow for the valve to move, and the feedback to be verified before the next cycle. Cfg_MinOnTime is set to 1 so that the output does not pulse if the calculated pulse time is less than 1 second.

Ladder Diagram



Function Block Diagram



Structured Text

```
FCV_110813.PSet_CV := FIC_110813_Out;
FCV_110813.Inp_PosFdbk := I_FVZ110813;
PAO(FCV_110813, 0);
O_FYO110813 := FCV_110813.Out_CVOpenData;
O_FYC110813 := FCV_110813.Out_CVCloseData;
```

See also

Process Analog Output feedback processing on page 377

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Process Analog Output feedback processing

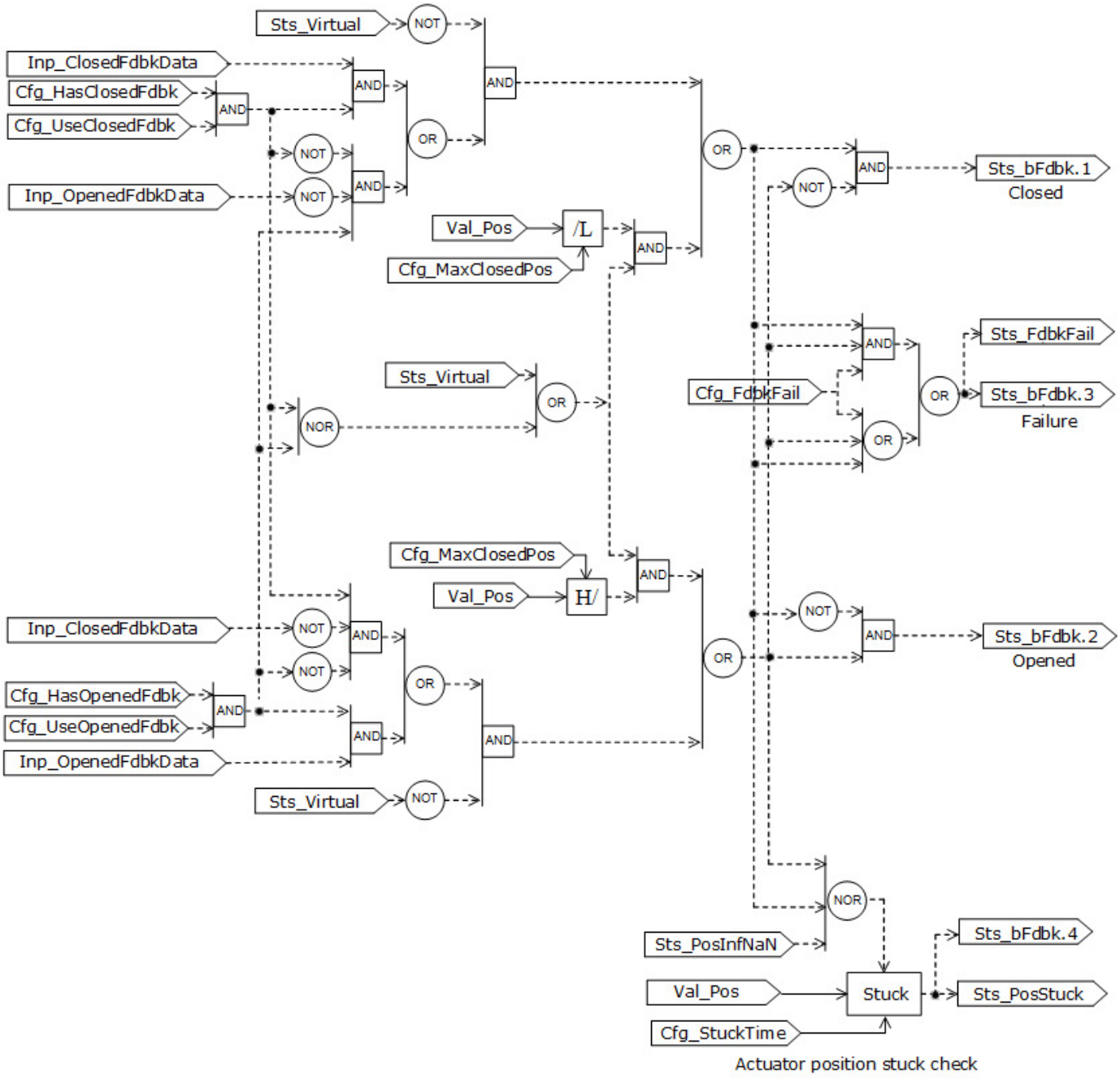
The PAO instruction evaluates feedback signals from limit switches provided by the device. The instruction reports device Open in one of these conditions:

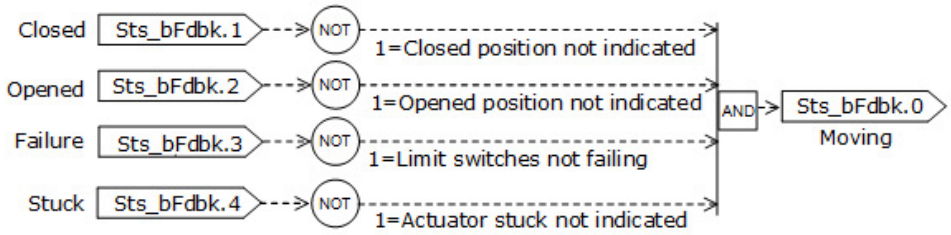
- Instruction is in virtual and Open limit switch is used and activated.
- Instruction is not in virtual, Open limit switch is not used, Closed limit is used but not activated.
- Instruction is in virtual, and Val_Pos > Cfg_MaxClosedPos.
- No limit switch in use and Val_Pos > Cfg_MaxClosedPos.

The instruction reports device Closed in one of these conditions:

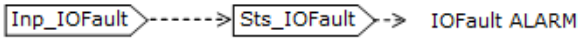
- Instruction is not in virtual, Closed limit switch is used and activated.
- Instruction is not in virtual, Open limit switch used and not activated, Closed limit switch not used.
- Instruction is in virtual and Val_Pos ≤ Cfg_MaxClosedPos.
- No limit switch in use and Val_Pos ≤ Cfg_MaxClosedPos.

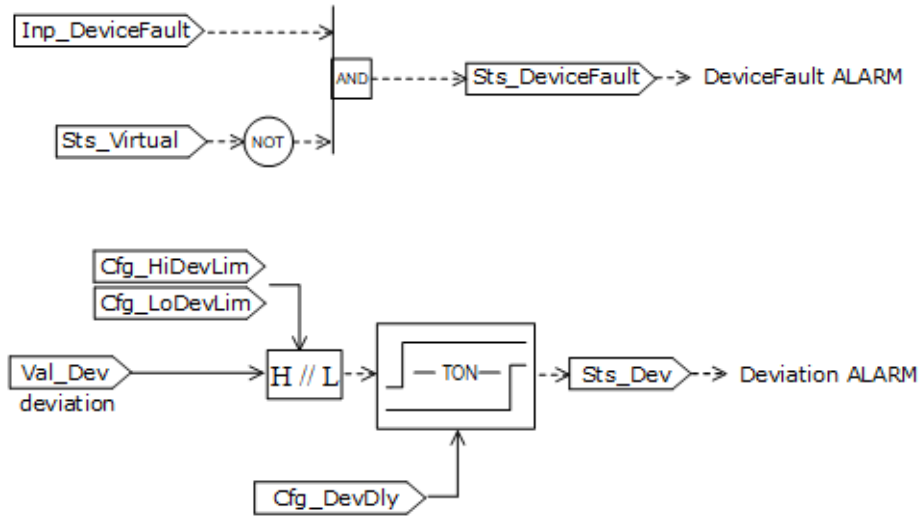
Instruction logic used in processing of feedback from limit switches and device/actuator confirmed position is shown in this diagram.





These diagrams show logic involved in IO fault, device fault, and deviation out of range alarm conditions:





Pulse Output

The PAO instruction can be used in connection with a pulse-driven device, typically a valve, if the instruction is configured for pulse outputs (Cfg_HasPulseOut=1). The instruction generates pulses with duration modulated by the position error, difference (Val_Dev) between desired position calculated by the instruction (Val_CVOut), and device confirmed actual position (Val_Pos). Actual position is either provided by the device via feedback (Inp_PosFdbk) or simulated by the instruction if the physical feedback is not available or used.

Pulses are generated in cycles, one pulse per cycle. Pulses cannot be too short and too long to minimize wear of the equipment, device, or actuator. These time constraints are user defined as Cfg_MinOnTime and Cfg_MaxOnTime. Cycle duration Cfg_CycleTime is also user defined.

The instruction uses three parameters associated with assumed behavior of the device or actuator, rate at which the device moves when opening Cfg_OpenRate, rate at which the device moves when closing Cfg_CloseRate, and delay Cfg_DeadTime in device or actuator reaction when the device or actuator is commanded to open or close. If the device or actuator responds to the Open/Close pulse with delay, e.g. due to stiction, compensate for the delay by entering Cfg_DeadTime. Uncompensated delay results in steady state position error.

IMPORTANT Failure to set correct values for Cfg_OpenRate, Cfg_CloseRate, and Cfg_DeadTime can result in undesired moves of the device or actuator.

The pulsing logic provides functionality of a position control loop. Pulse width is calculated for a cycle.

For Val_Dev /DIAGRAM_PATH_SPECIFIER>0 open pulse time is calculated as:

PulseDuration = min(Val_Dev/Cfg_OpenRate + Cfg_DeadTime, Cfg_MaxOnTime),

Val_OpenTime = PulseDuration, for PulseDuration ≥ Cfg_MinOnTime,

Val_OpenTime = 0, for PulseDuration < Cfg_MinOnTime.

For Val_Dev <0 pulse close time is calculated as:

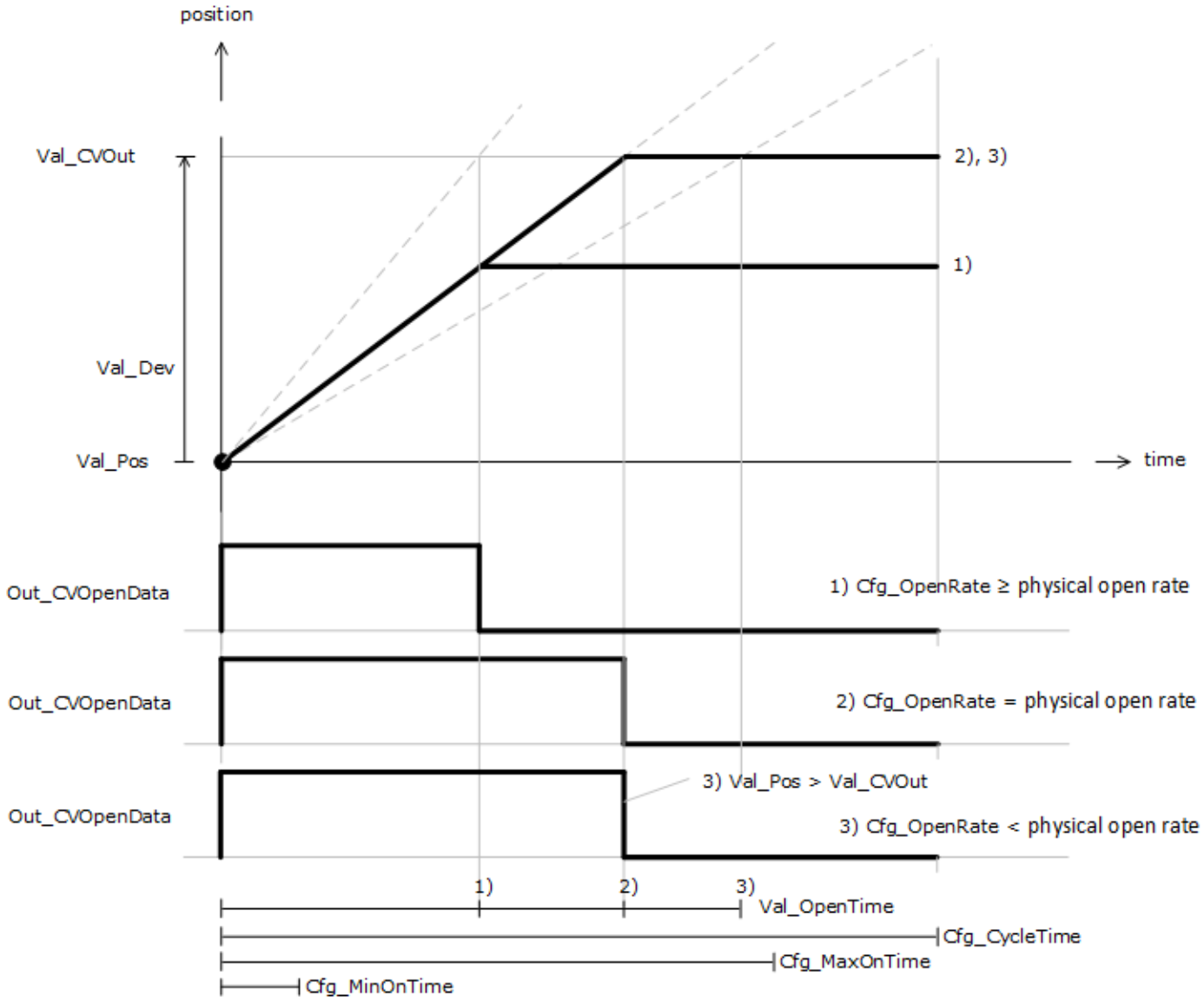
PulseDuration = min(-Val_Dev/Cfg_CloseRate + Cfg_DeadTime, Cfg_MaxOnTime),

Val_CloseTime = PulseDuration, for PulseDuration ≥ Cfg_MinOnTime,

Val_CloseTime = 0, for PulseDuration < Cfg_MinOnTime.

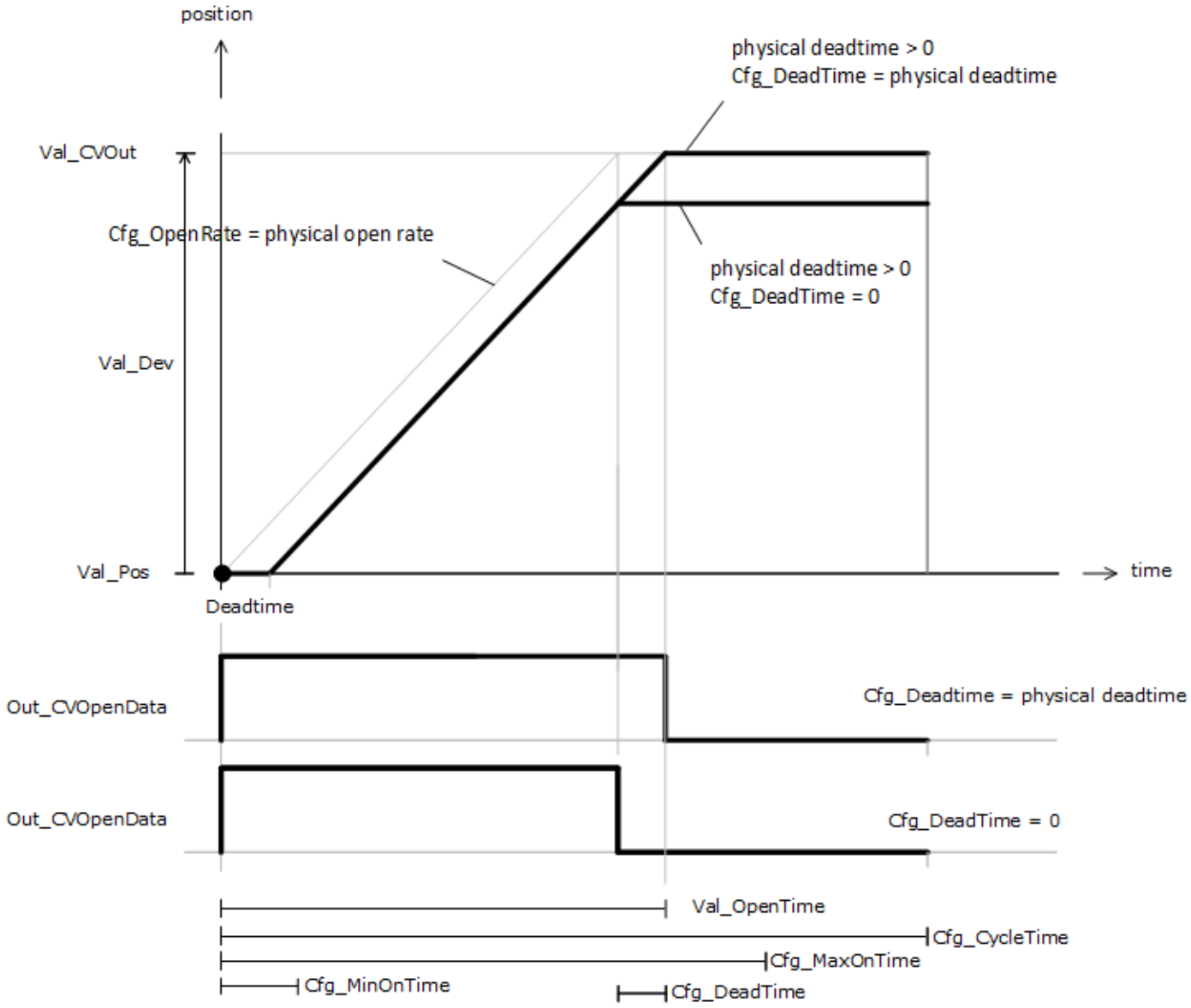
These diagrams provide examples of device/actuator response to step in desired position, Val_CVOut, for different settings of configuration parameters.

This diagram demonstrates OpenTime for a cycle resulting from actual setting of instruction parameters. Three cases are shown for different relations between entered Cfg_OpenRate compared to the real open rate of the device/actuator.

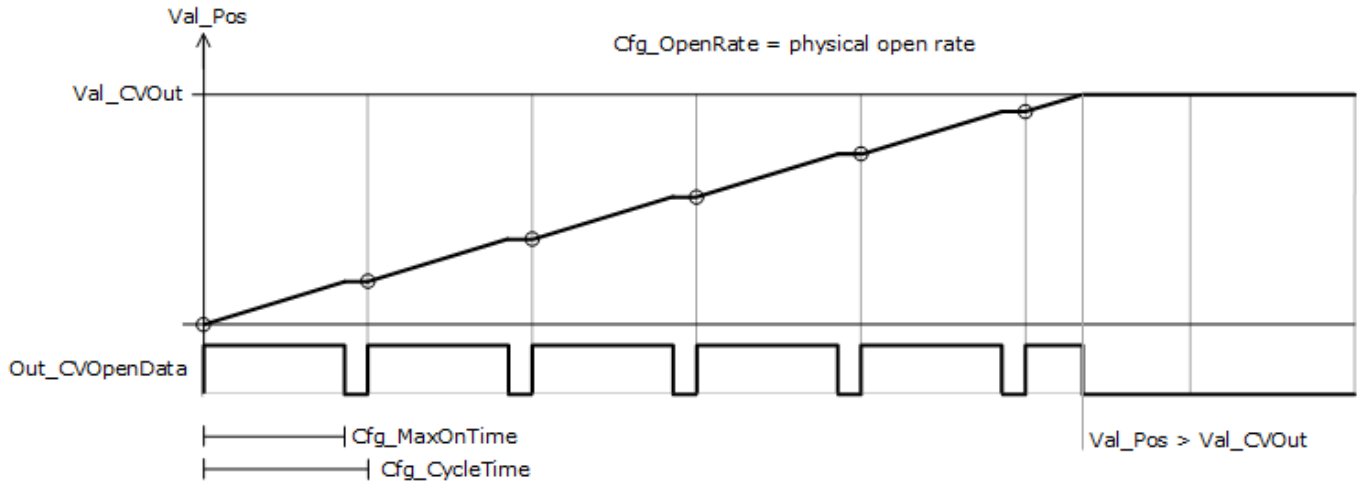


When the device starts to move or reverses direction, the instruction delays calculated Val_Pos by configured dead time (Cfg_DeadTime).

This figure demonstrates OpenTime for a cycle resulting from actual setting of instruction parameters and a device or actuator responding to the command with delay. The diagram shows two cases: delay ignored and delay compensated.

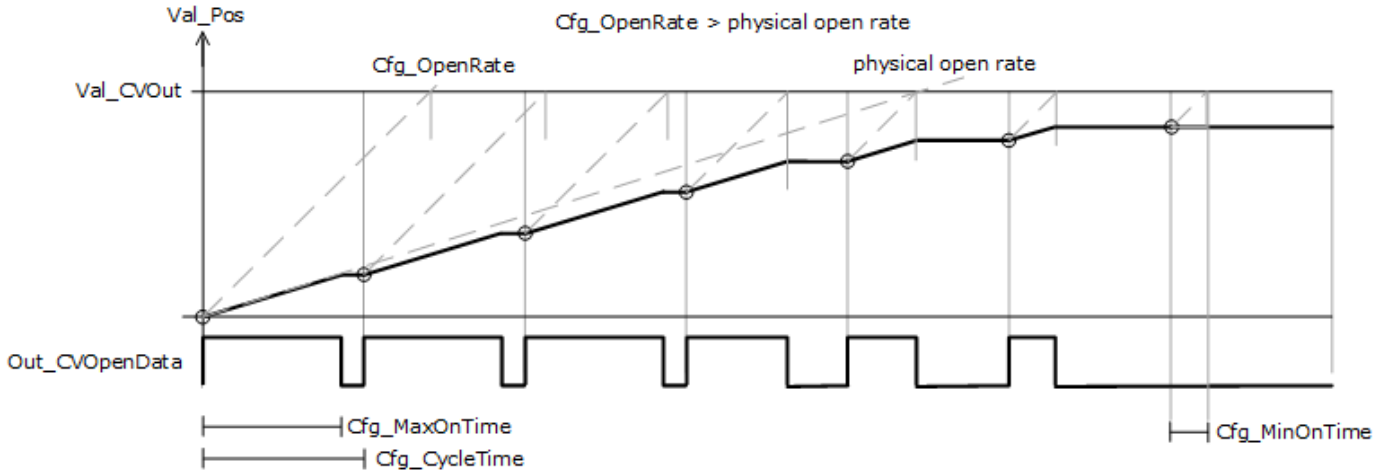


The following figures show a sequence of pulses for various instruction parameter settings and actuator behavior. In this diagram, the constant speed of the actuator is equal to Cfg_OpenRate or Cfg_CloseRate when in move.



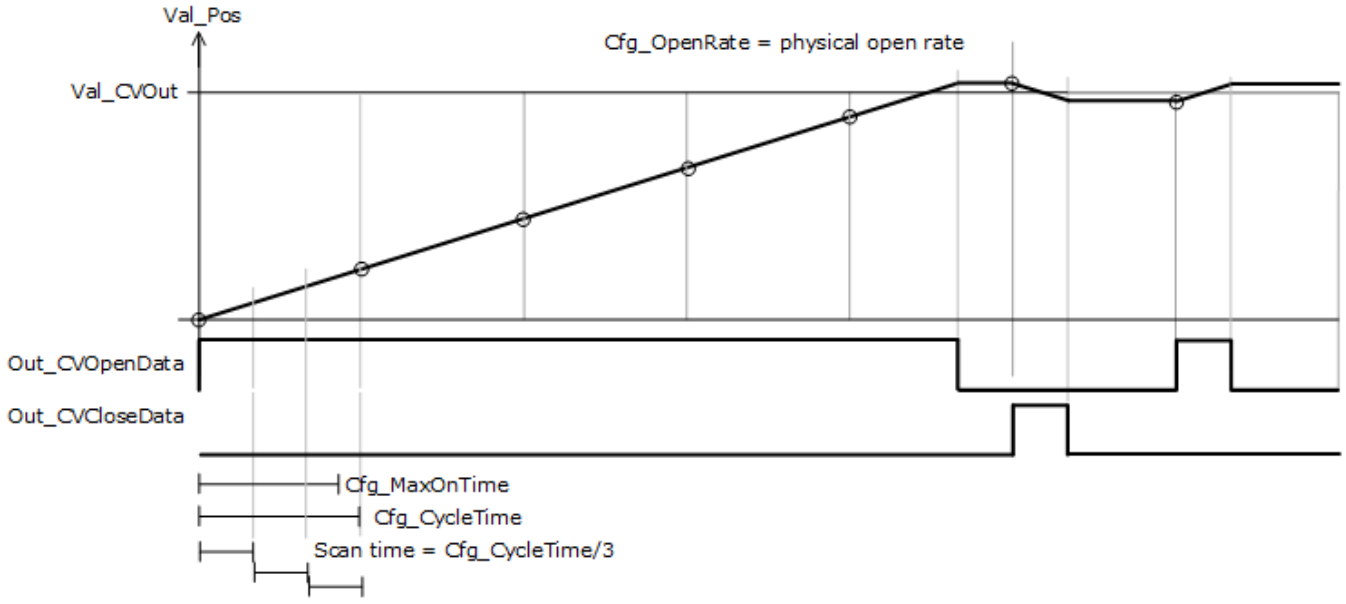
When configured rate Cfg_OpenRate is faster than the physical rate of the device, and position feedback is provided by the device, the position response differs. The Open pulse gets shorter when approaching the desired position until it violates the configured minimal On time (Cfg_MinOnTime), preventing the instruction from continuing pulsing.

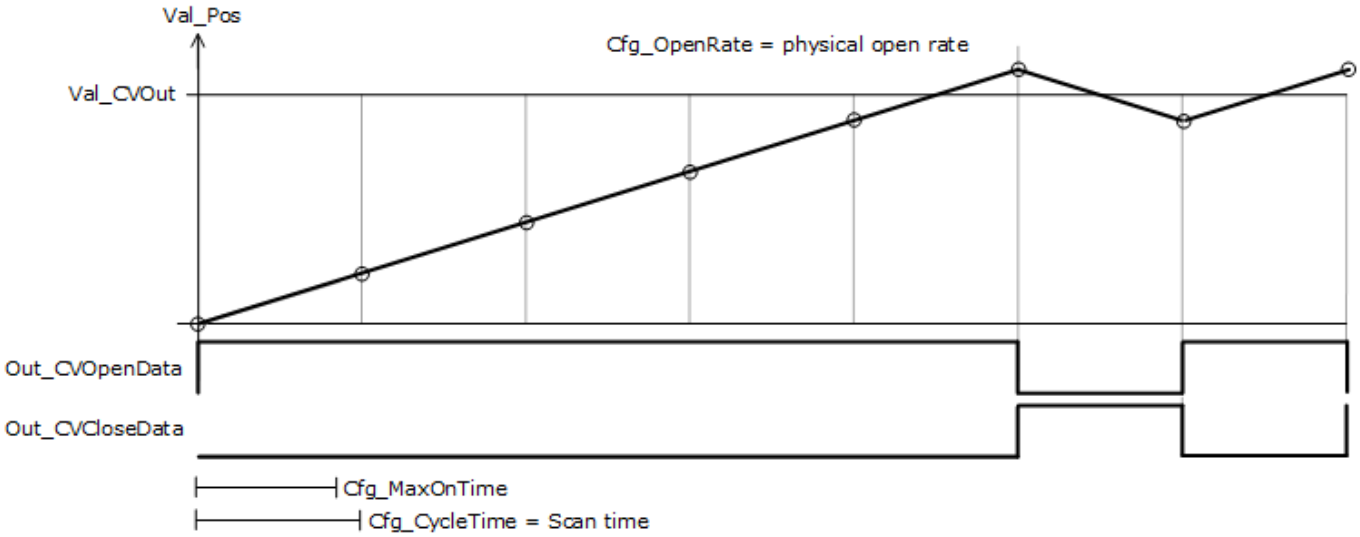
Do not use the pulsing function of the instruction when position feedback is not used, Val_Pos is calculated by the instruction, and configured rate Cfg_OpenRate differs from physical rate of the device.



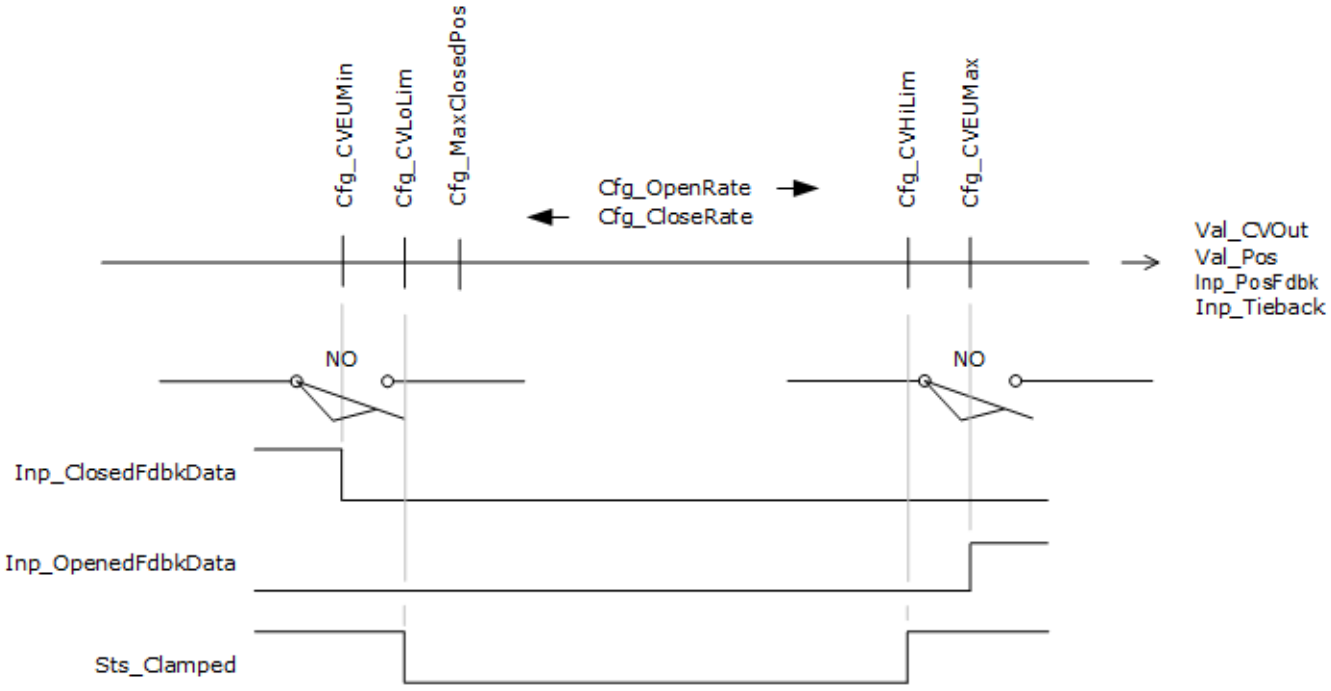
IMPORTANT Scan time of the instruction must be much shorter than Cfg_CycleTime. Failure to follow this recommendation can result in undesired movement of the device or actuator.

These figures show how larger scan time affects moves of the device or actuator driven by instruction pulse outputs.





The relationship between limit settings associated with device or actuator position is shown in the figure below for signals provided by normally open limit switches.



Position feedback simulation

When position feedback Inp_PosFdbk is not available, position Val_Pos is calculated. In virtual, or if the position feedback is not available, the position feedback is calculated from the last scan position, actual scan time, and configured open (closed) rate:

Val_Pos = Val_Pos + ScanTime*Cfg_OpenRate when opening,

Val_Pos = Val_Pos - ScanTime*Cfg_CloseRate when closing,

Val_Pos = max(min(Val_Pos,Cfg_CVEUMax),Cfg_CVEUMin).

When not in virtual and when position feedback is not available, the instruction relies on limit switch availability to reset the calculated position when limits are reached. Val_Pos is set to Cfg_CVEUMax if Open limit switch is activated and Closed limit switch is not active or not used. Val_Pos is set to Cfg_CVEUMin if Closed limit switch is activated and Open limit switch is not active or not used.

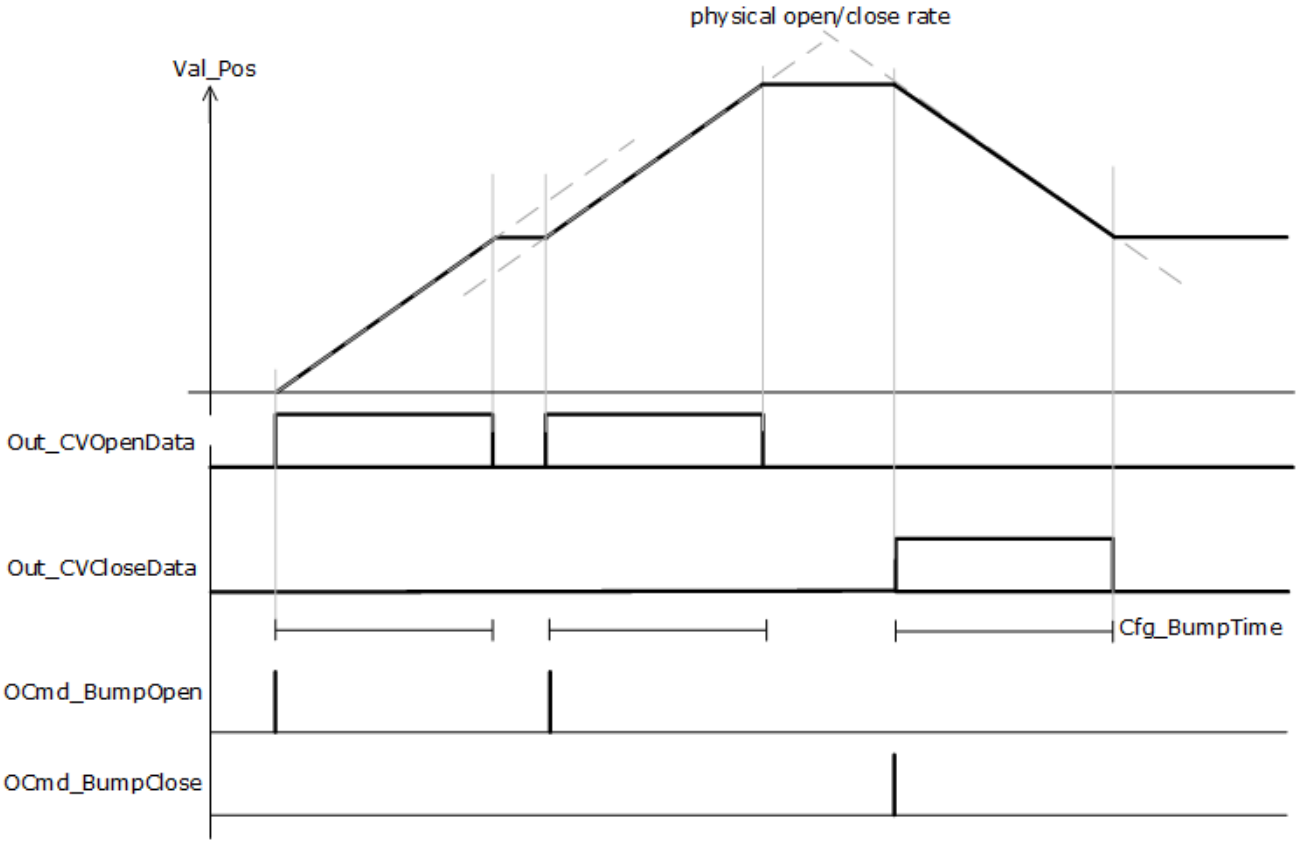
Bumping

The instruction can request the device to increment its position without position feedback. The user requests a bump to pulse the output. The pulse output is energized for configurable duration of time Cfg_BumpTime ≤ Cfg_CycleTime, as shown in the figure below.

The instruction is ready to execute bump open or bump close command when:

- Bump timer is greater than zero (zero disables the bump function), and
- Position feedback is infinite, not a number, or there is an I/O fault, or there is a device fault, and
- Device is in Operator or Maintenance mode, and
- Previous bump operation is not active, and

- Device is not at the end of travel, or limit switch is not being used, for the target direction of travel.



Operator command request confirmation

The PAO instruction allows an operator to use operator setting OSet_CV and command requests OCmd_BumpOpen, OCmd_BumpClose. Enforced security might require the request to be confirmed or canceled before the selected command executes. The instruction checks the security rules inspecting Cfg_CnfrmReqd. If Cfg_CnfrmReqd=0, no confirmation is required and the request executes immediately. If Cfg_CnfrmReqd=1 the instruction waits for confirmation OCmd_CmdCnfrm=1 and/or cancellation OCmd_CmdCncl=1. For Cfg_CnfrmReqd=2 or 3, eSignature is needed before the confirmation and cancellation is enabled.

See also

Process Analog Output

PlantPAx instructions on page 257

Process Boolean Logic (PBL)

This information applies to the ControlLogix 5380P and 5580P controllers.

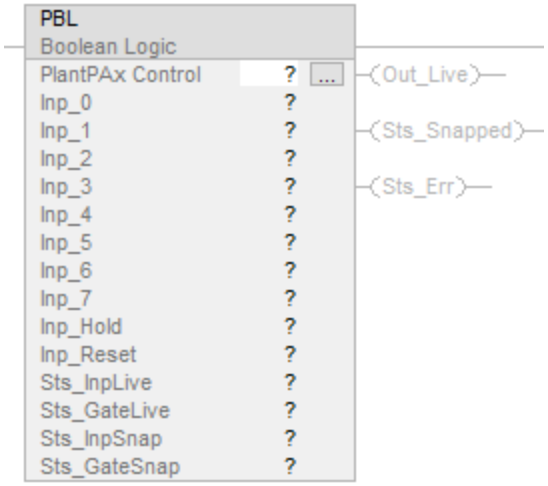
The Process Boolean Logic with Snapshot (PBL) instruction executes up to eight gates of configurable Boolean logic. Gate types available include AND, OR, XOR (Exclusive-OR), Set/Reset, Select, and Majority. Each gate provides up to four input conditions that are individually invertible using a configuration setting.

The PBL Instruction can record its current state:

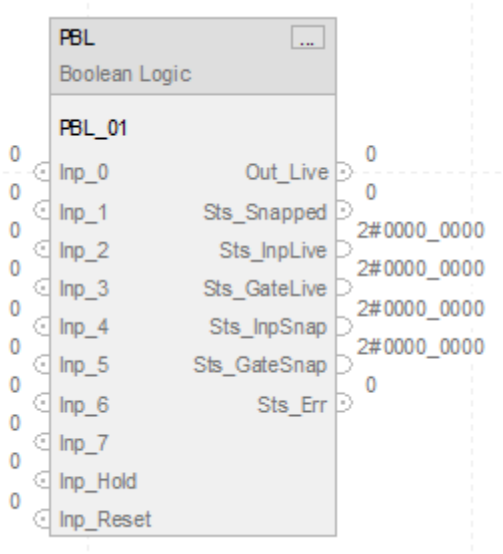
- After a change in output state.
- On Operator or Program command.
- Based on a logic loopback input.

Available Languages

Ladder Diagram



Function Block Diagram



Structured Text

PBL(PBL tag);

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PBL	P_BOOLEAN_LOGIC	tag	Data structure required for proper operation of instruction.

P_BOOLEAN_LOGIC Structure

Public members are standard, visible tag members that are programmatically accessible. Private, hidden members are used in HMI faceplates and are not programmatically accessible. Private members are in separate tables after public members.

Public Input Members	Data Type	Description
EnableIn	BOOL	Enable input. Ladder Diagram: Corresponds to the rung condition. Default is true.

Public Input Members	Data Type	Description
Inp_InitializeReq	BOOL	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request when reinitialization is needed. The instruction clears this operand automatically. Default is true.
Inp_0	BOOL	Logic input 0. Default is false.
Inp_1	BOOL	Logic input 1. Default is false.
Inp_2	BOOL	Logic input 2. Default is false.
Inp_3	BOOL	Logic input 3. Default is false.
Inp_4	BOOL	Logic input 4. Default is false.
Inp_5	BOOL	Logic input 5. Default is false.
Inp_6	BOOL	Logic input 6. Default is false.
Inp_7	BOOL	Logic input 7. Default is false.
Inp_Hold	BOOL	1 = Hold previous states in snapshot; 0 = Pass live states to snapshot. Default is false.
Inp_Reset	BOOL	1 = Reset snapshot latch, show live states. Default is false.
Cfg_UseInpHold	BOOL	1 = Use Inp_Hold to snap state; 0 = Use Cmds or Output transition to snap. Default is false.
Cfg_UsePCmd	BOOL	1 = Enable snapshot on PCmd_Snap 0 --> 1 (edge). Default is true.
Cfg_UseOCmd	BOOL	1 = Enable snapshot on OCmd_Snap 0 --> 1 (edge). Default is true.
Cfg_UseOut01	BOOL	1 = Enable snapshot on Output 0 --> 1 (rising edge). Default is true.
Cfg_UseOut10	BOOL	1 = Enable snapshot on Output 1 --> 0 (falling edge). Default is false.
Cfg_TimestampOnSnap	BOOL	1 = Generate a timestamp when snapshot occurs. Default is false.
Cfg_SnapOver	BOOL	1 = New snapshot overwrites without reset, 0 = Save first snapshot until reset. Default is false.
Cfg_HasMoreObj	BOOL	1 = Tells HMI an object with more information is available. Default is false.
Cfg_HasNav	SINT	Set bits indicate which navigation buttons are enabled. Default is 2#0000_0000.
Cfg_OnDly	REAL	Output ON delay time (seconds). Valid = 0.0 to 2147483.0 Default is 0.0.
Cfg_OffDly	REAL	Output OFF delay time (seconds). Valid = 0.0 to 2147483.0 Default is 0.0.
Cfg_CnfrmReqd	SINT	Operator Command Confirmation Required. Represents the type of command confirmation required. 0 = None, 1 = Command confirmation required, 2 = Performer e-signature required, 3 = Performer and approver e-signature required. Default is 0.

Public Input Members	Data Type	Description
PCmd_Snap	BOOL	Program command to capture Input, Gate states in snapshot. Default is false.
PCmd_Reset	BOOL	Program command to reset (re-arm) snapshot latch. Default is false.

Public Output Members	Data Type	Description
EnableOut	BOOL	Enable Output. This output state always reflects EnableIn input state.
Out_Live	BOOL	Condition logic output (result) after delay.
Out_Snap	BOOL	Condition logic output (result) at snapshot.
Val_DlyPctLive	DINT	Output OnDelay or OffDelay percent complete: live.
Val_DlyPctSnap	DINT	Output OnDelay or OffDelay percent complete: snapshot.
Val_SnapInit	DINT	Snapshot initiator: 1 = OCmd, 2 = PCmd, 3 = Out 0-->1, 4 = Out 1-->0, 5 = Inp_Hold.
Sts_Initialized	BOOL	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.
Sts_Snapped	BOOL	1 = Snapshot has been triggered, 0 = Snapshot showing live states.
Sts_RdyReset	BOOL	1 = A latched alarm or shed condition is ready to be reset.
Sts_InpLive	SINT	Live input status bits: .0 to .7 = Inp_0 to Inp_7.
Sts_GateLive	SINT	Live gate result status bits: .0 to .7 = Gate 0 to 7.
Sts_InpSnap	SINT	Snapshot of input status bits: .0 to .7 = Inp_0 to Inp_7.
Sts_GateSnap	SINT	Snapshot of gate result status bits: .0 to .7 = Gate 0 to 7.
Sts_GateSrc1Live	SINT	Live wire state for source 1 of each gate (bit# = gate#).
Sts_GateSrc2Live	SINT	Live wire state for source 2 of each gate (bit# = gate#).
Sts_GateSrc3Live	SINT	Live wire state for source 3 of each gate (bit# = gate#).
Sts_GateSrc4Live	SINT	Live wire state for source 4 of each gate (bit# = gate#).
Sts_GateSrc1Snap	SINT	Snapshot of wire state for source 1 of each gate (bit# = gate#).
Sts_GateSrc2Snap	SINT	Snapshot of wire state for source 2 of each gate (bit# = gate#).
Sts_GateSrc3Snap	SINT	Snapshot of wire state for source 3 of each gate (bit# = gate#).
Sts_GateSrc4Snap	SINT	Snapshot of wire state for source 4 of each gate (bit# = gate#).
Sts_OutInvertLive	BOOL	Output after inverter but before TON/TOF timers.
Sts_OutInvertSnap	BOOL	Snapshot of output after inverter but before TON/TOF.
Sts_Err	BOOL	1 = Error in configuration: See detail bits (Sts_Errxxx) for reason.
Sts_ErrGateFunc	SINT	1 = Error in each gate's function code (use 0 to 6).
Sts_ErrGateSrcPtr	SINT	1 = Error in each gate's source pointer (use 0 to 15).
Sts_ErrGateSrcMask	SINT	1 = Error in each gate's mask (source used) configuration (qty, choice).
Sts_ErrOutSrcPtr	BOOL	1 = Error in output's source pointer (use 0 to 15).
Sts_ErrTimer	BOOL	1 = Error in output's On Delay or Off Delay preset (use 0.0 to 2147483.0).

Private Input Members	Data Type	Description
Cfg_GateFunc	DINT[8]	Function code for gate M (1 = AND, 2 = OR, 3 = XOR, 4 = 2oo3, 5 = Set-Reset).
Cfg_GateSrc1Invert	SINT	Gate M Source #1 is Inverted (M by bit) (1 = invert). Default is 2#0000_0000.
Cfg_GateSrc1Mask	SINT	Gate M Source #1 is Used (M by bit) (1 = used). Default is 2#0000_0000.
Cfg_GateSrc1Ptr	DINT[8]	Pointer to Gate M Source #1 (0...7 = inputs, 8...15 = gate outputs).
Cfg_GateSrc2Invert	SINT	Gate M Source #2 is Inverted (M by bit) (1 = invert). Default is 2#0000_0000.
Cfg_GateSrc2Mask	SINT	Gate M Source #2 is Used (M by bit) (1 = used). Default is 2#0000_0000.
Cfg_GateSrc2Ptr	DINT[8]	Pointer to Gate M Source #2 (0...7 = inputs, 8...15 = gate outputs).

Private Input Members	Data Type	Description
Cfg_GateSrc3Invert	SINT	Gate M Source #3 is Inverted (M by bit) (1 = invert). Default is 2#0000_0000.
Cfg_GateSrc3Mask	SINT	Gate M Source #3 is Used (M by bit) (1 = used). Default is 2#0000_0000.
Cfg_GateSrc3Ptr	DINT[8]	Pointer to Gate M Source #3 (0...7 = inputs, 8...15 = gate outputs).
Cfg_GateSrc4Invert	SINT	Gate M Source #4 is Inverted (M by bit) (1 = invert). Default is 2#0000_0000.
Cfg_GateSrc4Mask	SINT	Gate M Source #4 is Used (M by bit) (1 = used). Default is 2#0000_0000.
Cfg_GateSrc4Ptr	DINT[8]	Pointer to Gate M Source #4 (0...7 = inputs, 8...15 = gate outputs).
Cfg_HasNav	SINT	Set bits indicate which navigation buttons are enabled. Default is 2#0000_0000.
Cfg_OutSrcInvert	BOOL	Out source (before minimum duration timer) is inverted (1 = invert). Default is false.
Cfg_OutSrcPtr	DINT	Source bit for Output (0...7 = inputs, 8...15 = gates). Default is 0.
HMI_Const	SINT[9]	Constants (for use in HMI indirection of parameters).
OCmd_Reset	BOOL	Operator command to reset (re-arm) snapshot latch. Default is false.
OCmd_Snap	BOOL	Operator Command to capture input, gate states in snapshot. Default is false.

Private Output Members	Data Type	Description
ORdy_Reset	BOOL	1 = Ready for OCmd_Reset (enables button).
ORdy_Snap	BOOL	1 =Ready for OCmd_Snap (enables button).
Val_LastGate	DINT[8]	Last gate pin which uses this gate result (0 = not used, 1...32 = gates, 33 = output).
Val_LastInp	DINT[8]	Last gate pin which uses this input (0 = not used, 1...32 = gates, 33 = output) For animation.
Val_SnapTimestamp	DINT[7]	Snapshot timestamp [0] = year, [1] = month, [2] = day, [3] = hour, [4] = minute, [5] = second, [6] = usecond.

Operation

The PBL instruction:

- Provides up to eight Boolean inputs and eight logic gates.
- Each gate has four inputs. Each input can be enabled or disabled and can be normal or inverted. Each enabled gate input can be linked to a source, which is an instruction input or the result of a preceding gate
- Use one of these methods to configure the eight gates:
  - Logical AND: The gate's output is true if all of the enabled gate inputs, after configured inversions, are true. An AND gate can have up to four inputs enabled.
  - Logical OR: The gate's output is true if any of the enabled gate inputs, after configured inversions, are true. An OR gate can have up to four inputs enabled.

- Logical XOR (Exclusive OR): The gate's output is true if an odd number of the enabled gate inputs, after configured inversions, are true. An XOR gate can have up to four inputs enabled.
- Set-Reset: The gate's output is set true if one of its Set inputs is true, and is cleared to false if one of its Reset inputs is true. The gate's four inputs are:

  Input 1: SET (dominant)
  Input 2: RESET (dominant)
  Input 3: SET
  Input 4: RESET

- Select: If input 3 is false, the state of input 1 is passed to the gate output. If input 3 is true, the state of input 2 is passed to the gate output. A Select gate must have input 3 enabled and either or both of inputs 1 and 2 enabled.
- Majority (labeled 'MooN' for 'M out of N'): The gate's output is set true if a majority of its inputs, after configured inversions, are true. A majority would consist of 2 out of 2, 2 out of 3, or 3 out of 4. A Majority gate can have two, three, or four inputs enabled.

- Provides a snapshot capability that captures the state of the instruction for use later, until reset: all input states, gate states, and output state. The snapshot capability captures the state of the logic at the time that it tripped or shut down equipment, even if the logic states change after the shutdown. The snapshot is optionally timestamped from the controller clock with the year, month, day, hour, minute, second, or microsecond.
- Provides options to enable these following snapshot trigger conditions:
  - Capture snapshot on Operator Command (OCmd_Snap).
  - Capture snapshot on Program Command (PCmd_Snap).
  - Capture snapshot when the output transitions from 0 to 1.
  - Capture snapshot when the output transitions from 1 to 0.
  - Capture snapshot of previous scan's state when a loopback input becomes true. This capability captures the snapshot when the PBL output condition is the first-out condition in a downstream PINTLK block. The first-out indication from the PINTLK instruction can be looped back to the PBL instruction's Inp_Hold input to hold the last-scan state in the snapshot, including last scan's time stamp.

Implementation

Use the PBL instruction in these situations:

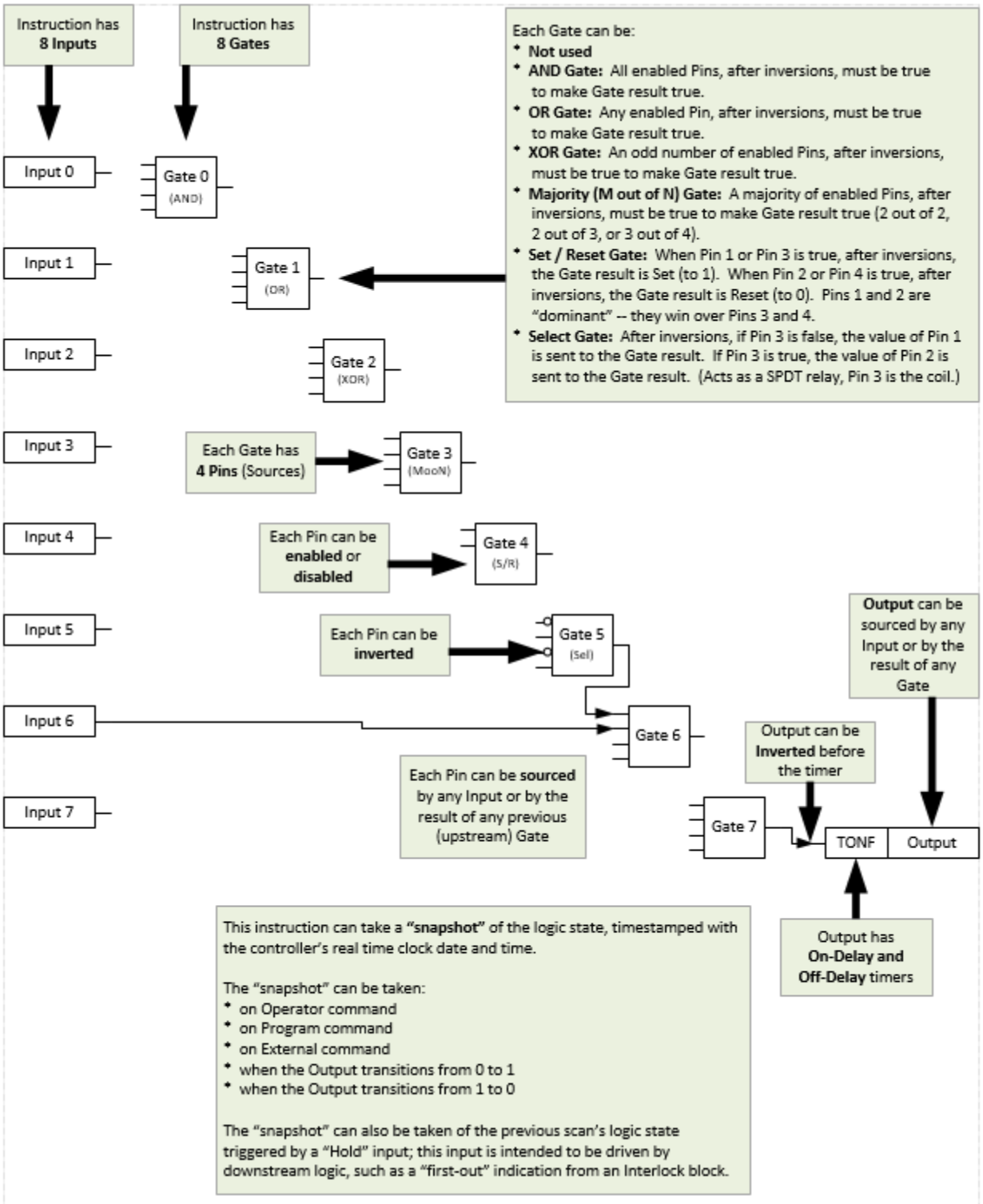
- A project requires an Interlock or Permissive condition that is more complicated than the simple OR-ing or AND-ing provided by the PINTLK (Interlocks) or PPERM (Permissives) Add-On Instructions.

- A project requires some Boolean (combination) logic that can be reconfigured from the HMI online, or which requires the snapshot capability for saving a copy of the logic state with a timestamp.
- A project contains more than the 16 interlock conditions or permissive conditions provided by the PINTLK and PPERM Add-On Instructions, but some of the conditions can be grouped together under one identification. For example, all of the bearing overtemperature signals for a pump and motor (Pump Inboard Bearing, Pump Outboard Bearing, Motor Inboard Bearing, and Motor Outboard Bearing) can be ORed together in a PBL instruction and the result presented to a PINTLK instruction as a single Bearing Overtemp condition.

Do not use this instruction in these situations:

- A project requires simple interlocks and permissives that can be handled by the PINTLK and PPERM instructions directly. These instructions can permit operation or trip operation.
- A project requires logic that is beyond the PBL Add-On Instruction capabilities or which is extremely time critical. The PBL instruction provides only eight inputs, eight gates, and one output with on-delay and off-delay timing, and it is implemented with table-driven code. Use hard-coded logic in native controller languages instead. The native programming languages are faster and provide functionality beyond what the PBL instruction can do.

This diagram illustrates the functionality of the PBL instruction:



Configuration

A maximum of eight gates can be configured using these tags:

Cfg_GateFunc[M] – This configuration is an array which defines the gate function. M = the gate number, 0-7.

- 1 = Logical AND Gate

- 2 = Logical OR Gate
- 3 = Logical XOR Gate
- 4 = Majority of Outputs are true
- 5 = Set/Reset
- 6 = A/B Selector

Each gate has four input pins. Each input pin can be enabled or disabled using this tag:

Cfg_GateSrc#Mask: This configuration is a SINT value that masks the gates (0-7) that are enabled for that input pin in a binary format. # = the input pin number (1-4). If the mask bit is high then the gate for that particular input pin is enabled.

Examples:

- Cfg_GateSrc1Mask.0 = 1 input pin 1 of gate 0 is enabled
- Cfg_GateSrc2Mask.1 = 1 input pin 2 of gate 1 is enabled
- Cfg_GateSrc3Mask.3 = 1 input pin 3 of gate 3 is enabled
- Cfg_GateSrc4Mask.0 = 1 input pin 4 of gate 0 is enabled

Each gate input pin can be normal or inverted.

Cfg_GateSrc#Invert: This configuration is a SINT value which inverts an input. # = the input pin number (1-4).

Examples:

- Cfg_GateSrc1Invert.0 = 1 input pin 1 of gate 0 is inverted.
- Cfg_GateSrc2Invert.2 = 1 input pin 2 of gate 2 is inverted
- Cfg_GateSrc3Invert.3 = 1 input pin 3 of gate 3 is inverted
- Cfg_GateSrc4Invert.6 = 1 input pin 4 of gate 6 is inverted

Each enabled gate input pin can be linked to a source, either an instruction input or the result of a preceding gate.

Cfg_GateSrc#Ptr[M] - This configuration is an array that defines the source for the input pin on each gate. Where # = the input pin number (1-4), and M = the gate number (0-7). A value of 0-7 represents an instruction input. A value of 8-15 represents the result of a preceding gate. Tip: A gate can only be used as an input into another gate with a higher gate value.

Examples:

- Cfg_GateSrc1Ptr[0] = 2 states that Inp_2 is configured in Gate 0 on pin 1.
- Cfg_GateSrc2Ptr[0] = 3 states that Inp_3 is configured in Gate 0 on pin 2.
- Cfg_GateSrc3Ptr[2] = 8 states that Gate 0 result is configured in Gate 2 on pin 3.
- Cfg_GateSrc4Ptr[7] = 12 states that Gate 4 result is configured in Gate 7 on pin 4.

Configure the output using these tags:

- Cfg_OutSrcPtr – This configuration determines which input/gate will be the output. A value of 0-7 represents an instruction input. A value of 8-15 represents the result of a gate.
- Cfg_OutSrcInvert – This configuration determines if the output will be inverted.

Configuration of Strings for HMI

Configure strings for HMI faceplates (FT View) and for the Logix Designer configuration dialog box. The strings are set to extended properties of tag items. Configure the strings in Logix Designer only.

- Description
- Label for graphic symbol
- Display Library for HMI Faceplate call-up
- Instruction name
- Area name
- URL link
- State name strings for 0-state and 1-state
- More Information

Monitor the PBL Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Resets the output on-delay and off-delay timers; clears the snapshot time stamp and data; clears any commands received while controller was in Program mode.
Instruction first run	Resets the output on-delay and off-delay timers; clears the snapshot time stamp and data; clears any commands received while controller was in Program mode.
Rung-condition-in is false	Clears output to false (off) and resets the output on-delay and off-delay timers.
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	Resets the output on-delay and off-delay timers; clears the snapshot time stamp and data; clears any commands received while controller was in Program mode.
Instruction first run	Resets the output on-delay and off-delay timers; clears the snapshot time stamp and data; clears any commands received while controller was in Program mode.
Instruction first scan	See instruction first run in the function block diagram table.
EnableIn is false	Clears output to false (off) and resets the output on-delay and off-delay timers.
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

In Structured Text, EnableIn is always true during normal scan. The instruction executes when it is in the control path activated by the logic.

Condition/State	Action Taken
Prescan	See Prescan in the Function Block Diagram table.
Instruction first run	See Instruction first run in the Function Block Diagram table.
EnableIn is true	See EnableIn is true in the Function Block Diagram table.
Postscan	See Postscan in the Function Block Diagram table.

Example

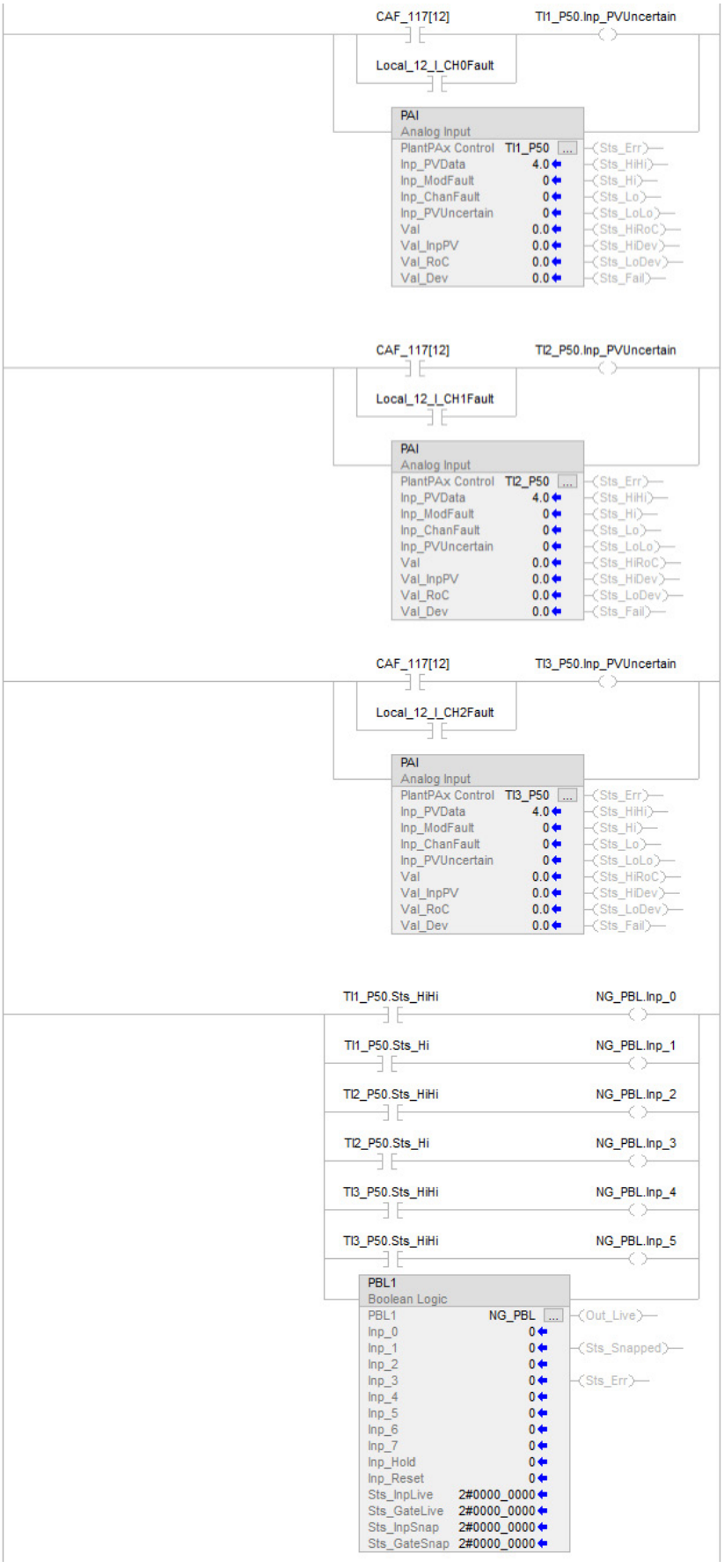
This example uses the PBL instruction to perform advanced interlocking logic that is based on the winding temperatures of a motor. This example navigates the parameter settings to fully illustrate the example.

In this example, there is a motor with three RTDs measuring temperature of the windings. To prevent damage to the windings, the motor must be interlocked if:

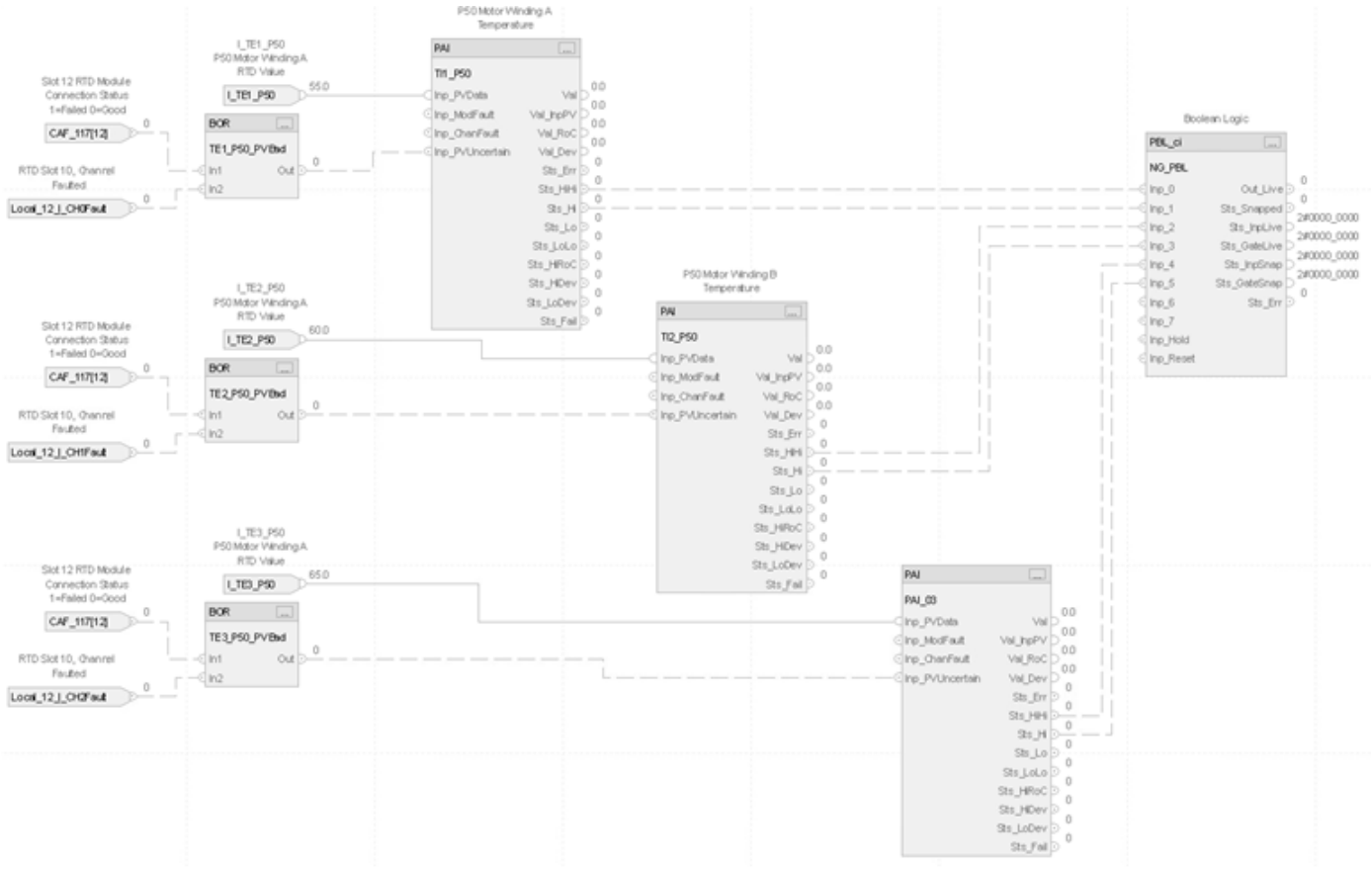
- Any of the windings are above the high-high temperature limit.
- The majority of the windings are above the high temperature limit.

PBL performs this function. The output of this logic feeds the interlock of the motor elsewhere in logic.

Ladder Diagram



Function Block Diagram



In this example, there is a motor with three RTDs measuring temperature of the windings. To prevent damage to the windings, the motor must be interlocked if any of the three windings are above the high-high temperature limit, or if the majority of the windings are above the high temperature limit. PBL is being used to perform this function. The output of this logic feeds the interlock of the motor elsewhere in logic.

The input parameters (Inp_0, Inp_1, Inp_2, Inp_3, Inp_4, Inp_5) are connected to the status outputs of the three winding temperature inputs. Three of the eight gates (0...7) in PBL are used in this example (1, 5, 6). Gate 1 is the OR of the three high-high status bits. Gate 5 checks if the majority of the high status bits are true. Gate 6 ORs the outputs of Gates 1 and 5 to set the output of PBL.

To set up the gate functions (Gates 1 and 6 as OR and Gate 5 as Majority), use these settings:

- Cfg_GateFunc[1] = 2
- Cfg_GateFunc[5] = 6
- Cfg_GateFunc[6] = 2

Gate 1 is set up to look at the three high-high status inputs (Inp_0, Inp_2, and Inp_4) by using these settings:

- Cfg_GateSrc1Mask.1 = 1, Cfg_GateSrc1Ptr[1] = 0
- Cfg_GateSrc2Mask.1 = 1, Cfg_GateSrc2Ptr[1] = 2
- Cfg_GateSrc3Mask.1 = 1, Cfg_GateSrc3Ptr[1] = 4

Gate 5 is set up to look at the three high status inputs (Inp_1, Inp_4, and Inp_5) by using these settings:

- Cfg_GateSrc1Mask.5 = 1, Cfg_GateSrc1Ptr[5] = 1
- Cfg_GateSrc2Mask.5 = 1, Cfg_GateSrc2Ptr[5] = 3
- Cfg_GateSrc3Mask.5 = 1, Cfg_GateSrc3Ptr[5] = 5

Lastly, Gate 6 is set up to look at the outputs of gates 1 and 5 by using these settings:

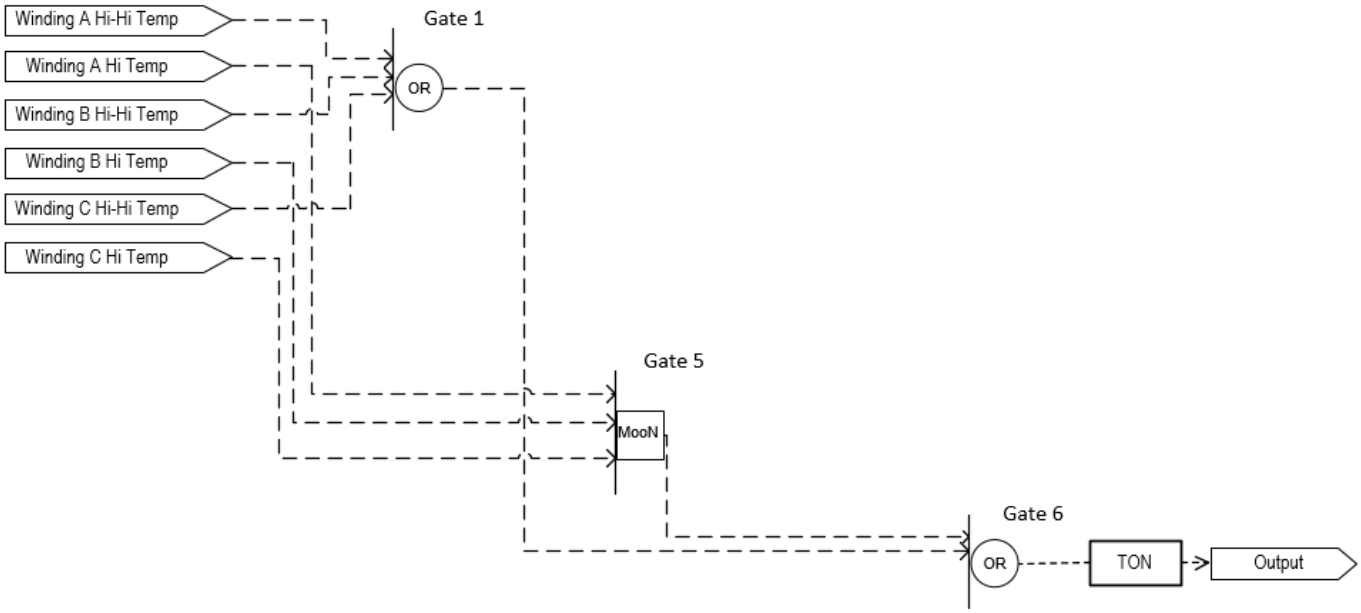
- Cfg_GateSrc1Mask.6 = 1, Cfg_GateSrc1Ptr[6] = 9
- Cfg_GateSrc2Mask.6 = 1, Cfg_GateSrc2Ptr[6] = 13

Cfg_OutSrcPtr needs to be set to 14 to take the output from Gate 6 and make it the output (Out_Live) of the PBL block. The on-delay time is then set to 5 seconds to prevent spurious trips of the output (Cfg_OnDly = 5).

Lastly, the descriptions provide documentation on the faceplate. In this example, these are the description settings:

- Out_Live.@State0 = OK
- Out_Live.@State1 = Tripped
- Inp_0.@Label = Winding A Hi-Hi Temp
- Inp_1.@Label = Winding A Hi Temp
- Inp_2.@Label = Winding B Hi-Hi Temp
- Inp_3.@Label = Winding B Hi Temp
- Inp_4.@Label = Winding C Hi-Hi Temp
- Inp_5.@Label = Winding C Hi Temp

This diagram illustrates the functionality of the example:



Structured Text

```
TI1_P50.Inp_PVData := I_TE1_P50;
TI1_P50.Inp_PVUncertain := (CAF_117[12] OR Local_12_I_CH0Fault);
PAI(TI1_P50);
TI2_P50.Inp_PVData := I_TE2_P50;
TI2_P50.Inp_PVUncertain := (CAF_117[12] OR Local_12_I_CH1Fault);
PAI(TI2_P50);
TI3_P50.Inp_PVData := I_TE3_P50;
TI3_P50.Inp_PVUncertain := (CAF_117[12] OR Local_12_I_CH2Fault);
PAI(TI3_P50);
TI1_P50.Sts_HiHi := NG_PBL.Inp_0;
TI1_P50.Sts_Hi := NG_PBL.Inp_1;
TI2_P50.Sts_HiHi := NG_PBL.Inp_2;
TI2_P50.Sts_Hi := NG_PBL.Inp_3;
TI3_P50.Sts_HiHi := NG_PBL.Inp_4;
TI3_P50.Sts_Hi := NG_PBL.Inp_5;
PBL(NG_PBL);
```

See also

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Process Command Source (PCMDSRC)

This information applies to the ControlLogix 5380P and 5580P controllers.

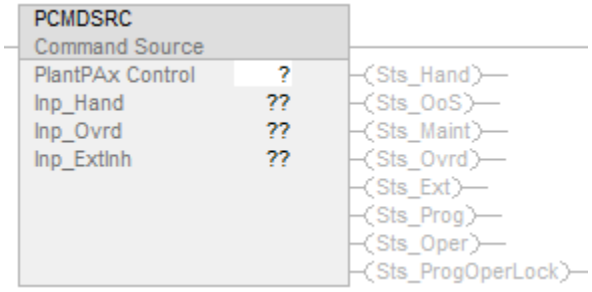
The Process Command Source (PCMDSRC) instruction selects the command source for a device.

The instruction includes these command sources:

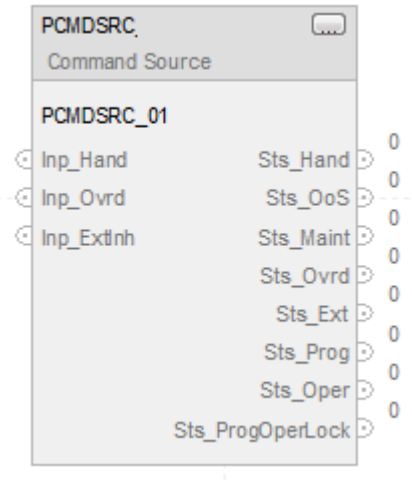
- Hand
- Out-of-Service
- Maintenance
- Override
- External
- Program locked
- Program
- Operator locked
- Operator

Available Languages

Ladder Diagram



Function Block Diagram



Structured Text

PCMDSRC (PCMDSRC tag);

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PlantPAx Control	P_COMMAND_SOURCE	tag	Data structure required for proper operation of the instruction.

P_COMMAND_SOURCE Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. Ladder Diagram: Corresponds to the rung condition. Default is true.

Input Members	Data Type	Description
Inp_OwnerCmd	DINT	Owner device command: 0 = None, Inp_OwnerCmd.10 = Operator Lock, Inp_OwnerCmd.11 = Operator Unlock, Inp_OwnerCmd.12 = Program Lock, Inp_OwnerCmd.13 = Program Unlock, Inp_OwnerCmd.14 = Acquire Maintenance, Inp_OwnerCmd.15 = Release Maintenance, Inp_OwnerCmd.16 = Acquire External, Inp_OwnerCmd.17 = Release External, Inp_OwnerCmd.29 = Echo. Default is 0.
Inp_InitializeReq	BOOL	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request when re-initialization is needed. The instruction clears this operand automatically. Default is true.
Inp_Hand	BOOL	1 = Acquire Hand (typically permanently set to local), 0 = Release Hand. Default is false.
Inp_Ovrd	BOOL	1 = Acquire Override (higher priority program logic), 0 = Release Override. Default is false.
Inp_ExtInh	BOOL	1 = Inhibit External acquisition, 0 = Allow External acquisition. Default is true.
Cfg_HasOper	BOOL	1 = Operator (unlocked) exists, can be selected. Default is true.
Cfg_HasOperLocked	BOOL	1 = Operator Locked exists, can be selected. Default is true.
Cfg_HasProg	BOOL	1 = Program (unlocked) exists, can be selected. Default is true.
Cfg_HasProgLocked	BOOL	1 = Program Locked exists, can be selected. Default is true.
Cfg_HasExt	BOOL	1 = External exists, can be selected. Default is false.
Cfg_HasMaint	BOOL	1 = Maintenance exists, can be selected. Default is true.
Cfg_HasMaintOoS	BOOL	1 = Maintenance Out of Service exists, can be selected. Default is true.
Cfg_OvrdOverLock	BOOL	1 = Override supersedes Program/Operator Lock, 0 = Do not override Lock. Default is true.
Cfg_ExtOverLock	BOOL	1 = External supersedes Program/Operator Lock, 0 = Do not override Lock. Default is false.
Cfg_ProgPwrUp	BOOL	1 = Power up to Program, 0 = Power up to Operator. Default is false.

Input Members	Data Type	Description
Cfg_ProgNormal	BOOL	Normal Source: 1 = Program if no requests, 0 = Operator if no requests. Default is false.
Cfg_PCmdPriority	BOOL	Command priority. 1 = Program commands win, 0 = Operator commands win. Default is false.
Cfg_PCmdProgAsLevel	BOOL	1 = PCmd_Prog used as a Level. Default is false.
Cfg_PCmdLockAsLevel	BOOL	1 = PCmd_Lock used as a Level (1=Lock, 0=Unlock). Default is false.
Cfg_ExtAcqAsLevel	BOOL	1 = XCmd_Acq used as Level (1 = Acquire, 0 = Release). Default is false.
PCmd_Oper	BOOL	Program command to select Operator (Program to Operator). The instruction clears this operand automatically. Default is false.
PCmd_Prog	BOOL	Program command to select Program (Operator to Program). The instruction clears this operand automatically. Default is false.
PCmd_Lock	BOOL	Program command to lock Program (disallow Operator). The instruction clears this parameter automatically if Cfg_PCmdLockAsLevel = 0. Default is false.
PCmd_Unlock	BOOL	Program command to unlock Program (allow Operator to acquire). The instruction clears this operand automatically. Default is false.
PCmd_Normal	BOOL	Program command to select Normal (Operator or Program). The instruction clears this operand automatically. Default is false.
MCmd_OoS	BOOL	Maintenance command to select Out of Service. The instruction clears this operand automatically. Default is false.
MCmd_IS	BOOL	Maintenance command to select In Service. The instruction clears this operand automatically. Default is false.
MCmd_Acq	BOOL	Maintenance command to acquire ownership (Operator/Program/External/Override to Maintenance). The instruction clears this operand automatically. Default is false.
MCmd_Rel	BOOL	Maintenance command to release ownership (Maintenance to Operator/Program/External/Override). The instruction clears this operand automatically. Default is false.
XCmd_Acq	BOOL	External command to acquire ownership (Operator/Program/Override/Maintenance to External). The instruction clears this operand automatically if Cfg_ExtAcqAsLevel = 0. Default is false.
XCmd_Rel	BOOL	External command to release ownership if Cfg_ExtAcqAsLevel = 0 (External to Operator/Program/Override/Maintenance). The instruction clears this operand automatically. Default is false.

Input Members	Data Type	Description
OCmd_Oper	BOOL	Operator command to select Operator (Program to Operator). The instruction clears this operand automatically. Default is false.
OCmd_Prog	BOOL	Operator command to select Program (Operator to Program). The instruction clears this operand automatically. Default is false.
OCmd_Lock	BOOL	Operator command to lock Operator (disallow Program). The instruction clears this operand automatically. Default is false.
OCmd_Unlock	BOOL	Operator command to unlock / release (allow Program to acquire) ownership. The instruction clears this operand automatically. Default is false.
OCmd_Normal	BOOL	Operator command to select Normal (Operator or Program). The instruction clears this operand automatically. Default is false.

Output Members	Data Type	Description
EnableOut	BOOL	Enable output. This output state always reflects EnableIn input state.
Sts_Initialized	BOOL	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.
Sts_eSrc	INT	The current command source is shown with status bits: Sts_eSrc.0: Lock, Sts_eSrc.1: Normal, Sts_eSrc.2: Hand, Sts_eSrc.3: Maintenance, Sts_eSrc.4: Override, Sts_eSrc.5: Program, Sts_eSrc.6: Operator, Sts_eSrc.7: Out of Service, Sts_eSrc.8: External.
Sts_bSrc	INT	Active selection bitmap (for HMI totem pole with command source request selection) Sts_bSrc.0: Hand, Sts_bSrc.1: Programmed Out of Service (rung false), Sts_bSrc.2: Maintenance Out of Service, Sts_bSrc.3: Maintenance, Sts_bSrc.4: Override, Sts_bSrc.5: External, Sts_bSrc.6: Program locked, Sts_bSrc.7: Program, Sts_bSrc.8: Operator locked, Sts_bSrc.9: Operator.
Sts_Hand	BOOL	1 = Hand is selected (supersedes OoS, Maintenance, Override, External, Program, Operator).
Sts_OoS	BOOL	1 = Out of Service is selected (supersedes Maintenance, Override, External, Program, Operator).
Sts_Maint	BOOL	1 = Maintenance is selected (supersedes Override, External, Program, Operator).
Sts_Ovrd	BOOL	1 = Override is selected (supersedes External, Program, Operator).
Sts_Ext	BOOL	1 = External is selected (supersedes Program, Operator).
Sts_Prog	BOOL	1 = Program is selected.
Sts_ProgLocked	BOOL	1 = Program is selected and Locked.
Sts_Oper	BOOL	1 = Operator is selected.
Sts_OperLocked	BOOL	1 = Operator is selected and Locked.
Sts_ProgOperSel	BOOL	Program/Operator selection (latch) state: 1 = Program, 0=Operator.

Output Members	Data Type	Description
Sts_ProgOperLock	BOOL	Program/Operator lock (latch) state, 1 = Locked, 0=Unlocked.
Sts_Normal	BOOL	1 = Selection equals the Normal (Program or Operator).
Sts_ExtReqInh	BOOL	1 = External request inhibited, cannot get to External from current state.
Sts_ProgReqInh	BOOL	1 = Program request inhibited, cannot get to Program from current state.
Sts_MAcqRcvd	BOOL	1 = Maintenance Acquire command received this scan.
MRdy_OoS	BOOL	1 = Ready for MCmd_OoS, enable HMI button.
MRdy_IS	BOOL	1 = Ready for MCmd_IS, enable HMI button.
MRdy_Acq	BOOL	1 = Ready for MCmd_Acq, enable HMI button.
MRdy_Rel	BOOL	1 = Ready for MCmd_Rel, enable HMI button.
XRdy_Acq	BOOL	1 = Ready for XCmd_Acq, enable HMI button.
XRdy_Rel	BOOL	1 = Ready for XCmd_Rel, enable HMI button.
ORdy_Oper	BOOL	1 = Ready for OCmd_Oper, enable HMI button.
ORdy_Prog	BOOL	1 = Ready for OCmd_Prog, enable HMI button.
ORdy_Lock	BOOL	1 = Ready for OCmd_Lock, enable HMI button.
ORdy_Unlock	BOOL	1 = Ready for OCmd_Unlock, enable HMI button.
ORdy_Normal	BOOL	1 = Ready for OCmd_Normal, enable HMI button.
Out_OwnerSts	DINT	Status of command source, owner command handshake and ready status: 0 = None, .10 = Operator Lock, .11 = Operator Unlock, .12 = Program Lock, .13 = Program Unlock, .14 = Acquire Maintenance, .15 = Release Maintenance, .16 = Acquire External, .17 = Release External, .18 = Has Maintenance, .19 = External Override Lock, .20 = Has External, .21 = Has Operator, .22 = Has Operator Locked, .23 = Has Program, .24 = Has Program Locked, .29 = Echo, .30 = Not Ready

Operation

The instruction uses these command sources. The command sources are prioritized in order from highest to lowest in this table.

Command Source	Description
Hand	Hardwired logic or other logic outside the instruction controls the device. The instruction tracks the state of the device for bumpless transfer back to one of the other command sources. (Highest priority command source)
Out-of-Service	The instruction is disabled and has no owner.
Maintenance	Maintenance controls the device and supersedes Operator, Program, External and Override control. Operator commands and settings from the HMI are accepted.
Override	Priority logic controls the device and supersedes Operator, Program and External control. Override Input (Inp_Ovrd) is accepted.

External	External logic (e.g. field pilot control or upstream SCADA) controls the device. External commands (XCmd_) are accepted.
Program locked	Program logic controls the device. Program commands (PCmd_) are accepted. Operator cannot take control from the Program. Override cannot take control from the Program unless Cfg_OvrdOverLock = 1.
Program	Program logic controls the device. Program commands (PCmd_) are accepted.
Operator locked	The Operator controls the device. Operator commands (OCmd_) from the HMI are accepted. Program cannot take control from Operator. Override cannot take control from Operator unless Cfg_OvrdOverLock = 1.
Operator	The Operator controls the device. Operator commands (OCmd_) from the HMI are accepted. (Lowest priority command source)

The instruction enables or disablesthese operations:

- Maintenance Out-of-Service exists
- Maintenance exists
- External exists
- Program (unlocked) exists
- Program locked exists
- Operator (unlocked) exists
- Operator locked exists
- PCmd_Lock used as a Level (1 = Lock, 0 = Unlock)

The instruction checks for invalid configurations of control and forces the nearest valid configuration.

The core control model arbitrates the source of the commands and parameters that the receiving function accepts. The core control model determines if the source is:

- A programmatic entity which resides entirely within the processing environment, or
- An external interface entity which issues commands and parameters external and asynchronously to the processing environment.

Locking a command source prevents the other command source from acquiring privilege.

Core command source model

The core control model consists of these command sources:

- Oper
- OperLocked
- Prog
- ProgLocked

The control model defaults to this configuration. Other command sources may be present in the model but act as overriding command sources, acting independent of the base Operator/Program state machine.

Enabling command sources as configuration

The individual command sources may be enabled or disabled by the user. The default configuration uses the entire base model; upon power-up of the processing environment the command source will be the designated default. Some combinations of enabled command sources are disallowed as they are either unnecessary or could create unintended changes.

Prog Power Up

Configuration allows the user to specify whether Operator or Program is the power-up default.

Prog Priority

Configuration allows the user to specify whether Operator or Program commands take priority when simultaneously asserted.

Automatic reset of commands

All commands are treated as one-shot-latched. This means that all commands are automatically cleared when the instruction executes and processes them.

Changing Destination States

Under certain configurations the destination command source for some commands may change. This is in keeping with the intent of the command. For example, if the Prog is disabled, the destination of the OCmd_Prog command is directed to the ProgLocked state instead of the Prog state. This maintains the intent of the OCmd_Prog command: the operator entity wishes to place the function in control of the program. If the command was eliminated there would be no way to accomplish this. This is only done in configurations where it would cause no conflict or race condition, but serves to preserve as much user functionality as is practical.

Higher Priority Command Sources

These Higher priority command sources operate independently within the model:

- External
- Override
- Maintenance

- Out-of-Service
- In-Service
- Hand

Initialization

The instruction is normally initialized in the instruction first run. Re-initialization can be requested any time by setting Inp_InitializeReq = 1. For proper initialization, when adding the instruction while performing an online edit of the code, make sure that Inp_InitializeReq = 1 (default value).

Monitor the PCMDSRC Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Rung-condition-out clears to false. Sts_eSrc is set to 0. Sts_bSrc is set to 0.
Instruction first run	All commands that are automatically cleared on each execution are cleared and ignored. The Program/Operator selection is set based on the configuration (Cfg_ProgPwrUp). The Program or Operator lock selection is set to unlocked. The Maintenance acquired/released state is not modified and persists through a controller powerup or PROG-to-RUN transition.

Condition/State	Action Taken
Rung-condition-in is false	The instruction is put Out of Service if Inp_Hand=0. The output is de-energized. Command source selection processing proceeds except that Program and Operators commands are ignored and cleared and all ownership status bits (Sts_Maint, Sts_Ovrd, Sts_Ext, Sts_Prog and Sts_Oper) are cleared to 0. When rung-condition-in becomes true, the Program/Operator selection is set based on the configuration (Cfg_ProgPwrUp).
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	EnableOut is cleared to false. Sts_eSrc is set to 0. Sts_bSrc is set to 0.
Instruction first run	All commands that are automatically cleared on each execution are cleared and ignored. The Program/Operator selection is set based on the configuration (Cfg_ProgPwrUp). The Program or Operator lock selection is set to unlocked. The Maintenance acquired/released state is not modified and persists through a controller powerup or PROG-to-RUN transition.
Instruction first scan	See Instruction first run in the Function Block Diagram table.
EnableIn is false	EnableOut is cleared to false. The instruction is put Out of Service if Inp_Hand=0. The output is de-energized. Command source selection processing proceeds except that Program and Operators commands are ignored and cleared and all ownership status bits (Sts_Maint, Sts_Ovrd, Sts_Ext, Sts_Prog and Sts_Oper) are cleared to 0. When rung-condition-in becomes true, the Program/Operator selection is set based on the configuration (Cfg_ProgPwrUp).
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

In Structured Text, EnableIn is always true during normal scan. Therefore, if the instruction is in the control path activated by the logic it will execute.

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Instruction first run	See Instruction first run in the Ladder Diagram table.
EnableIn is true	See Rung-condition-in is true in the Ladder Diagram table.
Postscan	EnableIn and EnableOut bits are cleared to false.

See also

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Data Conversions on page 1086

Process Command Source operating model

The core control model for the Process Command Source (PCMDSRC) instruction consists of these control sources:

- Oper
- OperLocked
- Prog
- ProgLocked

The control model defaults to this configuration. Other control sources may be present in the model but act as overriding control sources, acting independent of the base Operator/Program state machine.

Enabling control sources as Configuration

The individual control sources can be enabled or disabled by the user. The default configuration uses the entire base model; upon power-up of the processing environment the control source will be the designated default. Some combinations of enabled control sources are disallowed as they are either unnecessary or could create unintended changes.

Prog Power Up

Configuration allows the user to specify whether Operator or Program is the power-up default.

Prog Priority

Configuration allows the user to specify whether Operator or Program commands take priority when simultaneously asserted.

Automatic reset of commands

All commands are treated as one-shot latched. This means that all commands are automatically cleared when the instruction executes and processes them.

Changing Destination States

Under certain configurations the destination command source for some commands may change. This is in keeping with the intent of the command. Example: If the Prog is disabled, the destination of the OCmd_Prog command is directed to the ProgLocked state instead of the Prog state. This change of destination maintains the intent of the OCmd_Prog command: the operator entity wishes to place the function in control of the program. If the command was eliminated, there would be no way to accomplish this. This change of destination is only done in configurations where it would cause no conflict or race condition. It preserves as much user functionality as is practical.

Higher Priority Command Sources

These Higher priority command sources operate independently within the model:

- External
- Override
- Maintenance
- Out-of-Service
- In-Service
- Hand

See also

Process Command Source (PCMDSRC)

PlantPAx instructions on page 257

Process Deadband Controller (PDBC)

This information applies to the ControlLogix 5380P and 5580P controllers.

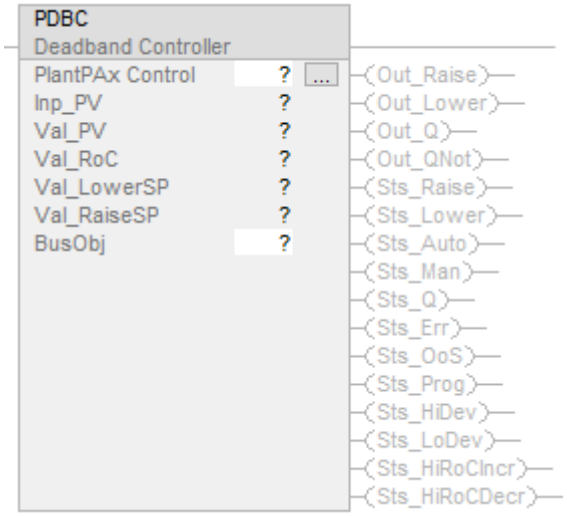
The Process Deadband Controller (PDBC) provides:

- A Raise output, which is activated when the PV is less than the entered Raise threshold, and a Lower output, which is activated when the PV is greater than the entered Lower threshold.
- Q and Q-Not outputs. Q is set when the PV falls below the Raise threshold and cleared when the PV rises above the Lower threshold; Q-Not is the inverse of Q.
- High and Low Deviation alarms with configurable thresholds and deadbands. These alarms can provide notification that the PV is approaching an out-of-control condition.
- Alarms for High PV Rate of Change Increasing and High PV Rate of Change Decreasing. These alarms can provide notification that the PV is changing faster than expected.

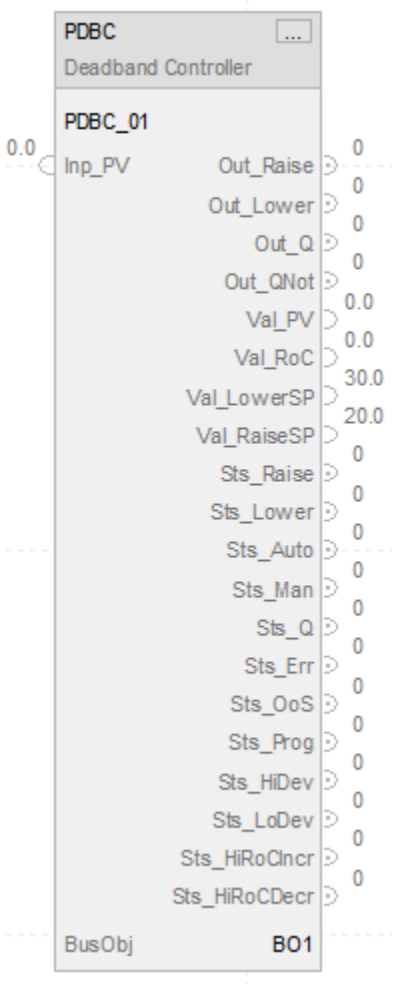
- Operation in Manual and Automatic Loop Modes. In Automatic Loop Mode, the outputs are triggered by the control algorithm to keep the PV within limits. In Manual Loop Mode, the operator directly manipulates the Raise and Lower outputs from the HMI.
- Operation in Operator, Program, Override, and Maintenance command sources.

Available Languages

Ladder Diagram



Function Block Diagram



Structured Text

PDBC(PDBCTag, o);

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PlantPAx Control	P_DEADBAND	tag	PDBC structure

BusObj	BUS_OBJ	tag	Bus component

P_ DEADBAND Structure

Public members are standard, visible tag members that are programmatically accessible. Private, hidden members are used in HMI faceplates and are not programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	Description
EnableIn	BOOL	Enable input. Ladder Diagram: Corresponds to the rung condition. Default is true.
Inp_InitializeReq	BOOL	Use this request when reinitializing. Default is true.
Inp_PV	REAL	Process variable being controlled (engineering units). Valid = Any float. Default is 0.0.
Inp_PVSrcQ	SINT	Input source and quality from channel object, if available (enumerator). Valid = 0 to 32. Default is 0.
Inp_PVNotify	SINT	Related process variable object alarm priority and acknowledgement status. 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged. Default is 0.
Inp_PVBad	BOOL	1 = Process variable or input/output communications status bad, 0 = Process variable and input/output communications healthy. Default is false.
Inp_OvrdCmd	SINT	Override command: 0 = No command, 1 = Raise, 2 = Lower, 3 = None, 4 = Manual, 5 = Auto. Default is 0.
Inp_OvrdRaiseSP	REAL	Override mode raise setpoint (engineering units). Valid = Any float. Default is 0.0.
Inp_OvrdLowerSP	REAL	Override mode lower setpoint (engineering units). Valid = Any float. Default is 0.0.
Inp_HiDevGate	BOOL	The gate input used for high deviation status detection. 1 = The corresponding analog input threshold monitoring is enabled. 0 = detection is disabled and the corresponding status output is forced off. Default is false.
Inp_LoDevGate	BOOL	The gate input used for low deviation status detection. 1 = The corresponding analog input threshold monitoring is enabled. 0 = detection is disabled and the corresponding status output is forced off. Default is false.
Inp_HiRoCIncrGate	BOOL	The gate input used for high rate of change (increasing) status detection. 1 = The corresponding analog input threshold monitoring is enabled. 0 = detection is disabled and the corresponding status output is forced off. Default is false.

Public Input Members	Data Type	Description
Inp_HiRoCDecrGate	BOOL	The gate input used for high rate of change (decreasing) status detection. 1 = The corresponding analog input threshold monitoring is enabled. 0 = detection is disabled and the corresponding status output is forced off. Default is false.
Inp_OwnerCmd	DINT	Owner device command: Inp_OwnerCmd.0 = None, Inp_OwnerCmd.10 = Operator lock, Inp_OwnerCmd.11 = Operator unlock, Inp_OwnerCmd.12 = Program lock, Inp_OwnerCmd.13 = Program unlock, Inp_OwnerCmd.14 = Acquire maintenance, Inp_OwnerCmd.15 = Release maintenance, Inp_OwnerCmd.16 = Acquire external, Inp_OwnerCmd.17 = Release external. Default is 0.
Inp_ExtInh	BOOL	1 = Inhibit external acquisition, 0 = Allow external acquisition. Default is false.
Inp_Hand	BOOL	1 = Acquire hand (typically hardwired local), 0 = Release hand. Default is false.
Inp_Ovrd	BOOL	1 = Acquire override (higher priority program logic), 0 = Release override. Default is false.
Cfg_PVDecPlcs	SINT	Number of decimal places for process variable display. Valid = 0 to 6. Default is 2.
Cfg_SetTrack	BOOL	1 = PSets track OSets in operator, OSets track PSets in program, 0 = no tracking. Default is false.
Cfg_HasMoreObj	BOOL	1 = Tells HMI an object with more info is available. Default is false.
Cfg_HasPVNav	BOOL	1 = Tells HMI to enable navigation to a connected process variable object. Default is false.
Cfg_HasOutNav	BOOL	1 = Tells HMI to enable navigation to a connected output object, 0 = No connected output object. Default is false.
Cfg_PVEUMin	REAL	Input process variable range minimum (engineering units). Valid = Any float. Default is 0.0.
Cfg_PVEUMax	REAL	Input process variable range maximum (engineering units). Valid = Any float. Default is 100.0.
Cfg_SPHiLim	REAL	Setpoint high limit clamp (engineering units). Valid = 0.0 to maximum positive float. Default is 100.0.
Cfg_SPLoLim	REAL	Setpoint low limit clamp (engineering units). Valid = 0.0 to maximum positive float. Default is 0.0.
Cfg_RaiseDB	REAL	Deadband for the raise output (above raise limit). Valid = 0.0 to (Cfg_SPHiLim - Cfg_SPLoLim). Default is 1.0.
Cfg_LowerDB	REAL	Deadband for the lower output (below lower limit). Valid = 0.0 to (Cfg_SPHiLim - Cfg_SPLoLim). Default is 1.0.
Cfg_RateTime	REAL	Process variable rate of change time base (seconds), 1.0 = /second, 60.0 = /minute, 3600.0 = /hour. Valid = 0.0 to maximum positive float. Default is 1.0.
Cfg_HiDevLim	REAL	High deviation status threshold (engineering units). Valid = 0.0 to maximum positive float. Default is 100.0.

Public Input Members	Data Type	Description
Cfg_HiDevDB	REAL	High deviation status deadband (engineering units). Valid = 0.0 to high deviation threshold. Default is 1.0.
Cfg_LoDevLim	REAL	Low deviation status threshold (engineering units). Valid = minimum negative float to 0.0. Default is -99.0.
Cfg_LoDevDB	REAL	Low deviation status deadband (engineering units). Valid = 0.0 to -(low deviation threshold). Default is 1.0.
Cfg_HiDevGateDly	REAL	High deviation status gate delay (seconds). Time .Inp_HiDev must be 1 before high deviation condition is checked. Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_LoDevGateDly	REAL	Low deviation status gate delay (seconds). Time .Inp_LoDev must be 1 before low deviation condition is checked. Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_HiRoCIncrLim	REAL	Program - entered high rate of change (increasing) status threshold (engineering units / rate time). Valid = 0.0 to maximum positive float. Default is 100.0.
Cfg_HiRoCIncrDB	REAL	High rate of change (increasing) status deadband (engineering units / rate time). Valid = 0.0 to Cfg_HiRoCIncrLim. Default is 1.0.
Cfg_HiRoCIncrGateDly	REAL	High rate of change (increasing) status gate delay (seconds). Valid = 0.0 to 2147483.0. Default is 0.0.
Cfg_HiRoCDecrLim	REAL	Program - entered high rate of change (decreasing) status threshold (engineering units / rate time). Valid = 0.0 to maximum positive float. Default is 100.0.
Cfg_HiRoCDecrDB	REAL	High rate of change (decreasing) status deadband (engineering units / rate time). Valid = 0.0 to Cfg_HiRoCDecrLim. Default is 1.0.
Cfg_HiRoCDecrGateDly	REAL	High rate of change (decreasing) status gate delay (seconds). Valid = 0.0 to 2147483.0. Default is 0.0.
Cfg_ExtAcqAsLevel	BOOL	1 = XCmd_Acq used as level (1 = Acquire, 0 = Release). Default is false.
Cfg_ExtOverLock	BOOL	1 = External supersedes program / Operator Lock, 0 = Don't override Lock. Default is false.
Cfg_HasExt	BOOL	1 = External exists, can be selected. Default is false.
Cfg_HasMaint	BOOL	1 = Maintenance exists, can be selected. Default is true.
Cfg_HasMaintOoS	BOOL	1 = Maintenance out of service exists, can be selected. Default is true.
Cfg_HasOper	BOOL	1 = Operator (unlocked) exists, can be selected. Default is true.
Cfg_HasOperLocked	BOOL	1 = Operator locked exists, can be selected. Default is true.
Cfg_HasProg	BOOL	1 = Program (unlocked) exists, can be selected. Default is true.
Cfg_HasProgLocked	BOOL	1 = Program locked exists, can be selected. Default is true.
Cfg_OvrdOverLock	BOOL	1 = Override supersedes program / operator lock, 0 = Don't override lock. Default is true.
Cfg_PCmdLockAsLevel	BOOL	1 = .PCmd_Lock used as a Level (1 = Lock, 0 = Unlock). Default is false.

Public Input Members	Data Type	Description
Cfg_AllowDisable	BOOL	1 = Allow maintenance to disable alarms. Default is true.
Cfg_AllowShelve	BOOL	1 = Allow operator to shelve alarms. Default is true.
Cfg_PCmdPriority	BOOL	1 = Program commands take priority, 0 = Operator commands take priority. Default is false.
Cfg_PCmdProgAsLevel	BOOL	1 = PCmd_Prog used as a level. Default is false.
Cfg_ProgNormal	BOOL	Normal source: 1 = Program if no requests, 0 = Operator if no requests. Default is false.
Cfg_ProgPwrUp	BOOL	1 = Power up to program mode, 0 = Power up to operator mode. Default is false.
Cfg_CnfrmReqd	SINT	Operator Command Confirmation Required. Represents the type of command confirmation required. 0 = None, 1 = Command confirmation required, 2 = Performer e-signature required, 3 = Performer and approver e-signature required. Default is 0.
PSet_LowerSP	REAL	Program setting for raise setpoint (engineering units). Valid = Any float. Default is 30.0.
PSet_RaiseSP	REAL	Program setting for raise setpoint (engineering units). Valid = Any float. Default is 20.0.
PSet_Owner	DINT	Program owner request ID (non-zero) or release (zero). Valid = Any integer greater or equal to 0. Default is 0.
XSet_LowerSP	REAL	External setting for lower setpoint (engineering units). Valid = Any float. Default is 30.0.
XSet_RaiseSP	REAL	External setting for raise setpoint (engineering units). Valid = Any float. Default is 20.0.
PCmd_Raise	BOOL	Program command to set output to raise, when in program manual. 1 = Raise. The instruction clears this operand automatically. Default is false.
PCmd_Lower	BOOL	Program command to set output to lower, when in program manual. 1 = Lower. The instruction clears this operand automatically. Default is false.
PCmd_None	BOOL	Program command to clear raise, lower outputs, when in program manual. 1 = Clear .PCmd_Raise and PCmd_Lower. The instruction clears this operand automatically. Default is false.
PCmd_Auto	BOOL	Program command to select automatic loop mode. 1 = Select automatic loop mode. The instruction clears this operand automatically. Default is false.
PCmd_Man	BOOL	Program command to select manual loop mode. 1 = Select manual loop mode. The instruction clears this operand automatically. Default is false.
PCmd_Reset	BOOL	Program Command to reset shed latches and cleared alarms. 1 = Reset shed latches and cleared alarms. The instruction clears this operand automatically. Default is false.
PCmd_Lock	BOOL	Program command to lock program (disallow operator). The instruction clears this operand automatically if Cfg_PCmdLockAsLevel = 0. Default is false.

Public Input Members	Data Type	Description
PCmd_Normal	BOOL	Program command to select normal command source (operator or program). The instruction clears this operand automatically. Default is false.
PCmd_Oper	BOOL	Program command to select operator (program to operator). The instruction clears this operand automatically. Default is false.
PCmd_Prog	BOOL	Program command to select program (operator to program). The instruction clears this operand automatically. Default is false.
PCmd_Unlock	BOOL	Program command to unlock Program (allow Operator to acquire). The instruction clears this operand automatically. Default is false.
XCmd_Raise	BOOL	External command to set output to raise, when in external manual. 1 = Lower. The instruction clears this operand automatically. Default is false.
XCmd_Lower	BOOL	External command to set output to lower, when in external manual. 1 = Lower. The instruction clears this operand automatically. Default is false.
XCmd_None	BOOL	External command to clear raise, lower outputs, when in external manual. 1 = Clear XCmd_Raise and XCmd_Lower. The instruction clears this operand automatically. Default is false.
XCmd_Auto	BOOL	External command to select automatic loop mode. 1 = select automatic loop mode. The instruction clears this operand automatically. Default is false.
XCmd_Man	BOOL	External command to select manual loop mode. 1 = Select manual loop mode. The instruction clears this operand automatically. Default is false.
XCmd_Reset	BOOL	External command to reset shed latches and cleared alarms. 1 = Reset shed latches and cleared alarms. The instruction clears this operand automatically. Default is false.
XCmd_ResetAckAll	BOOL	External command to acknowledge and reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.
XCmd_Acq	BOOL	External command to acquire ownership (operator/program/override/maintenance to external). The instruction clears this operand automatically if Cfg_ExtAcqAsLevel = 0. Default is false.
XCmd_Rel	BOOL	External command to release ownership, if Cfg_ExtAcqAsLevel = 0 (external to operator/program/override/maintenance). The instruction clears this operand automatically. Default is false.

Public Output Members	Data Type	Description
EnableOut	BOOL	Enable Output - This output state always reflects EnableIn input state.
Out_Raise	BOOL	Output to drive process variable to raise.
Out_Lower	BOOL	Output to drive process variable to lower.
Out_Q	BOOL	1 = Setpoint raised, 0 = Setpoint lowered.
Out_QNot	BOOL	1 = Setpoint lowered, 0 = Setpoint raised.

Public Output Members	Data Type	Description
Out_OwnerSts	DINT	Status of command source, owner command handshake and ready status: Out_OwnerSts.0 = None, Out_OwnerSts.10 = Operator lock, Out_OwnerSts.11 = Operator unlock, Out_OwnerSts.12 = Program lock, Out_OwnerSts.13 = Program unlock, Out_OwnerSts.14 = Acquire maintenance, Out_OwnerSts.15 = Release maintenance , Out_OwnerSts.16 = Acquire external, Out_OwnerSts.17 = Release external, Out_OwnerSts.18 = Has maintenance, Out_OwnerSts.19 = External override lock, Out_OwnerSts.20 = Has External, Out_OwnerSts.21 = Has operator, Out_OwnerSts.22 = Has program, Out_OwnerSts.30 = Not ready.
Val_PV	REAL	Process variable value (engineering units). (scaled range between minimum and maximum).
Val_RoC	REAL	Process variable rate of change value (engineering units / rate time).
Val_LowerSP	REAL	Accepted value for lower setpoint (engineering units).
Val_RaiseSP	REAL	Accepted value for raise setpoint (engineering units).
Val_PVEUMin	REAL	Minimum of scaled range, minimum between Cfg_PVEUMin and Cfg_PVEUMax.
Val_PVEUMax	REAL	Maximum of scaled range, maximum between Cfg_PVEUMin and Cfg_PVEUMax.
Val_Owner	DINT	Current object owner ID, 0 = Not owned.
SrcQ_IO	DINT	Source and quality of primary input / output (enumeration).
SrcQ	DINT	Source and quality of primary value / status (enumeration).
Sts_Initialized	BOOL	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.
Sts_Raise	BOOL	1 = Process variable below raise setpoint, Out_Raise = 1.
Sts_Lower	BOOL	1 = Process variable above lower setpoint, Out_Lower = 1.
Sts_Auto	BOOL	1 = Current loop mode is automatic.
Sts_Man	BOOL	1 = Current loop mode is manual.
Sts_Q	BOOL	1 = Out_Q = 1, Out_QNot = 0.
Sts_Available	BOOL	1 = Loop available for manipulation in program mode.
Sts_NotRdy	BOOL	1 = Device not ready, see detail bits for reason.
Sts_NrdyCfgErr	BOOL	1 = Device not ready: Configuration error.
Sts_NrdyOoS	BOOL	1 = Device is not ready: Device disabled by maintenance.
Sts_AlmInh	BOOL	1 = An alarm is inhibited, disabled or suppressed (Display icon on HMI).
Sts_Err	BOOL	1 = Error in configuration: See detail bits (Sts_ErrXXX) for reason.
Sts_ErrEU	BOOL	1 = Error in configuration: Scaled engineering units Cfg_PVEUMin = Cfg_PVEUMax.
Sts_ErrRateTime	BOOL	1 = Error in configuration: Process variable rate of change time base.
Sts_ErrAlm	BOOL	1 = Error in configuration: Alarm minimum on time or severity.
Sts_OoS	BOOL	1 = Out of service is selected (supersedes maintenance, override, external, program, operator).
Sts_Prog	BOOL	1 = Program mode is selected.
Sts_RdyReset	BOOL	1 = A latched alarm or shed condition is ready to be reset.
Sts_RdyAck	BOOL	1 = An alarm is ready to be acknowledged.
Sts_HiDevCmp	BOOL	Process variable high deviation comparison result = 1.
Sts_HiDevGate	BOOL	Process variable high deviation gate delay status, 1 = Done.
Sts_HiDev	BOOL	1 = Analog input deviation is above high limit. There is a predefined default discrete Logix tag-based alarm for the status. Set standard configuration members of the discrete Logix tag-based alarm. Access alarm elements in this format: PDBCTag.@Alarms.Alm_HiDev.AlarmElement.

Public Output Members	Data Type	Description
Sts_LoDevCmp	BOOL	Process variable low deviation comparison result = 1.
Sts_LoDevGate	BOOL	Process variable low deviation gate delay status, 1 = Done.
Sts_LoDev	BOOL	1 = Analog input deviation is below low limit. There is a predefined default discrete Logix tag-based alarm for the status. Set standard configuration members of the discrete Logix tag-based alarm. Access alarm elements in this format: PDBCTag.@Alarms.Alm_HiRoCIncr.AlarmElement.
Sts_HiRoCIncrCmp	BOOL	Process variable high rate of change (increasing) comparison result = 1.
Sts_HiRoCIncrGate	BOOL	Process variable high rate of change (increasing) gate delay status, 1 = Done.
Sts_HiRoCIncr	BOOL	1 = Analog input PV rate of change (increasing) is above high limit. There is a predefined default discrete Logix tag-based alarm for the status. Set standard configuration members of the discrete Logix tag-based alarm. Access alarm elements in this format: PDBCTag.@Alarms.Alm_HiRocIncr.AlarmElement.
Sts_HiRoCDecrCmp	BOOL	Process variable high rate of change (decreasing) comparison result = 1.
Sts_HiRoCDecrGate	BOOL	Process variable high rate of change (decreasing) gate delay status, 1 = Done.
Sts_HiRoCDecr	BOOL	1 = Analog input PV rate of change (decreasing) is above high limit. There is a predefined default discrete Logix tag-based alarm for the status. Set standard configuration members of the discrete Logix tag-based alarm. Access alarm elements in this format: PDBCTag.@Alarms.Alm_HiRocDecr.AlarmElement.
Sts_eNotify	SINT	All alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyAll	SINT	All alarm status enumerated values including related objects: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyHiDev	SINT	High Deviation Gate alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.

Public Output Members	Data Type	Description
Sts_eNotifyHiRoCDecr	SINT	High Rate of Change (Decreasing) alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyHiRoCIncr	SINT	High Rate of Change (Increasing) alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyLoDev	SINT	Low Deviation alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_UnackAlmCount	DINT	Count of unacknowledged alarms.
Sts_Alm	BOOL	1 = An alarm is active.
Sts_ErrHiDevGateDly	BOOL	1 = Error in configuration: Cfg_HiDevGateDly value is invalid.
Sts_ErrLoDevGateDly	BOOL	1 = Error in configuration: Cfg_LoDevGateDly value is invalid.
Sts_ErrHiRoCIncrGateDly	BOOL	1 = Error in configuration: Cfg_HiRoCIncrGateDly value is invalid.
Sts_ErrHiRoCDecrGateDly	BOOL	1 = Error in configuration: Cfg_HiRoCDecrGateDly value is invalid.
Sts_Oper	BOOL	1 = Operator mode is selected.
Sts_Maint	BOOL	1 = Maintenance is selected (supersedes override, external, program, operator).
Sts_Ext	BOOL	1 = External is selected (supersedes program and operator).
Sts_Ovrd	BOOL	1 = Override is selected (supersedes external, program, operator).
Sts_eFault	INT	Device Fault Status: 0 = None, 1 = Low deviation, 2 = High deviation, 3 = High rate of change (decreasing), 4 = High rate of change (increasing), 5 = Input source bad, 6 = Configuration error.

Public Output Members	Data Type	Description
Sts_eSts	INT	Device status: 0 = Deadband, Q is off, 1 = Deadband Q is on, 2 = Above lower SP, 3 = Below lower SP, 4-7 = same, in manual, 8 = Out of service
Sts_bSrc	INT	Active selection bitmap (for HMI totem pole with command source request selection): Sts_bSrc.0 = Hand, Sts_bSrc.1 = Programmed out of service (rung false), Sts_bSrc.2 = Maintenance out of service, Sts_bSrc.3 = Maintenance, Sts_bSrc.4 = Override, Sts_bSrc.5 = External, Sts_bSrc.6 = Program locked, Sts_bSrc.7 = Program, Sts_bSrc.8 = Operator locked, Sts_bSrc.9 = Operator.
Sts_eSrc	INT	The current command source enumerated values: 0 = Logic not in use, 4 = Hand, 8 = Maintenance, 16 = Override, 32 = Program, 33 = Program locked, 34 = Program by default (Normal), 64 = Operator, 65 = Operator locked, 66 = Operator by default (Normal), 128 = Maintenance Out of Service, 129 = Programmed Out of Service (rung false), 256 = External.
Sts_ExtReqInh	BOOL	1 = External request inhibited, cannot get to external from current state.
Sts_Hand	BOOL	1 = Hand is selected (supersedes out of service, maintenance, override, external, program, operator).
Sts_MAcqRcvd	BOOL	1 = Maintenance acquire command received this scan (Read only).
Sts_Normal	BOOL	1 = Selection equals the normal (program or operator).
Sts_OperLocked	BOOL	1 = Operator is selected and locked.
Sts_ProgLocked	BOOL	1 = Program is selected and locked.
Sts_ProgOperLock	BOOL	Program/operator lock (latch) state, 1 = Locked, 0 = Unlocked.
Sts_ProgOperSel	BOOL	Program/operator selection (latch) state: 1 = Program, 0 = Operator.
Sts_ProgReqInh	BOOL	1 = Program request inhibited, cannot get to program from current state.
XRdy_Acq	BOOL	1 = Ready for XCmd_Acq, enable HMI button.
XRdy_Rel	BOOL	1 = Ready for XCmd_Rel, enable HMI button (Read Only).
XRdy_Reset	BOOL	1 = Ready for XCmd_Reset, enable HMI button.
Rdy_ResetAckAll	BOOL	1 = Ready for XCmd_ResetAckAll, enable HMI button.
XRdy_Auto	BOOL	1 = Ready for XCmd_Auto.
XRdy_Lower	BOOL	1 = Ready for XCmd_Lower.

Public Output Members	Data Type	Description
XRdy_Man	BOOL	1 = Ready for XCmd_Man.
XRdy_None	BOOL	1 = Ready for XCmd_None.
XRdy_Raise	BOOL	1 = Ready for XCmd_Raise.
Private Input Members	Data Type	Description
HMI_BusObjIndex	DINT	HMI bus object index Default is 0.
MCmd_Acq	BOOL	Maintenance command to acquire ownership (operator / program / external / override to maintenance). The instruction clears this operand automatically. Default is false.
MCmd_IS	BOOL	Maintenance command to select in service. The instruction clears this operand automatically. Default is false.
MCmd_OoS	BOOL	Maintenance command to select out of service. The instruction clears this operand automatically. Default is false.
MCmd_Rel	BOOL	Maintenance command to release ownership (maintenance to operator / program / external / override). The instruction clears this operand automatically. Default is false.
OCmd_Auto	BOOL	Operator command to select automatic loop mode. Default is false.
OCmd_Lock	BOOL	Operator command to lock operator (disallow program). The instruction clears this operand automatically. Default is false.
OCmd_Lower	BOOL	Operator command to set output to lower (in manual). Default is false.
OCmd_Man	BOOL	Operator command to select manual loop mode. Default is false.
OCmd_None	BOOL	Operator command to clear raise, lower outputs (in manual). Default is false.
OCmd_Normal	BOOL	Operator command to select normal (operator or program). The instruction clears this operand automatically. Default is false.
OCmd_Oper	BOOL	Operator command to select operator (program to operator). The instruction clears this operand automatically. Default is false.
OCmd_Prog	BOOL	Operator command to select program (operator to program). The instruction clears this operand automatically. Default is false.
OCmd_Raise	BOOL	Operator command to set output to raise (in manual). Default is false.
OCmd_Reset	BOOL	Operator command to reset shed latches and cleared alarms. Default is false.
OCmd_ResetAckAll	BOOL	Operator command to reset shed latches and cleared alarms, plus acknowledge alarms. Default is false.
OCmd_Unlock	BOOL	Operator command to unlock or release ownership and allow program to acquire ownership. The instruction clears this operand automatically. Default is false.
OSet_LowerSP	REAL	Operator setting for lower setpoint (engineering units). Default is 30.0.
OSet_RaiseSP	REAL	Operator setting for raise setpoint (engineering units). Default is 20.0.

Private Output Members	Data Type	Description
MRdy_Acq	BOOL	1 = Ready for MCmd_Acq, enable HMI button.
MRdy_IS	BOOL	1 = Ready for MCmd_IS, enable HMI button.
MRdy_OoS	BOOL	1 = Ready for MCmd_OoS, enable HMI button.
MRdy_Rel	BOOL	1 = Ready for MCmd_Rel, enable HMI button.
ORdy_Auto	BOOL	1 = Ready for OCmd_Auto.
ORdy_Lock	BOOL	1 = Ready for OCmd_Lock, enable HMI button.
ORdy_Lower	BOOL	1 = Ready for OCmd_Lower.
ORdy_Man	BOOL	1 = Ready for OCmd_Man.
ORdy_None	BOOL	1 = Ready for OCmd_None.
ORdy_Normal	BOOL	1 = Ready for OCmd_Normal, enable HMI button.
ORdy_Oper	BOOL	1 = Ready for OCmd_Oper, enable HMI button.
ORdy_Prog	BOOL	1 = Ready for OCmd_Prog, enable HMI button.
ORdy_Raise	BOOL	1 = Ready for OCmd_Raise.
ORdy_Reset	BOOL	1 = Ready for OCmd_Reset (enables HMI button).
ORdy_ResetAckAll	BOOL	1 = Ready for OCmd_ResetAckAll (enables HMI button).
ORdy_SP	BOOL	1 = Ready for Setpoint OSets (enables data entry fields).
ORdy_Unlock	BOOL	1 = Ready for OCmd_Unlock, enable HMI button.

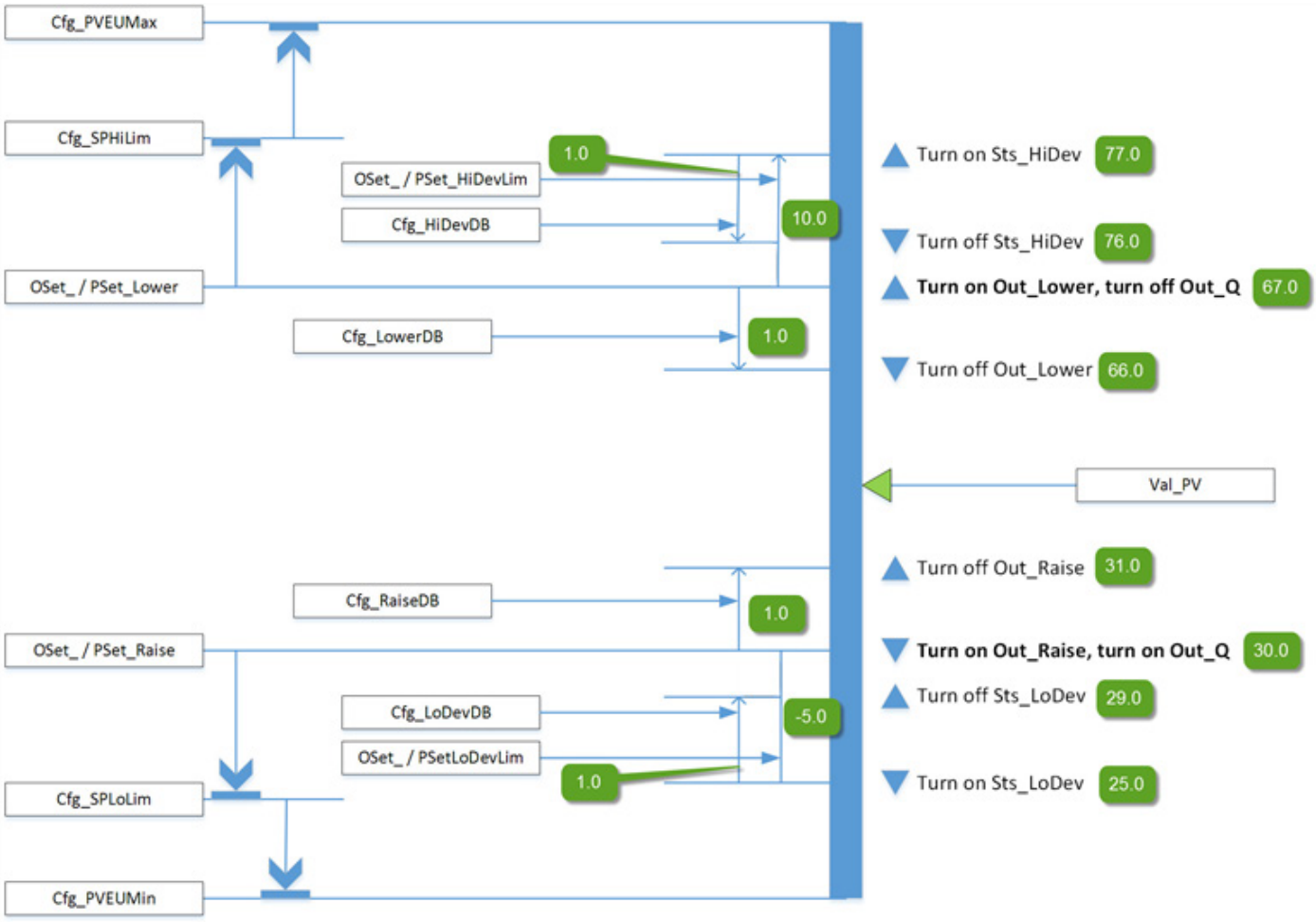
Public InOut Members	Data Type	Description
BusObj	BUS_OBJ	Bus component

BUS_OBJ Structure

Members	Data Type	Description
Inp_Cmd	DINT	Input to assert commands
Out_Cmd	DINT	Resultant commands
Inp_CmdLLH	DINT	Input for level normally high commands
Out_CmdLLH	DINT	Resultant line level high commands
Inp_Sts	DINT	Input to assert status
Out_Sts	DINT	Resultant status
Inp_CmdAck	DINT	Input to assert a command acknowledgement
Out_CmdAck	DINT	Resultant command acknowledgements
Inp_SeverityMax	DINT	Input: maximum alarm severity
Out_SeverityMax	DINT	Resultant of maximum alarm severity
Cfg_CmdMask	DINT	Propagation mask for commands
Cfg_CmdLLHMask	DINT	Propagation mask for line level high commands
Cfg_StsMask	DINT	Propagation mask for status
Ref_Index	DINT	Bus array index

Operation

This diagram illustrates the functionality of the PDBC instruction:



Configuration of Strings for HMI

Configure strings for HMI faceplates (FT View) and for the Logix Designer configuration dialog box. The strings are set to extended properties of tag items. Configure the strings in Logix Designer only.

- Description
- Label for graphic symbol
- Display Library for HMI Faceplate call-up
- Instruction name
- Area name
- URL link
- More Information
- Number of decimal places
- Output Units
- Input Units
- Allow Navigation Object Tag Name Output
- Allow Navigation Object Tag Name Input

Implementation

This illustration shows normal implementation with the input condition mapped to Inp_PV on a separate branch. This approach controls an analog process variable (PV), such as temperature, level or pressure, between upper and lower control limits by triggering one or two discrete outputs.

Monitor the PDBC Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Rung-condition-out is cleared to false. Sts_eSrc is set to 0. Sts_bSrc is set to 0.

Condition/State	Action Taken
Instruction first run	All commands that are automatically cleared each execution are cleared and ignored. Inp_OvrdCmd is set to 0 (no command). The Program/Operator selection is set based on the configuration (Cfg_ProgPwrUp). The Program or Operator lock selection is set to unlocked. The Maintenance acquired/released state is not modified and persists through a controller powerup or PROG-to-RUN transition. PSet_Owner and Out_OwnerSts are set to 0.
Rung-condition-in is false	Rung-condition-out is cleared to false. The instruction is put Out of Service if Inp_Hand=0. The output is set to Interlock CV and all alarm conditions are cleared. Latched alarms are reset. Command source selection processing proceeds except that Program and Operators commands are ignored and cleared and all ownership status bits (Sts_Maint, Sts_Ovrd, Sts_Ext, Sts_Prog and Sts_Oper) are cleared to 0. When rung-condition-in becomes true, the Program/Operator selection is set based on the configuration (Cfg_ProgPwrUp).
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	EnableOut is cleared to false. Sts_eSrc is set to 0. Sts_bSrc is set to 0.
Instruction first run	All commands that are automatically cleared on each execution are cleared and ignored. Inp_OvrdCmd is set to 0 (no command). The Program/Operator selection is set based on the configuration (Cfg_ProgPwrUp). The Program or Operator lock selection is set to unlocked. The Maintenance acquired/released state is not modified and persists through a controller powerup or PROG-to-RUN transition. PSet_Owner and Val_Owner are set to 0.
Instruction first scan	See Instruction first run in the Function Block Diagram table.
EnableIn is false	EnableOut is cleared to false. The instruction is put Out of Service if Inp_Hand=0. The output is set to Interlock CV and all alarm conditions are cleared. Latched alarms are reset. Command source selection processing proceeds except that Program and Operators commands are ignored and cleared and all ownership status bits (Sts_Maint, Sts_Ovrd, Sts_Ext, Sts_Prog and Sts_Oper) are cleared to 0. When rung-condition-in becomes true, the Program/Operator selection is set based on the configuration (Cfg_ProgPwrUp).
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

In Structured Text, EnableIn is always true during normal scan. The instruction executes when it is in the control path activated by the logic.

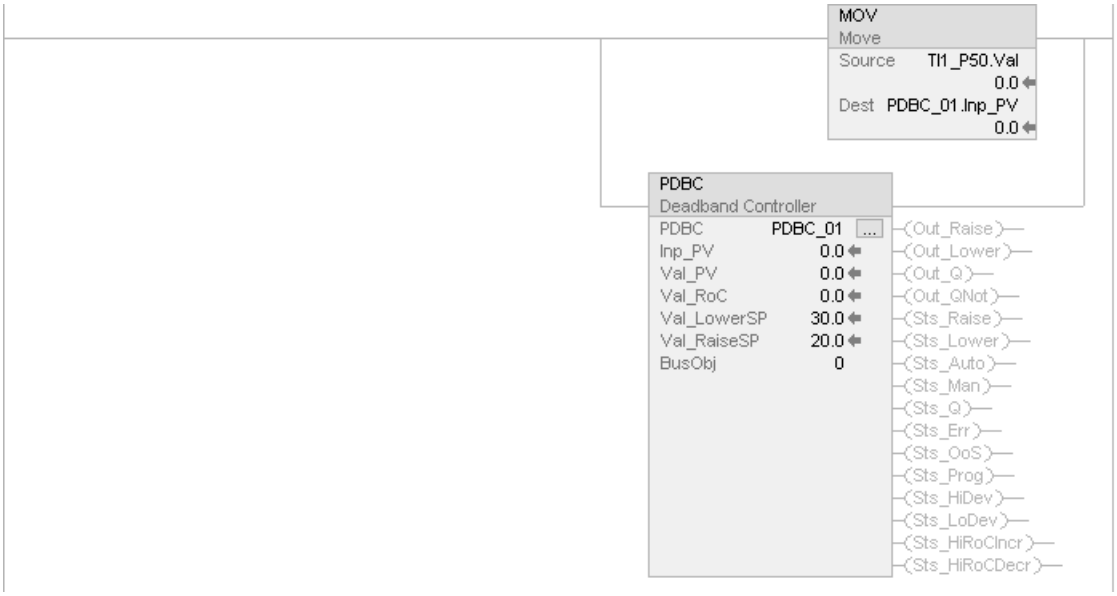
Condition/State	Action Taken
Prescan	See Prescan in the Function Block Diagram table.
Instruction first run	See Instruction first run in the Function Block Diagram table.
EnableIn is true	See EnableIn is true in the Function Block Diagram table.
Postscan	See Postscan in the Function Block Diagram table.

Example

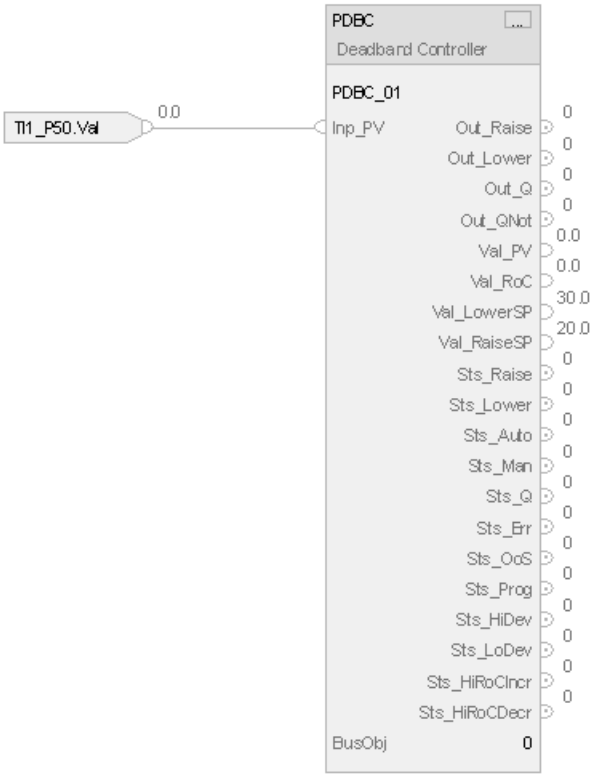
In this example, tag TI1_P50 is the temperature value monitored by the PDBC instruction. This tag provides a real indication of analog PV value.

Inp_PV is connected to the analog values tag (TI1_P50.Val) that comes from the Value output of the PAI instruction instance.

Ladder Diagram



Function Block Diagram



Structured Text

PDBC01.Inp_PV := TI1_P50.Out;

PDBC(PDBC_01, 0);

See also

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Process Discrete Input (PDI)

This information applies to the ControlLogix 5380P and 5580P controllers.

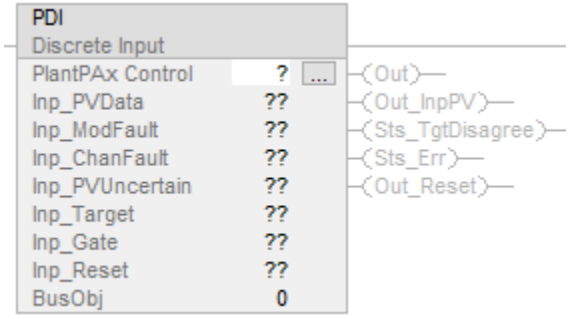
The Process Discrete Input (PDI) instruction monitors a discrete (true or false) input, and checks for alarm conditions. Use the PDI instruction to process a signal from a channel of a discrete input module. Use the PDI instruction with any discrete (BOOL) signal.

The PDI instruction provides these capabilities:

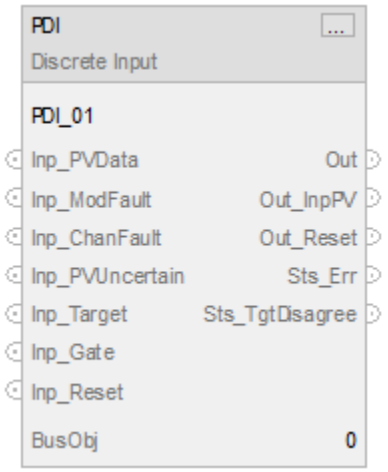
- De-bounce of the discrete input signal to filter out fast status changes by specifying a minimum time status must maintain state.
- Display of the input state; the 0-state and 1-state names are configurable. The input state is also displayed independently, even when the input is substituted.
- Target Disagree status based on comparing the input state against a target, or normal, state. The Target Disagree status is enabled by a gating input signal with a configurable gate delay. The Target Disagree status on and off delays are configurable. The Target Disagree status has an associated tag-based alarm.
- Handle a process variable (PV) fault input by displaying the fault to the operator. The PV fault has an associated tag-based alarm.
- The operator can select and manually enter a substitute PV. This manual override is made clearly visible to the operator. Optionally, the user can configure the substitute PV signal to track the Target input so that no Target Disagree status or alarm is generated.
- Support for a virtual PV for use in instruction testing, demonstration, or operator training.

Available Languages

Ladder Diagram



Function Block Diagram



Structured Text

PDI(PDI tag, BusObj);

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PDI	P_DISCRETE_INPUT	tag	Data structure required for proper operation of instruction.
BusObj	BUS_OBJ	tag	Bus component. May be null.

P_DISCRETE_INPUT Structure

Public members are standard, visible tag members that are programmatically accessible. Private, or hidden, members are used in HMI faceplates and are not programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	Description
EnableIn	BOOL	Enable input. Ladder Diagram: Corresponds to the rung condition. Default is true.
Inp_InitializeReq	BOOL	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request to re-initialize. The instruction clears this operand automatically. Default is true.
Inp_PVData	BOOL	Input signal (process variable) from device. When EnableIn is false the instruction executes and uses the inverse of the Inp_PVData signal for processing. Default is true.
Inp_ModFault	BOOL	1 = I/O module failure or module communication status bad, 0 = OK. Default is false.
Inp_ChanFault	BOOL	1 = I/O channel fault or failure, 0 = OK. Default is false.
Inp_PVUncertain	BOOL	Indicates the channel data accuracy is undetermined. 1 = The channel data is uncertain. This input sets Sts_PVUncertain if not in Virtual. Default is false.
Inp_PVNotify	SINT	Related PV object alarm priority and acknowledgement status. 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged. Default = 0.
Inp_Target	BOOL	Target state of input. Input Inp_PVData is compared with the target state using Gate function. If not in target state, the Target Disagree status (Sts_TgtDisagree) is set to 1 and Target Disagree tag-based alarm is raised (if enabled). Default is true.
Inp_Gate	BOOL	The gate input used for status detection. 1 = Target Disagree monitoring is enabled. 0 = Target Disagree detection is disabled and the Target Disagree status output is forced off. Default is true.
Inp_Reset	BOOL	1 = Reset Shed Latches and Cleared Alarms. Default is false.
Cfg_AllowDisable	BOOL	1 = Allow Maintenance to disable alarms. Default is true.
Cfg_AllowShelve	BOOL	1 = Allow Operator to shelve alarms. Default is true.

Public Input Members	Data Type	Description
Cfg_NoSubstPV	BOOL	Disables the maintenance substitution feature. 0 = The Substitute PV Maintenance function is enabled, 1 = The Substitute PV Maintenance function is disabled. When Cfg_NoSubstPV is 0, the commands MCmd_SubstPV and MCmd_InpPV are used to select the input PV or the substitute PV. Sts_SubstPV is set to 1 when the substitute PV is selected. Default is false.
Cfg_SubstTracksTarget	BOOL	1 = The substitute PV tracks Inp_Target, 0 = The substitute PV is set by MSet_SubstPV. Default is false.
Cfg_NormTextVis	BOOL	1 = The state text is displayed in Normal state, 0 = The state text is hidden in Normal state. Default is true.
Cfg_HasMoreObj	BOOL	1 = Tells HMI an object with more info is available. Default is false.
Cfg_Debounce	REAL	Minimum time status must maintain state (seconds). Debounces the input PV, ensuring that the status stays in each state a minimum time. Valid = 0.0 to 2147483.0 seconds. Default = 0.0.
Cfg_GateDly	REAL	Target Disagree Gate delay (seconds). Time Inp_Gate must be 1 before Target Disagree condition is checked. Valid = 0.0 to 2147483.0 seconds. Default = 0.0.
Cfg_TgtDisagreeOffDly	REAL	Minimum time for input to agree with target to clear status (seconds). Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_TgtDisagreeOnDly	REAL	Minimum time for input to disagree with target to raise status (seconds). Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_CnfrmReqd	SINT	Operator command confirmation required. Represents the type of command confirmation required. 0 = None, 1 = Command confirmation required, 2 = Performer e-signature required, 3 = Performer and approver e-signature required. Default is 0.
Set_VirtualPV	BOOL	PV used in Virtual (Sts_Virtual is 1). If the instruction is not in Virtual (Inp_Virtual is 0), the Set_VirtualPV input tracks the input PV (Inp_PVData) for bumpless transfer into Virtual. Default is false.
PCmd_Virtual	BOOL	Program command to select Virtual (simulated) device operation. The instruction clears this operand automatically. Default is false.
PCmd_Physical	BOOL	Program command to select Physical device operation (not simulated). The instruction clears this operand automatically. Default is false.
PCmd_Reset	BOOL	Program command to reset all alarms and latched shed conditions requiring reset. The instruction clears this operand automatically. Default is false.
XCmd_Reset	BOOL	External command to reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.
XCmd_ResetAckAll	BOOL	External command to acknowledge and reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.

Public Output Members	Data Type	Description
EnableOut	BOOL	Enable output. This output state always reflects EnableIn input state.
Out	BOOL	Discrete input status (including de-bounce and manual override, if used). 0 = The discrete input is Off, 1 = The discrete input is On.
Out_InpPV	BOOL	Echo of Inp_PVData (actual raw or virtual input).
Out_Reset	BOOL	1 = Reset command has been received and accepted.
Sts_Initialized	BOOL	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.
Sts_PVUncertain	BOOL	Indicates the channel data accuracy is undetermined. 1 = The channel data is uncertain. This output is set by Inp_PVUncertain (if not in Virtual).
Sts_SubstPV	BOOL	1 = Using substitute PV (Override).
Sts_InpPV	BOOL	1 = Using input PV (Normal).
Sts_Virtual	BOOL	1 = Using virtual PV instead of the input from the device (Inp_PVData) to calculate output. 0 = The instruction uses input operand Inp_PVData to calculate output.
SrcQ_IO	SINT	Source and quality of primary input or output (enumerated): 0 = Good, live, confirmed good, 1 = Good, live, assumed good, 2 = Good, no feedback, assumed good, 8 = Test, virtualized, 9 = Test, loopback, 10 = Test, manually entered, 16 = Uncertain, live, off-spec, 17 = Uncertain, substituted at device or bus, 18 = Uncertain, substituted at instruction, 19 = Uncertain, using last known good, 20 = Uncertain, using replacement value, 32 = Bad, signal failure, 33 = Bad, channel fault, 34 = Bad, module or communication fault, 35 = Bad, invalid configuration.
SrcQ	SINT	Source and quality of primary value or status (enumerated): 0 = Good, live, confirmed good, 1 = Good, live, assumed good, 2 = Good, no feedback, assumed good, 8 = Test, virtualized, 9 = Test, loopback, 10 = Test, manually entered, 16 = Uncertain, live, off-spec, 17 = Uncertain, substituted at device or bus, 18 = Uncertain, substituted at instruction, 19 = Uncertain, using last known good, 20 = Uncertain, using replacement value, 32 = Bad, signal failure, 33 = Bad, channel fault, 34 = Bad, module or communication fault, 35 = Bad, invalid configuration.

Public Output Members	Data Type	Description
Sts_eSts	SINT	Device confirmed status values: 0 = PV Good, 1 = PV uncertain, 2 = PV bad, 3 = PV substituted.
Sts_eFault	SINT	Device fault status values: 0 = None, 1 = Target disagree, 2 = Configuration error.
Sts_eNotify	SINT	All alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyAll	SINT	All alarm status enumerated values including related objects: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyIOFault	SINT	IOFault alarm status enumerated values. 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.

Public Output Members	Data Type	Description
Sts_eNotifyTgtDisagree	SINT	TgtDisagree alarm status enumerated values. 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_UnackAlmCount	DINT	Count of unacknowledged alarms.
Sts_MaintByp	BOOL	1 = The device has a Maintenance Bypass function active.
Sts_Err	BOOL	1 = Error in configuration: See detail bits (Sts_Errxxx) for reason.
Sts_ErrGateDly	BOOL	1 = Error in configuration: Cfg_GateDly value is invalid.
Sts_ErrTgtDisagreeOffDly	BOOL	1 = Error in configuration: Cfg_TgtDisagreeOffDly value is invalid.
Sts_ErrTgtDisagreeOnDly	BOOL	1 = Error in configuration: Cfg_TgtDisagreeOnDly value is invalid.
Sts_ErrDebounce	BOOL	1 = Error in configuration: Cfg_Debounce value is invalid.
Sts_ErrAlm	BOOL	1 = Error in tag-based alarm settings.
Sts_Alm	BOOL	1 = An alarm is active.
Sts_AlmInh	BOOL	1 = An alarm is shelved or disabled.
Sts_IOFault	BOOL	IO Fault Status (0 = OK, 1 = Bad). 1 = Channel data is inaccurate. This output is set by Inp_IOFault if not in Virtual. There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PDITag.@Alarms.Alm_IOFault.AlarmElement
Sts_TgtDisagreeCmp	BOOL	Input versus Target comparison result before gating. 1 = The input does not match its target.
Sts_TgtDisagreeGate	BOOL	Target Disagree Gate Delay Status. 1 = The target disagree gate is open.
Sts_TgtDisagree	BOOL	Gated input versus target comparison result. 1 = Input is not in target state. There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PDITag.@Alarms.Alm_TgtDisagree.AlarmElement
Sts_RdyAck	BOOL	1 = An alarm is ready to be acknowledged.
Sts_RdyReset	BOOL	1 = A latched alarm or shed condition is ready to be reset.
XRdy_Reset	BOOL	1 = Ready for XCmd_Reset, enable HMI button.
XRdy_ResetAckAll	BOOL	1 = Ready for XCmd_ResetAckAll, enable HMI button.

Private Input Members	Data Type	Description
HMI_BusObjIndex	DINT	This object's index in the bus array, for use by HMI display. Default is 0.
MSet_SubstPV	BOOL	Maintenance-entered substitute PV that overrides input PV when Sts_SubstPV is 1. If not using the substitute (Sts_SubstPV is false), the MSet_SubstPV setting tracks the Out value for bumpless transfer from input PV to substitute PV. Default is false.
MCmd_SubstPV	BOOL	Maintenance command to use Substitute PV (Override input). The instruction clears this operand automatically. Default is false.

Private Input Members	Data Type	Description
MCmd_InpPV	BOOL	Maintenance command to use Input PV (Normal). The instruction clears this operand automatically. Default is false.
MCmd_Physical	BOOL	Maintenance command to select Physical device operation (not simulated). The instruction clears this operand automatically. Default is false.
MCmd_Virtual	BOOL	Maintenance command to select Virtual (simulated) device operation. The instruction clears this operand automatically. Default is false.
OCmd_Reset	BOOL	Operator command to reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.
OCmd_ResetAckAll	BOOL	Operator command to acknowledge and reset all alarms and latched shed conditions. The use of OCmd_ResetAckAll is restricted to HMI. The instruction clears this operand automatically. Default is false.

Private Output Members	Data Type	Description
MRdy_SubstPV	BOOL	1 = The instruction is ready for SubstPV command.
MRdy_InpPV	BOOL	1 = The instruction is ready for InpPV command.
MRdy_Physical	BOOL	1 = Ready for MCmd_Physical, enable HMI button.
MRdy_Virtual	BOOL	1 = Ready for MCmd_Virtual, enable HMI button.
ORdy_Reset	BOOL	1 = A latched alarm or shed condition is ready to be reset.
ORdy_ResetAckAll	BOOL	1 = A latched alarm or shed condition is ready to be reset or acknowledged.

Public InOut Members	Data Type	Description
BusObj	BUS_OBJ	Bus component

Alarms

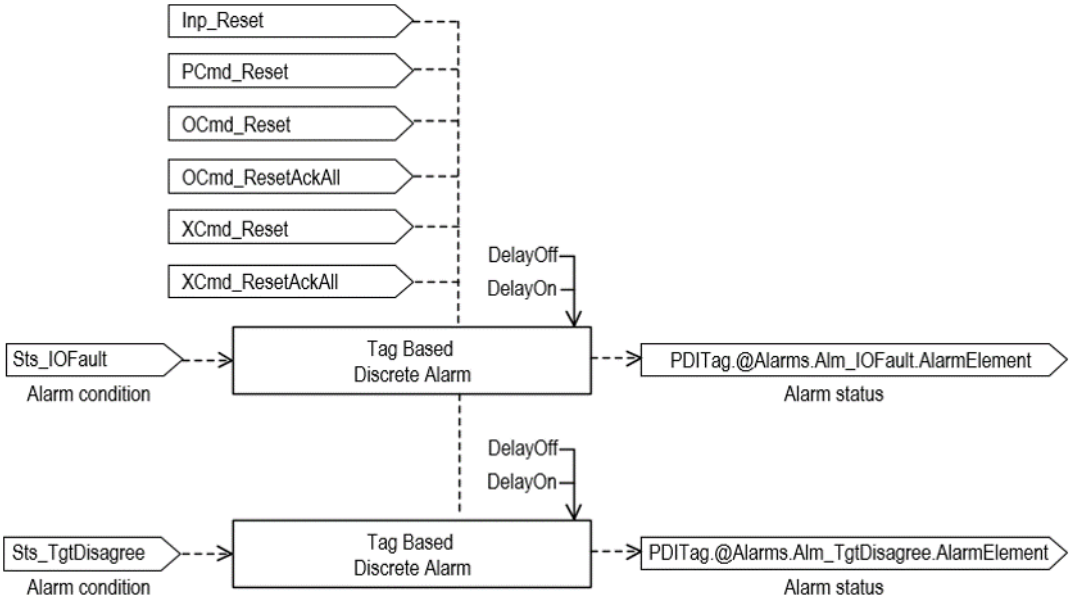
Discrete tag-based alarms are defined for these members.

Member	Alarm Name	Description
Sts_TgtDisagree	Alm_TgtDisagree	Target Disagree status.
Sts_IOFault	Alm_IOFault	I/O Fault status (not generated when PV Substitution is active).

Mark the alarm as used or unused and set standard configuration members of the discrete Logix Tag based alarm. Access alarm elements using this format:

PDITag.@Alarms.AlarmName.AlarmElement

There are Program, Operator, and External commands that enable the Reset and Reset & Acknowledge of all alarms of the instruction (Alarm Set) at the same time. This diagram shows how the commands interact with the instruction.



Operation

The Gate function provides the ability to raise an abnormal condition (alarm condition) when another condition is true. For example, a high vibration switch should only generate an alarm when the associated motor is running long enough to stabilize. The Inp_Gate input must be set to 1 (its default value) and the tag-based alarm for Sts_TgtDisagree enabled for alarm to occur. The alarm will not occur until the Inp_Gate input has been set for the Gate Delay (Cfg_GateDly) time.

The alarm is generated when the Inp_PVData (process variable) input is different from the Inp_Target (target) input. The Target indicates the normal condition. For example, a flow switch should indicate flow when a pump is running and should not indicate flow when a pump is stopped. The pump run status is used as the Target input, and when the switch does not match the target (within the allotted time), the Alarm (Flow Loss / Switch Failure) is generated.

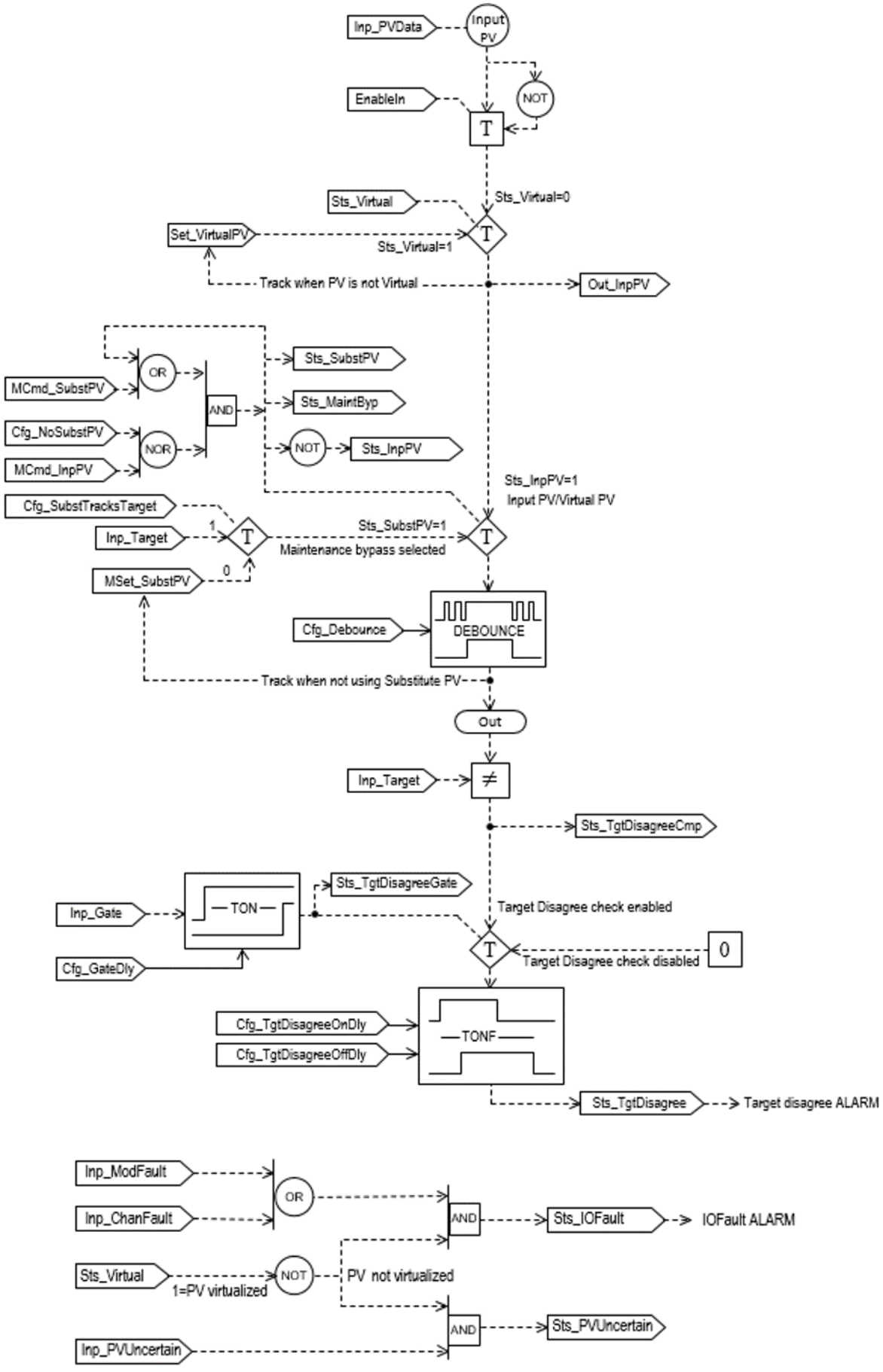
Tip: In Ladder Diagram if the rung-condition-in is false, the instruction uses the inverse of the Inp_PVData signal for processing. This allows the input to be a condition on the rung with PDI rather than mapped into Inp_PVData. To use the rung-condition-in mapping method, set Inp_PVData to 1, its default value.

This instruction includes a substitute PV capability for a manually-entered state. This is useful when a sensor is out of order or for simulation and testing.

The Discrete Input instruction and its input, target and gate signals handle alarm conditions described by these use cases:

- Raises an alarm when the input is in a given alarm state for a configurable amount of time, such as a low level alarm from a float level switch.
- Raises an alarm when the input does not follow another given signal within a configurable amount of time, such as a flow switch which should indicate flow when an associated pump has been running for a period of time and which should indicate NO flow when the associated pump has been stopped for a period of time.
- Raises an alarm when the input is in a given alarm state for a configurable amount of time after enabled by a gating signal, such as a vibration switch on a motor, which should only alarm when the motor has been running long enough for startup vibration to have settled out.

This diagram illustrates the functionality of the PDI instruction:


Inp_PVData
Input PV
EnableIn
NOT
T
Sts_Virtual
Sts_Virtual=0
Set_VirtualPV
Sts_Virtual=1
Track when PV is not Virtual
Out_InpPV
OR
MCmd_SubstPV
Cfg_NoSubstPV
NOR
MCmd_InpPV
AND
Sts_SubstPV
Sts_MaintByp
NOT
Sts_InpPV
Sts_InpPV=1
Input PV/Virtual PV
Cfg_SubstTracksTarget
Inp_Target
1
Sts_SubstPV=1
Maintenance bypass selected
MSet_SubstPV
0
Cfg_Debounce
DEBOUNCE
Track when not using Substitute PV
Out
Inp_Target
≠
Sts_TgtDisagreeCmp
Inp_Gate
TON
Sts_TgtDisagreeGate
Cfg_GateDly
Target Disagree check enabled
Target Disagree check disabled
0
Cfg_TgtDisagreeOnDly
Cfg_TgtDisagreeOffDly
TONF
Sts_TgtDisagree
Target disagree ALARM
Inp_ModFault
Inp_ChanFault
OR
AND
Sts_IOFault
IOFault ALARM
Sts_Virtual
1=PV virtualized
NOT
PV not virtualized
AND
Sts_PVUncertain
Inp_PVUncertain

Virtualization

Virtualization in PDI provides a virtual 0-state or 1-state input (Set_VirtualPV) that processes like an input. Use virtualization for instruction testing and operator training. Use PCmd_Virtual or MCmd_Virtual to enable virtualization. After finishing virtualization, use PCmd_Physical or MCmd_Physical to return to normal (physical device) operation.

Initialization

The instruction is normally initialized in the instruction first run. Re-initialization can be requested any time by setting Inp_InitializeReq = 1. For proper initialization, when adding the instruction while performing an online edit of the code, make sure that Inp_InitializeReq = 1, the default value.

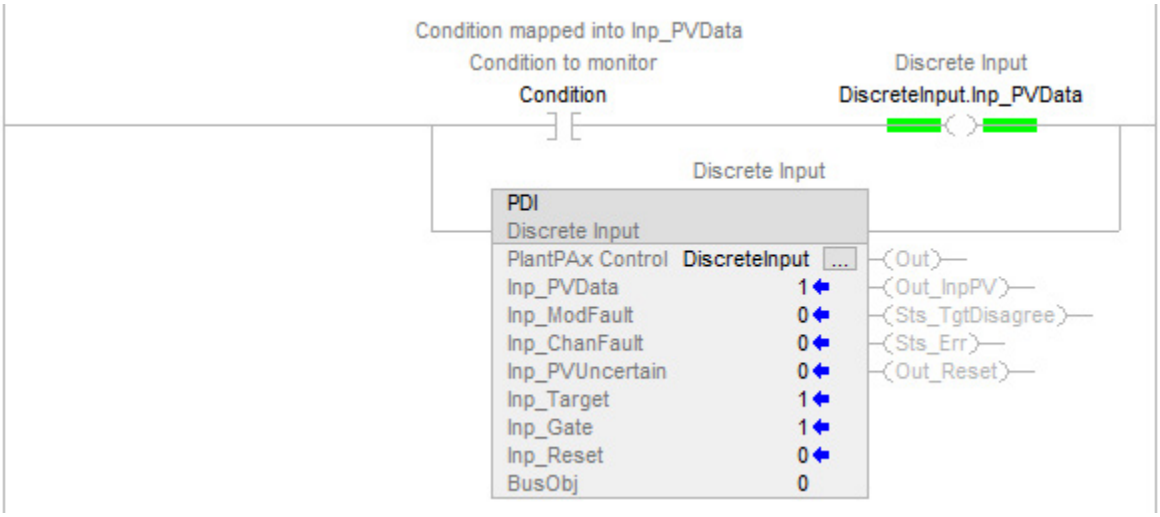
Configuration of Strings for HMI

Configure strings for HMI faceplates, as seen in FactoryTalk View, and for the Logix Designer configuration dialog box. The strings are set to extended properties of tag items. Configure the strings in the Logix Designer application only.

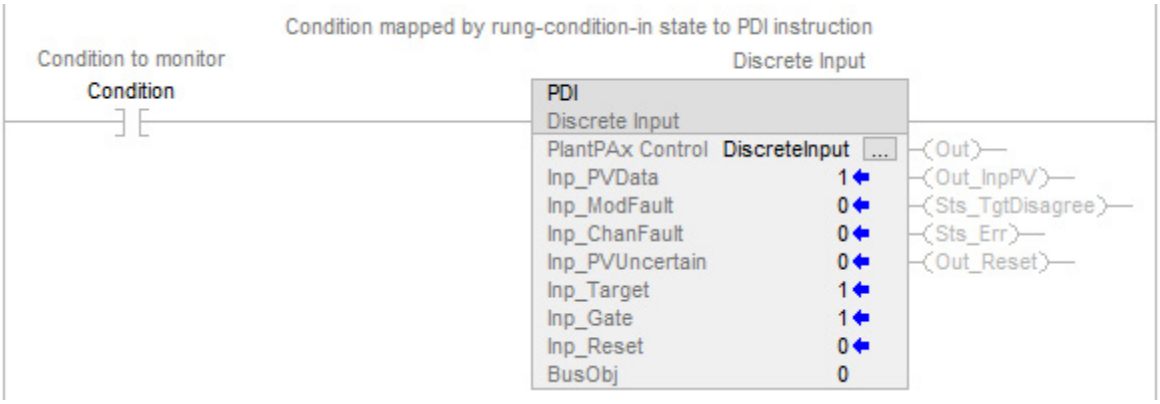
- Description
- State name strings for 0-state and 1-state
- Label for graphic symbol
- Display Library for HMI Faceplate call-up
- Instruction name
- Area name
- URL link
- More Information

Implementation

This illustration shows normal implementation with the input condition mapped to Inp_PVData on a separate branch.



This illustration shows the implementation with the input condition mapped to the PDI instruction using the rung-condition-in.



Monitor the PDI Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Rung-condition-out is cleared to false. The state of using raw input or maintenance substitute PV is not modified and persists through a controller powerup or PROG-to-RUN transition. The state of the physical/virtual selection persists through a control power or PROG-to-Run transition.
Instruction first run	All commands that are automatically cleared each execution are cleared and ignored. The instruction executes normally.
Rung-condition-in is false	Rung-condition-out is cleared to false. The instruction executes normally, except it uses the inverse of the Inp_PVData signal for processing.
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	EnableOut is cleared to false. The state of using raw input or maintenance substitute PV is not modified and persists through a controller powerup or PROG-to-RUN transition. The state of the physical/virtual selection persists through a control power or PROG-to-Run transition.
Instruction first run	All commands that are automatically cleared on each execution are cleared and ignored. The instruction executes normally.
Instruction first scan	See Instruction first run in the Function Block Diagram table.
EnableIn is false	EnableOut is cleared to false. The instruction executes normally, except it uses the inverse of the Inp_PVData signal for processing.
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

In Structured Text, EnableIn is always true during normal scan. The instruction executes when it is in the control path activated by the logic.

Condition/State	Action Taken
Prescan	See Prescan in the Function Block Diagram table.
Instruction first run	See Instruction first run in the Function Block Diagram table.
EnableIn is true	See EnableIn is true in the Function Block Diagram table.

Condition/State	Action Taken
Postscan	See Postscan in the Function Block Diagram table.

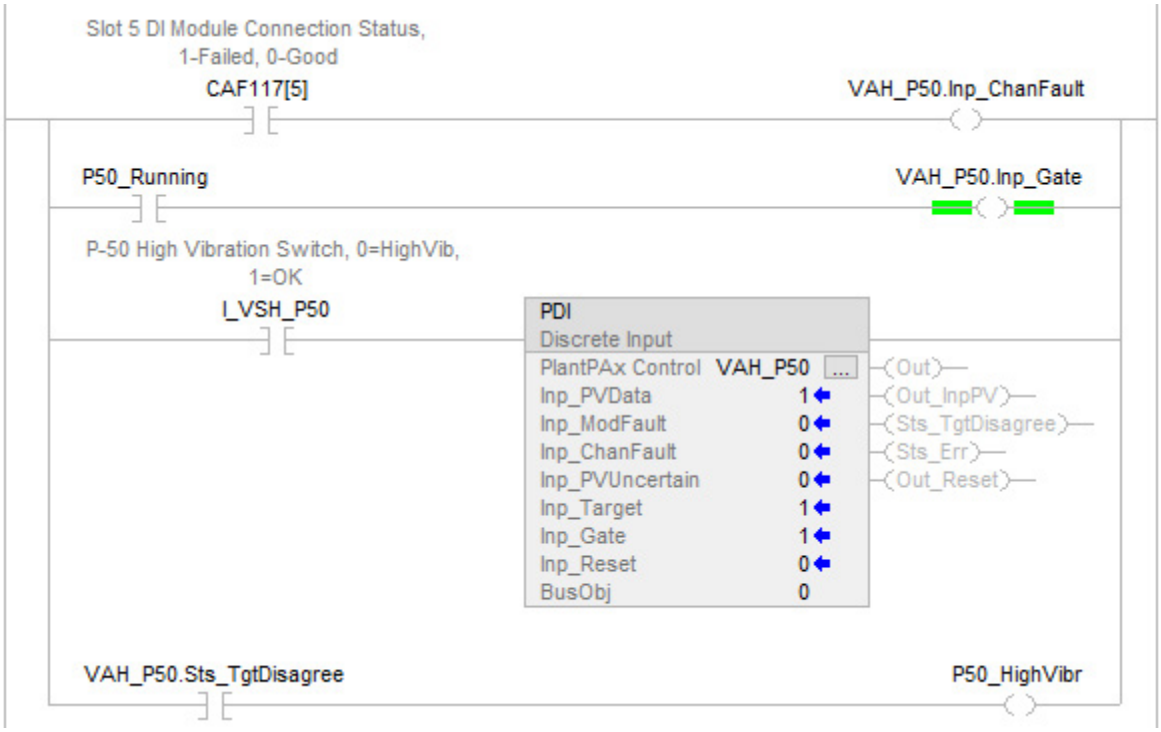
Example

In this example, tag I_VSH_P50 is the digital process value monitored by the PDI instruction. This tag provides a Boolean indication of High Vibration. The bad quality indication for the value of the process variable (Inp_ChanFault) comes from the connection status indication on the input module.

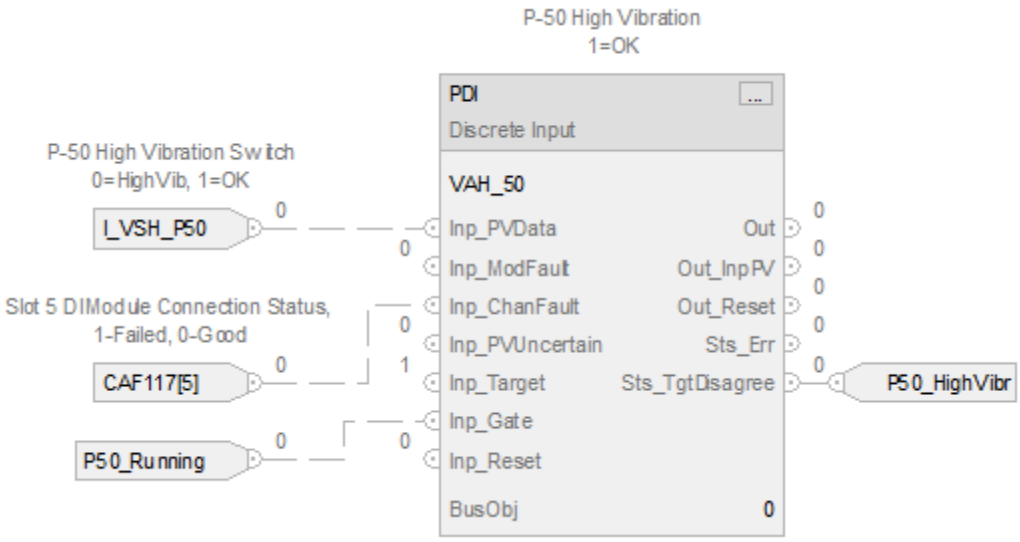
Inp_Target is defaulted to 1 indicating that the normal condition for I_VSH_P50 is also 1, and tag comments confirm that 1=OK for this process value. Inp_Gate is connected to the Motor Running status tag (P50_Running) that comes from the Sts_Running output of the P_Motor instruction instance for this motor (P50_Motor). The gate delay is configured to give the motor sufficient time after starting to settle into full normal speed run before enabling the high vibration indication (Sts_TgtDisagree) and alarm. The tag-based alarm for Target Disagree status (Sts_TgtDisagree) applies On Delay timing so the alarm will not raise until after delay time has expired.

Finally, P50_HighVibr is the output tag that indicates the status of I_VSH_P50 with appropriate gate delays based on whether the motor is running.

Ladder Diagram



Function Block Diagram



Structured Text

```
VAH_50.Inp_PV_Data := I_VSH_P50;

VAH_50.Inp_ChanFault := CAF_117[5];

VAH_50.Inp_Gate := P50_Running;

PDI(VAH_50);

P50_HighVibr := VAH_P50.Sts_TgtDisagree;
```

See also

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Process Discrete Output (PDO)

This information applies to the ControlLogix 5380P and 5580P controllers.

The Process Discrete Output (PDO) instruction drives a discrete (true / false) output, monitors discrete inputs serving as feedbacks from a device driven by the discrete output, and checks for alarm conditions. Use the PDO instruction for a channel of a discrete output module. Use the PDO instruction with any discrete (BOOL) signal.

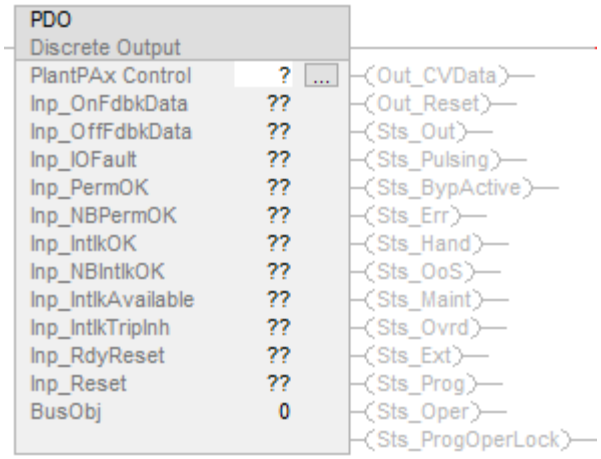
The PDO instruction:

- Controls one discrete output, with configurable text labels for the On and Off states of the output.

- Provides Operator, Program and External commands to set the output state to On or Off, to pulse the output On once, to pulse the output Off once, or to set the output to a continuous pulsing operation. Pulse times (on-time and off-time) are configurable.
- Monitors two discrete feedback inputs, monitoring the actual position of the device.
- Detects failure to reach the target state, after a configurable time, and alarms the failure when the feedback inputs are used. Optionally sheds to the de-energized state on a feedback failure.
- Monitors Permissive conditions that enable commanding the device to the On state.
- Monitors Interlock conditions that return the device to its de-energized Off state.
- Provides virtualization of a normally working device, while holding the output to the real device de-energized, for use in testing or operator training.
- Monitors I/O status and alarms on an I/O fault. Optionally sheds to the de-energized state on an I/O fault condition.
- Operates in Operator, Program, External, Override, Maintenance, Out of Service and Hand command sources.
- Provides an Available status, when in Program command source and operating normally, for use by higher-level automation logic to determine if the logic is able to manipulate the discrete output.

Available Languages

Ladder Diagram



Function Block Diagram

Structured Text

PDO(PDO tag, BusObj);

Operands

Important:	Unexpected operation may occur if: • Output tag operands are overwritten. • Members of a structure operand are overwritten. • Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions.

Configuration Operands

Operand	Type	Format	Description
PlantPAx Control	P_DISCRETE_OUTPUT	tag	Data structure required for proper operation of instruction.
BusObj	BUS_OBJ	tag	Bus component. May be null.

P_DISCRETE_OUTPUT Structure

Public members are standard, visible tag members that are programmatically accessible. Private, or hidden, members are used in HMI faceplates and are not programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	Description
EnableIn	BOOL	Enable input. Ladder Diagram: Corresponds to the rung condition. Default is true.
Inp_InitializeReq	BOOL	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request to re-initialize. The instruction clears this operand automatically. Default is true.
Inp_OwnerCmd	DINT	Owner device command. 0 = None, Inp_OwnerCmd.10 = Operator Lock, Inp_OwnerCmd.11 = Operator Unlock, Inp_OwnerCmd.12 = Program Lock, Inp_OwnerCmd.13 = Program Unlock, Inp_OwnerCmd.14 = Acquire Maintenance, Inp_OwnerCmd.15 = Release Maintenance, Inp_OwnerCmd.16 = Acquire External, Inp_OwnerCmd.17 = Release External, Inp_OwnerCmd.29 = Echo. Default is 0.
Inp_OnFdbkData	BOOL	On feedback from device. 1 = Device confirmed On. Default is false.
Inp_OffFdbkData	BOOL	Off feedback from device. 1 = Device confirmed Off. Default is false.
Inp_IOFault	BOOL	Indicates the IO data is inaccurate. 0 = The IO data is good, 1 = The IO data is bad, causing fault. This input sets Sts_IOFault, if the device is not virtual, which raises IOFault Alarm. Default is false.
Inp_PermOK	BOOL	1 = On permissives OK, device can turn On. Default is true.
Inp_NBPermOK	BOOL	1 = Non-bypassable On permissives OK, device can turn On. Default is true.
Inp_IntlkOK	BOOL	1 = Interlocks OK, device can turn On and stay On. Default is true.
Inp_NBIntlkOK	BOOL	1 = Non-bypassable interlocks OK, device can turn On and stay On. Default is true.
Inp_IntlkAvailable	BOOL	1 = Interlock Availability OK. Default is false.
Inp_IntlkTripInh	BOOL	1 = Inhibit Interlock Trip Status Default is false.
Inp_RdyReset	BOOL	1 = Related object, reset by this object, is ready to be reset. Default is false.

Public Input Members	Data Type	Description
Inp_Hand	BOOL	1 = Acquire Hand (typically permanently set to local), 0 = Release Hand. Default is false.
Inp_Ovrd	BOOL	1 = Acquire Override (higher priority program logic), 0 = Release Override Default is false.
Inp_OvrdCmd	SINT	Override device command: 0 = None, 1 = Off, 2 = On, 3 = Pulse off, 4 = Pulse on, 5 = Pulse continuously. Default is 0.
Inp_ExtInh	BOOL	1 = Inhibit External acquisition, 0 = Allow External acquisition. Default is false.
Inp_HornInh	BOOL	1 = Inhibit audible alert, 0 = Allow audible alert. Default is false.
Inp_Reset	BOOL	1 = Reset shed latches and cleared alarms. Default is false.
Cfg_HornOnChange	BOOL	0 = Horn on energize only. 1 = Horn on any state change. Default is false.
Cfg_ExtOffPrio	BOOL	1 = XCmd_Off any time, 0 = XCmd_Off only when External selected. Default is false.
Cfg_XCmdResets	BOOL	1 = New device XCmd resets shed latches and cleared alarms, 0 = XCmdReset required. Default is false.
Cfg_AllowDisable	BOOL	1 = Allow Maintenance to disable alarms. Default is true.
Cfg_AllowShelve	BOOL	1 = Allow Operator to shelve alarms. Default is true.
Cfg_HasPulse	BOOL	1 = Enable pulsing functions, 0 = On/Off only. Default is false.
Cfg_CompletePulse	BOOL	1 = Finish pulse in progress when commanded On or Off, 0 = Switch immediately to On or Off state when commanded. Default is false.
Cfg_FdbkFail	BOOL	1 = Both feedbacks On are invalid, 0 = Both feedbacks Off are invalid. Default is false.
Cfg_HasOnFdbk	BOOL	1 = Device provides an On feedback signal. Default is false.
Cfg_HasOffFdbk	BOOL	1 = Device provides an Off feedback signal. Default is false.
Cfg_UseOnFdbk	BOOL	1 = Use Device On feedback for failure checking. Default is false.
Cfg_UseOffFdbk	BOOL	1 = Use Device Off feedback for failure checking. Default is false.

Public Input Members	Data Type	Description
Cfg_OperOffPrio	BOOL	1 = OCmd_Off has priority, accepted any time, 0 = OCmd_Off only in Operator and Maintenance command sources. Default is false.
Cfg_OCmdResets	BOOL	1 = New Operator state command resets fault, 0 = Reset required to clear fault. Default is false.
Cfg_ShedOnIOFault	BOOL	1 = Go to Off state and alarm on IO fault, 0 = Alarm only on IO fault. **Important**: If a condition is configured to shed the device to the Off state on a fault, a reset is required to clear the shed fault to command the device to a state other than Off. Default is true.
Cfg_ShedOnFail	BOOL	1 = Go to Off state and alarm on Fail to reach position, 0 = Alarm only on Fail. **Important**: If a condition is configured to shed the device to the Off state on a fault, a reset is required to clear the shed fault to command the device to a state other than Off. Default is true.
Cfg_HasPermObj	BOOL	1 = Tells HMI a permissive object (for example, P_Perm) is used for Inp_PermOK and navigation to the permissive object's faceplate is enabled. **Important**: The name of the Permissive object in the controller must be this instruction's name with the suffix _Perm. For example, if the PDO instruction has the name PDOut123, then its Permissive object must be named PDOut123_Perm. Default is false.
Cfg_HasIntlkObj	BOOL	1 = Tells HMI an interlock object (for example, P_Intlk) is used for Inp_IntlkOK and navigation to the interlock object's faceplate is enabled. **Important**: The name of the interlock object in the controller must be this PDO object's name with the suffix _Intlk. For example, if the PDO instruction has the name PDOut123, then its interlock object must be named PDOut123_Intlk. Default is false.
Cfg_HasMoreObj	BOOL	1 = Tells HMI an object with more info is available. Default is false.
Cfg_HasOper	BOOL	1 = Operator (unlocked) exists, can be selected. Default is true.
Cfg_HasOperLocked	BOOL	1 = Operator Locked exists, can be selected. Default is true.
Cfg_HasProg	BOOL	1 = Program (unlocked) exists, can be selected. Default is true.
Cfg_HasProgLocked	BOOL	1 = Program Locked exists, can be selected. Default is true.
Cfg_HasExt	BOOL	1 = External exists, can be selected. Default is false.
Cfg_HasMaint	BOOL	1 = Maintenance exists, can be selected. Default is true.
Cfg_HasMaintOoS	BOOL	1 = Maintenance Out of Service exists, can be selected. Default is true.
Cfg_OvrdOverLock	BOOL	1 = Override supersedes Program/Operator Lock, 0 = Do not override Lock. Default is true.
Cfg_ExtOverLock	BOOL	1 = External supersedes Program/Operator Lock, 0 = Do not override Lock. Default is false.

Public Input Members	Data Type	Description
Cfg_ProgPwrUp	BOOL	1 = Power Up to Program, 0 = Power Up to Operator. Default is false.
Cfg_ProgNormal	BOOL	Normal Source: 1= Program if no requests; 0 = Operator if no requests. Default is false.
Cfg_PCmdPriority	BOOL	Command priority. 1 = Program commands win, 0 = Operator commands win. Default is false.
Cfg_PCmdProgAsLevel	BOOL	1 = PCmd_Prog used as a Level. Default is false.
Cfg_PCmdLockAsLevel	BOOL	1 = PCmd_Lock used as a Level (1 = Lock, 0 = Unlock). Default is false.
Cfg_ExtAcqAsLevel	BOOL	1 = XCmd_Acq used as Level (1 = Acquire, 0 = Release). Default is false.
Cfg_OvrdPermIntlk	BOOL	1 = Override ignores bypassable permissives/interlocks, 0 = Always use permissives/interlocks. Default is false.
Cfg_OnDly	REAL	Delay before initially turning output On (seconds). Valid = 0.0 to 2147483.0 seconds. Default is 0.
Cfg_OffDly	REAL	Delay before initially turning output Off (seconds). Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_OnPulseTime	REAL	Output On time for pulse On or pulse continuous (seconds). Valid = 0.0 to 2147483.0 seconds. Default is 0.5.
Cfg_OffPulseTime	REAL	Output Off time for pulse Off or pulse continuous (seconds). Valid = 0.0 to 2147483.0 seconds. Default is 0.5.
Cfg_OnFailTime	REAL	Time after output On to get On feedback before fault (seconds). Valid = 0.0 to 2147483.0 seconds. Default is 10.0.
Cfg_OffFailTime	REAL	Time after output Off to get Off feedback before fault (seconds). Valid = 0.0 to 2147483.0 seconds. Default is 10.0
Cfg_StartHornTime	REAL	Time in seconds to sound audible on commanded energize. Valid = 0.0 to 1000.0 seconds, 0.0 = disabled. Default is 0.0.
Cfg_VirtualFdbkTime	REAL	Delay to echo back of On/Off status when the device is treated as virtual (seconds). Valid = 0.0 to 2147483.0 seconds. Default is 2.0.
Cfg_CnfrmReqd	SINT	Operator command confirmation required. Represents the type of command confirmation required. 0 = None, 1 = Command confirmation required, 2 = Performer e-signature required, 3 = Performer and approver e-signature required. Default is 0.
PSet_Owner	DINT	Program owner request ID (non-zero) or release (zero). Default is 0.

Public Input Members	Data Type	Description
PCmd_Virtual	BOOL	Program command to select Virtual (simulated) device operation. The instruction clears this operand automatically. Default is false.
PCmd_Physical	BOOL	Program command to select Physical device operation (not simulated). The instruction clears this operand automatically. Default is false.
PCmd_On	BOOL	Program command to turn device On. The instruction clears this operand automatically. Default is false.
PCmd_Off	BOOL	Program command to turn device Off. The instruction clears this operand automatically. Default is false.
PCmd_OnPulse	BOOL	Program command to pulse device (which is Off) On once. The instruction clears this operand automatically. Default is false.
PCmd_OffPulse	BOOL	Program command to pulse device (which is On) Off once. The instruction clears this operand automatically. Default is false.
PCmd_ContPulse	BOOL	Program command to pulse device continuously (blink). The instruction clears this operand automatically. Default is false.
PCmd_Oper	BOOL	Program command to select Operator (Program to Operator). The instruction clears this operand automatically. Default is false.
PCmd_Prog	BOOL	Program command to select Program (Operator to Program). The instruction clears this operand automatically. Default is false.
PCmd_Lock	BOOL	Program command to lock Program (disallow Operator). The instruction clears this operand automatically if Cfg_PCmdLockAsLevel = 0. Default is false.
PCmd_Unlock	BOOL	Program command to unlock Program (allow Operator to acquire). The instruction clears this operand automatically. Default is false.
PCmd_Normal	BOOL	Program command to select Normal (Operator or Program). The instruction clears this operand automatically. Default is false.
PCmd_Reset	BOOL	Program command to reset all alarms and latched shed conditions requiring reset. The instruction clears this operand automatically. Default is false.
XCmd_Acq	BOOL	External command to acquire ownership (Operator/Program/Override/Maintenance to External). The instruction clears this operand automatically if Cfg_ExtAcqAsLevel = 0. Default is false.
XCmd_Rel	BOOL	External command to release ownership if Cfg_ExtAcqAsLevel = 0 (External to Operator/Program/Override/Maintenance). The instruction clears this operand automatically. Default is false.
XCmd_On	BOOL	External command to turn device On. The instruction clears this operand automatically.
XCmd_Off	BOOL	External command to turn device Off. The instruction clears this operand automatically. Default is false.
XCmd_OnPulse	BOOL	External command to pulse device (which is Off) On once. The instruction clears this operand automatically. Default is false.
XCmd_OffPulse	BOOL	External command to pulse device (which is On) Off once. The instruction clears this operand automatically. Default is false.

Public Input Members	Data Type	Description
XCmd_ContPulse	BOOL	External command to pulse device continuously (blink). The instruction clears this operand automatically. Default is false.
XCmd_Reset	BOOL	External command to reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.
XCmd_ResetAckAll	BOOL	External command to acknowledge and reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.

Public Output Members	Data Type	Description
EnableOut	BOOL	Enable output. This output state always reflects EnableIn input state.
Out_CVData	BOOL	Primary output. 1 = On, 0 = Off.
Out_HornData	BOOL	1 = Sound audible prior to commanded state change.
Out_Reset	BOOL	1 = Reset command has been received and accepted.
Out_OwnerSts	DINT	Status of command source, owner command handshake and ready status. 0 = None, Out_OwnerSts.10 = Operator Lock, Out_OwnerSts.11 = Operator Unlock, Out_OwnerSts.12 = Program Lock, Out_OwnerSts.13 = Program Unlock, Out_OwnerSts.14 = Acquire Maintenance, Out_OwnerSts.15 = Release Maintenance, Out_OwnerSts.16 = Acquire External, Out_OwnerSts.17 = Release External, Out_OwnerSts.18 = Has Maintenance, Out_OwnerSts.19 = External Override Lock, Out_OwnerSts.20 = Has External. Out_OwnerSts.21 = Has Operator Out_OwnerSts.22 = Has Operator Locked Out_OwnerSts.23 = Has Program Out_OwnerSts.24 = Has Program Locked Out_OwnerSts.29 = Echo Out_OwnerSts.30 = Not Ready.
Sts_Initialized	BOOL	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.
Sts_Out	BOOL	1 = Output is On (energized), 0 = Output is Off (de-energized).
Sts_Pulsing	BOOL	1 = Output is in a pulsing sequence.
Sts_FdbkOff	BOOL	1 = Device feedback shows device in Off state.
Sts_FdbkOn	BOOL	1 = Device feedback shows device in On state.
Sts_FdbkFail	BOOL	1 = Feedbacks are in an Invalid state (not On, Off, or Transition).
Sts_Horn	BOOL	1 = Audible alert (horn) is active.
Sts_Virtual	BOOL	1 = The instruction treats the device as virtual. The instruction acts as normal but the output is kept de-energized (Out_CVData = 0). 0 = The instruction operates the device normally.

Public Output Members	Data Type	Description
SrcQ_IO	SINT	Source and quality of primary input or output enumerated value: 0 = Good, live, confirmed good, 1 = Good, live, assumed good, 2 = Good, no feedback, assumed good, 8 = Test, virtualized, 9 = Test, loopback, 10 = Test, manually entered, 16 = Uncertain, live, off-spec, 17 = Uncertain, substituted at device or bus, 18 = Uncertain, substituted at instruction, 19 = Uncertain, using last known good, 20 = Uncertain, using replacement value, 32 = Bad, signal failure, 33 = Bad, channel fault, 34 = Bad, module or communication fault, 35 = Bad, invalid configuration.
SrcQ	SINT	Source and quality of primary value or status enumerated value: 0 = Good, live, confirmed good, 1 = Good, live, assumed good, 2 = Good, no feedback, assumed good, 8 = Test, virtualized, 9 = Test, loopback, 10 = Test, manually entered, 16 = Uncertain, live, off-spec, 17 = Uncertain, substituted at device or bus, 18 = Uncertain, substituted at instruction, 19 = Uncertain, using last known good, 20 = Uncertain, using replacement value, 32 = Bad, signal failure, 33 = Bad, channel fault, 34 = Bad, module or communication fault, 35 = Bad, invalid configuration.
Sts_eCmd	SINT	Device command: 0 = None, 1 = Off, 2 = On, 3 = Pulse off, 4 = Pulse on, 5 = Pulse continuously.
Sts_eFdbk	SINT	Device feedback: 0 = Transition, 1 = Off, 2 = On, 3 = Invalid.

Public Output Members	Data Type	Description
Sts_eSts	SINT	Device status: 0 = Off, 1= On, 2 = Pulse off, 3 = Pulse on, 4 = Pulse continuously, 5 = Turning off, 6 = Turning on, 7 = Horn 8 = Out of Service.
Sts_eFault	SINT	Device fault status: 0 = None, 1 = Feedback fault, 2 = IO fault, 3 = Configuration error.
Sts_eState	SINT	Internal Logic State (for animating STD on faceplate).
Sts_eNotify	SINT	Alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyAll	SINT	All alarm status enumerated values including related objects: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyIOFault	SINT	IOFault alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.

Public Output Members	Data Type	Description
Sts_eNotifyOnFail	SINT	OnFail alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyOffFail	SINT	OffFail alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyIntlkTrip	SINT	IntlkTrip alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_UnackAlmCount	DINT	Count of unacknowledged alarms.
Sts_eSrc	INT	The current command source enumerated values: 0 = Logic not in use, 4 = Hand, 8 = Maintenance, 16 = Override, 32 = Program, 33 = Program locked, 34 = Program by default (Normal), 64 = Operator, 65 = Operator locked, 66 = Operator by default (Normal), 128 = Maintenance Out of Service, 129 = Programmed Out of Service (rung false), 256 = External.

Public Output Members	Data Type	Description
Sts_bSrc	INT	Active selection bitmap (for HMI totem pole with command source request selection): Sts_bSrc.0: Hand, Sts_bSrc.1: Programmed Out of Service (rung false), Sts_bSrc.2: Maintenance Out of Service, Sts_bSrc.3: Maintenance, Sts_bSrc.4: Override, Sts_bSrc.5: External, Sts_bSrc.6: Program locked, Sts_bSrc.7: Program, Sts_bSrc.8: Operator locked, Sts_bSrc.9: Operator.
Sts_Available	BOOL	1 = Discrete output available for control by automation (Program).
Sts_IntlkAvailable	BOOL	1 = Interlock availability OK. Device can be acquired by program and is available for control when interlocks are OK.
Sts_Bypass	BOOL	1 = Bypassable interlocks are bypassed.
Sts_BypActive	BOOL	1 = Interlock bypassing active (bypassed or maintenance).
Sts_MaintByp	BOOL	1 = Device has a maintenance bypass function active.
Sts_NotRdy	BOOL	1 = Device is not ready, for HMI use hidden detail bits (Sts_Nrdyxxx) for reason.
Sts_NrdyOoS	BOOL	1 = Device is not ready: Device disabled by Maintenance.
Sts_NrdyCfgErr	BOOL	1 = Device is not ready: Configuration error.
Sts_NrdyIntlk	BOOL	1 = Device is not ready: Interlock not OK.
Sts_NrdyPerm	BOOL	1 = Device is not ready: Permissive not OK.
Sts_NrdyPrioOff	BOOL	1 = Device is not ready: Operator or External priority Off command requires reset.
Sts_NrdyFail	BOOL	1 = Device is not ready: Device failure (Shed requires Reset).
Sts_NrdyIOFault	BOOL	1 = Device is not ready: IO Fault (Shed requires Reset).
Sts_Err	BOOL	1 = Error in configuration: See detail bits (Sts_Errxxx) for reason.
Sts_ErrOnDly	BOOL	1 = Error in configuration: Invalid OnDelay timer preset (use 0.0 to 2147483.0).
Sts_ErrOffDly	BOOL	1 = Error in configuration: Invalid OffDelay timer preset (use 0.0 to 2147483.0).
Sts_ErrOnPulseTime	BOOL	1 = Error in configuration: Invalid OnPulse timer preset (use 0.0 to 2147483.0).
Sts_ErrOffPulseTime	BOOL	1 = Error in configuration: Invalid OffPulse timer preset (use 0.0 to 2147483.0).
Sts_ErrOnFailTime	BOOL	1 = Error in configuration: Invalid OnFail timer preset (use 0.0 to 2147483.0).
Sts_ErrOffFailTime	BOOL	1 = Error in configuration: Invalid OffFail timer preset (use 0.0 to 2147483.0).
Sts_ErrStartHornTime	BOOL	1 = Error in configuration: Invalid start horn timer (use 0.0 to 1000.0).
Sts_ErrVirtualFdbkTime	BOOL	1 = Error in configuration: Invalid virtual feedback timer (use 0.0 to 2147483.0).
Sts_ErrAlm	BOOL	1 = Error in tag-based alarm settings.
Sts_Hand	BOOL	1 = Hand is selected (supersedes OoS, Maintenance, Override, External, Program, Operator).
Sts_OoS	BOOL	1 = Out of Service is selected (supersedes Maintenance, Override, External, Program, Operator).
Sts_Maint	BOOL	1 = Maintenance is selected (supersedes Override, External, Program, Operator).
Sts_Ovrd	BOOL	1 = Override is selected (supersedes External, Program, Operator).
Sts_Ext	BOOL	1 = External is selected (supersedes Program and Operator).
Sts_Prog	BOOL	1 = Program is selected.
Sts_ProgLocked	BOOL	1 = Program is selected and Locked.
Sts_Oper	BOOL	1 = Operator is selected.
Sts_OperLocked	BOOL	1 = Operator is selected and Locked.
Sts_ProgOperSel	BOOL	Program/Operator selection (latch) state: 1 = Program, 0 = Operator.
Sts_ProgOperLock	BOOL	Program/Operator lock (latch) state, 1 = Locked, 0 = Unlocked.
Sts_Normal	BOOL	1 = Selection equals the Normal (Program or Operator).
Sts_ExtReqInh	BOOL	1 = External request inhibited, cannot get to External from current state.
Sts_ProgReqInh	BOOL	1 = Program request inhibited, cannot get to Program from current state.
Sts_MAcqRcvd	BOOL	1 = Maintenance Acquire command received this scan.

Public Output Members	Data Type	Description
Sts_Alm	BOOL	1 = An alarm is active.
Sts_AlmInh	BOOL	1 = One or more alarms shelved, disabled or suppressed.
Sts_IOFault	BOOL	IO Fault status: 0 = OK, 1 = Bad. There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PDOTag.@Alarms.Alm_IOFault.AlarmElement
Sts_OnFail	BOOL	1 = Device failed to turn On. There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PDOTag.@Alarms.Alm_OnFail.AlarmElement
Sts_OffFail	BOOL	1 = Device failed to turn Off. There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PDOTag.@Alarms.Alm_OffFail.AlarmElement
Sts_IntlkTrip	BOOL	1 = Device turned Off by an interlock Not OK. There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PDOTag.@Alarms.Alm_IntlkTrip.AlarmElement
Sts_RdyAck	BOOL	1 = An alarm is ready to be acknowledged.
Sts_RdyReset	BOOL	1 = A latched alarm or shed condition is ready to be reset.
XRdy_Acq	BOOL	1 = Ready for XCmd_Acq, enable HMI button.
XRdy_Rel	BOOL	1 = Ready for XCmd_Rel, enable HMI button.
XRdy_On	BOOL	1 = Ready for XCmd_On, enable HMI button.
XRdy_Off	BOOL	1 = Ready for XCmd_Off, enable HMI button.
XRdy_OnPulse	BOOL	1 = Ready for XCmd_OnPulse, enable HMI button.
XRdy_OffPulse	BOOL	1 = Ready for XCmd_OffPulse, enable HMI button.
XRdy_ContPulse	BOOL	1 = Ready for XCmd_ContPulse, enable HMI button.
XRdy_Reset	BOOL	1 = Ready for XCmd_Reset, enable HMI button.
XRdy_ResetAckAll	BOOL	1 = Ready for XCmd_ResetAckAll, enable HMI button.
Val_Owner	DINT	Current object owner ID (0 = not owned).

Private Input Members	Data Type	Description
HMI_BusObjIndex	DINT	This object's index in the bus array, for use by HMI display. Default is 0.
MCmd_Bypass	BOOL	Maintenance command to bypass all bypassable interlocks and permissives. The instruction clears this operand automatically. Default is false.
MCmd_Check	BOOL	Maintenance command to check (not bypass) all interlocks. The instruction clears this operand automatically. Default is false.
MCmd_OoS	BOOL	Maintenance command to select Out of Service. The instruction clears this operand automatically. Default is false.
MCmd_IS	BOOL	Maintenance command to select In Service. The instruction clears this operand automatically. Default is false.

Private Input Members	Data Type	Description
MCmd_Acq	BOOL	Maintenance command to acquire ownership (Operator/Program/Override to Maintenance). The instruction clears this operand automatically. Default is false.
MCmd_Rel	BOOL	Maintenance command to release ownership (Maintenance to Operator/Program/External/Override). The instruction clears this operand automatically. Default is false.
MCmd_Physical	BOOL	Maintenance command to select Physical device operation (not simulated). The instruction clears this operand automatically. Default is false.
MCmd_Virtual	BOOL	Maintenance command to select Virtual (simulated) device operation. The instruction clears this operand automatically. Default is false.
OCmd_On	BOOL	Operator command to turn device On. The instruction clears this operand automatically. Default is false.
OCmd_Off	BOOL	Operator Command to turn device Off. The instruction clears this operand automatically. Default is false.
OCmd_OnPulse	BOOL	Operator command to pulse device that is Off, On once. The instruction clears this operand automatically. Default is false.
OCmd_OffPulse	BOOL	Operator command to pulse device that is On, Off once. The instruction clears this operand automatically. Default is false.
OCmd_ContPulse	BOOL	Operator command to pulse device continuously (blink). The instruction clears this operand automatically. Default is false.
OCmd_Oper	BOOL	Operator command to select Operator (Program to Operator). The instruction clears this operand automatically. Default is false.
OCmd_Prog	BOOL	Operator command to select Program (Operator to Program). The instruction clears this operand automatically. Default is false.
OCmd_Lock	BOOL	Operator command to lock Operator (disallow Program). The instruction clears this operand automatically. Default is false.
OCmd_Unlock	BOOL	Operator command to unlock / release (allow Program to acquire) ownership. The instruction clears this operand automatically. Default is false.
OCmd_Normal	BOOL	Operator command to select Normal (Operator or Program). The instruction clears this operand automatically. Default is false.
OCmd_Reset	BOOL	Operator command to reset all alarms requiring reset. The instruction clears this operand automatically. Default is false.
OCmd_ResetAckAll	BOOL	Operator command to reset all alarms and latched shed conditions. The use of OCmd_ResetAckAll is restricted to HMI. The instruction clears this operand automatically. Default is false.

Private Output Members	Data Type	Description
MRdy_Bypass	BOOL	1 = Ready for MCmd_Bypass, enable HMI button.
MRdy_Check	BOOL	1 = Ready for MCmd_Check, enable HMI button.
MRdy_OoS	BOOL	1 = Ready for MCmd_OoS, enable HMI button.
MRdy_IS	BOOL	1 = Ready for MCmd_IS, enable HMI button.

Private Output Members	Data Type	Description
MRdy_Acq	BOOL	1 = Ready for MCmd_Acq, enable HMI button.
MRdy_Rel	BOOL	1 = Ready for MCmd_Rel, enable HMI button.
MRdy_Physical	BOOL	1 = Ready for MCmd_Physical, enable HMI button.
MRdy_Virtual	BOOL	1 = Ready for MCmd_Virtual, enable HMI button.
ORdy_Oper	BOOL	1 = Ready for OCmd_Oper, enable HMI button.
ORdy_Prog	BOOL	1 = Ready for OCmd_Prog, enable HMI button.
ORdy_Lock	BOOL	1 = Ready for OCmd_Lock, enable HMI button.
ORdy_Unlock	BOOL	1 = Ready for OCmd_Unlock, enable HMI button.
ORdy_Normal	BOOL	1 = Ready for OCmd_Normal, enable HMI button.
ORdy_On	BOOL	1 = Ready for OCmd_On, enable HMI button.
ORdy_Off	BOOL	1 = Ready for OCmd_Off, enable HMI button.
ORdy_OnPulse	BOOL	1 = Ready for OCmd_OnPulse, enable HMI button.
ORdy_OffPulse	BOOL	1 = Ready for OCmd_OffPulse, enable HMI button.
ORdy_ContPulse	BOOL	1 = Ready for OCmd_ContPulse, enable HMI button.
ORdy_Reset	BOOL	1 = A latched alarm or shed condition is ready to be reset.
ORdy_ResetAckAll	BOOL	1 = A latched alarm or shed condition is ready to be reset or acknowledged.

Public InOut Members	Data Type	Description
BusObj	BUS_OBJ	Bus component

Alarms

Discrete tag-based alarms are defined for these members.

Member	Alarm Name	Description
Sts_IOFault	Alm_IOFault	IO Failure. Raised when the Inp_IOFault input is true. Use this input to indicate to the instruction that a connection with the module is in fault. This input also indicates if a module reports field power loss/no load/short circuit is occurring for its I/O. If the I/O Fault is configured as a shed fault, the device is commanded Off and cannot be commanded to another state until reset.
Sts_OnFail	Alm_OnFail	Device failed to turn on (On Feedback not confirmed within configured period of time). Raised when the device is commanded On, but the device feedback does not confirm that the device is actually On within the configured failure time (Cfg_OnFailTime). If the Failure is configured as a shed fault, the device is commanded Off and cannot be commanded On until reset.
Sts_OffFail	Alm_OffFail	Device failed to turn off (Off Feedback not confirmed within configured period of time). Raised when the device is commanded Off, but the device feedback does not confirm that the device is actually Off within the configured failure time (Cfg_OffFailTime).
Sts_IntlkTrip	Alm_IntlkTrip	Interlock Trip alarm. Raised when an interlock not-OK condition causes the device to transition from the On state or a pulsing state to the Off state. If interlocks are not bypassed, a bypassable interlock or a non-bypassable interlock not-OK condition initiates an interlock trip. If interlocks are bypassed, only a non-bypassable interlock not-OK condition initiates an interlock trip.

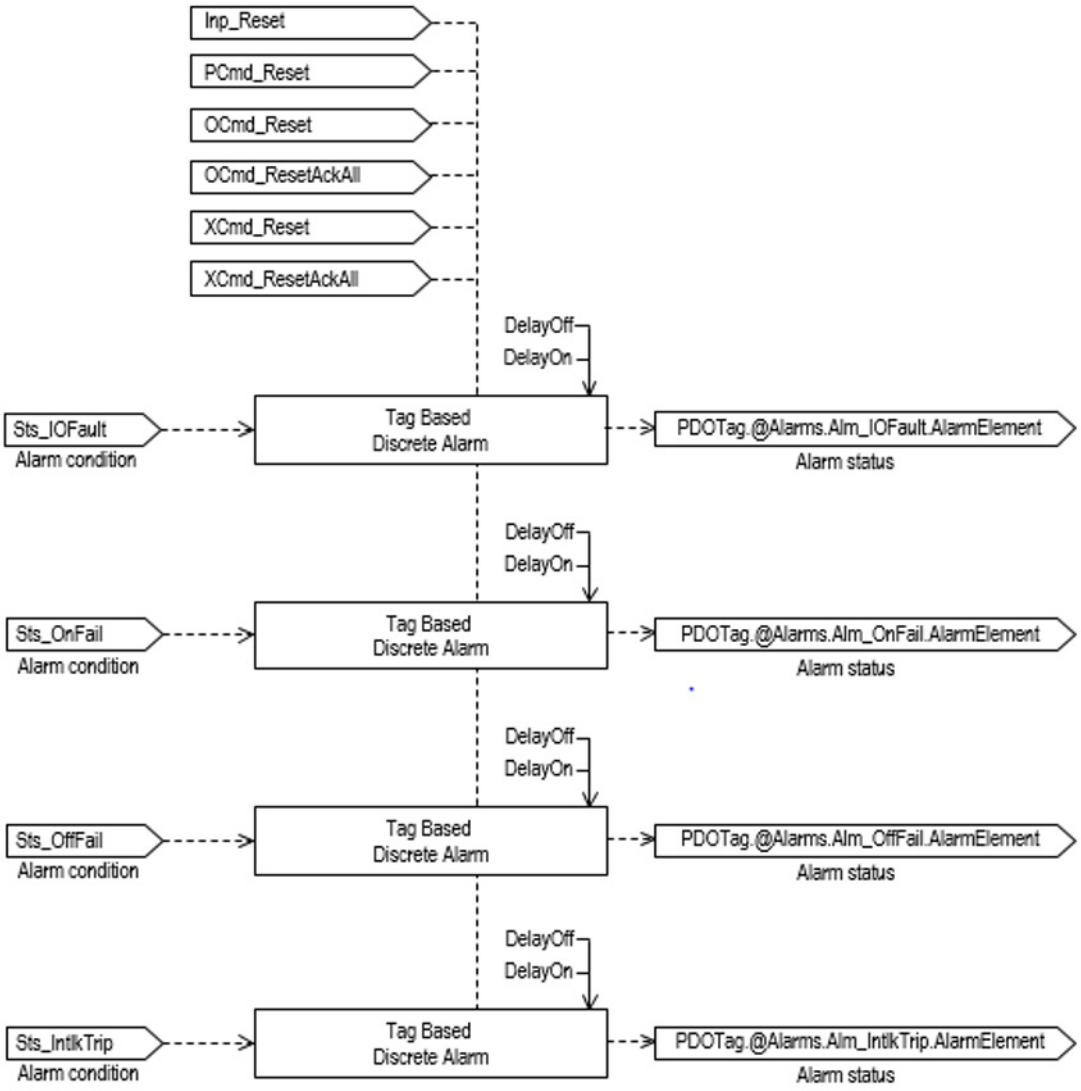
Mark the alarm as used or unused and set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format:

PDOTag.@Alarms.AlarmName.AlarmElement

The PDO instruction handles alarm conditions described by these four use cases and conditions:

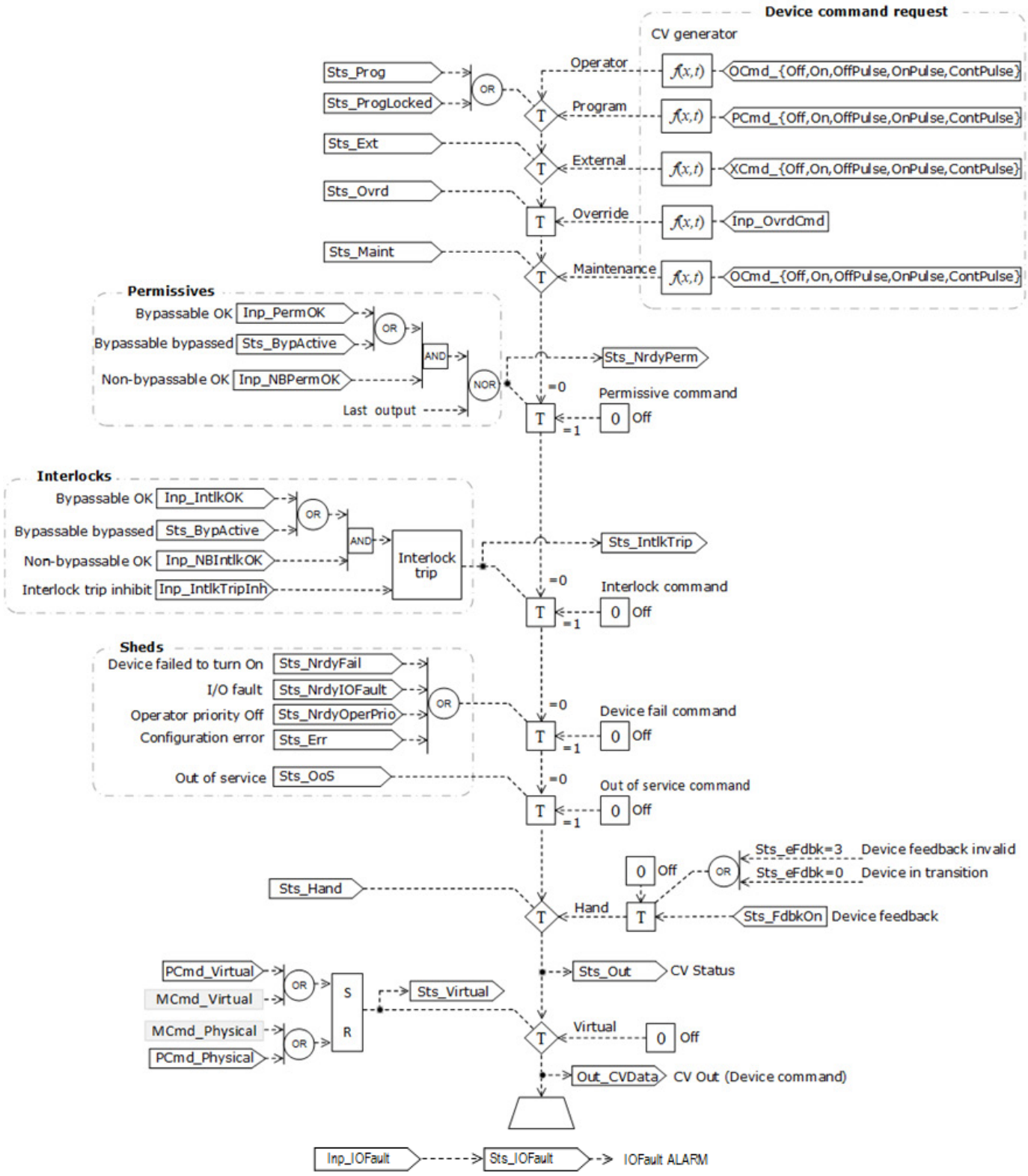
- I/O Fault Status - raised when the I/O Fault input is true. This input usually indicates to the instruction that I/O data is inaccurate and cannot be trusted for use in the application. If the I/O Fault is configured as a shed fault, the device is commanded Off and cannot be commanded to another state until reset.
- Interlock Trip Status - if interlocks are not bypassed, a bypassable interlock or a non-bypassable interlock not-OK condition initiates an interlock trip. If interlocks are bypassed, only a non-bypassable interlock 'not OK' condition initiates an interlock trip.
- Off Feedback Fail Status - raised when the device is commanded Off, but the device feedback does not confirm that the device is actually Off within the configured failure time.
- On Feedback Fail Status - raised when the device is commanded On, but the device feedback does not confirm that the device is actually On within the configured failure time. If the Failure is configured as a shed fault, the device is commanded Off and cannot be commanded On until reset.

There are Program, Operator, and External commands that enable the Reset and Reset & Acknowledge of all alarms of the instruction (Alarm Set) at the same time. This diagram shows how the commands interact with the PDO instruction.



Operation

The following diagram illustrates functionality of the PDO instruction:



Operator command request confirmation

The PDO instruction enables operator command requests OCmd_Off, OCmd_On, OCmd_OffPulse, OCmd_OnPulse and OCmd_ContPulse. Enforced security might require the request to be confirmed or canceled

before the selected command executes. The instruction checks the security rules inspecting Cfg_CnfrmReqd. If Cfg_CnfrmReqd=0 no confirmation is required and the request executes immediately. If Cfg_CnfrmReqd=1 the instruction waits for confirmation before executing. For Cfg_CnfrmReqd=2 or 3 eSignature is needed before the confirmation and cancellation is enabled.

Virtualization

Use virtualization for instruction testing and operator training. Set the Inp_Virtual operand to 1 to enable virtualization. After finishing virtualization, set the Inp_Virtual operand to 0 to return to normal operation.

When Virtualization is active, the output of the discrete output holds at 0, virtual feedback of a device is provided and I/O faults are ignored. Setting of Cfg_VirtualFdbkTime operand delays the echo of the On/Off status of the device. Manipulate the instruction to operate as if a working discrete output is present.

Initialization

The instruction is normally initialized in the instruction first run. Request re-initialization by setting Inp_InitializeReq = 1. For proper initialization, when adding the instruction while performing an online edit of the code, make sure that Inp_InitializeReq = 1, the default value.

Configuration of Strings for HMI

Configure strings for HMI faceplates, as seen in FactoryTalk View, and for the configuration dialog box. The strings are set to extended properties of tag items. Configure the strings in the Logix Designer application only.

- Description
- Label for graphic symbol
- Display Library for HMI Faceplate call-up
- Instruction name
- Area name
- URL link
- Path to an object with more information
- Target state 0
- Target state 1
- Transition state 0
- Transition state 1
- Command button off
- Command button on
- Command button pulse off

- Command button pulse on
- Command button pulse continuously
- IOFault alarm name
- IntlkTrip alarm name
- On Fail alarm name
- Off Fail alarm name

Command Source

The instruction uses these command sources. The command sources are prioritized in order from highest to lowest in this table.

Command Source	Description
Hand	Hardwired logic or other logic outside the instruction controls the device. The instruction tracks the state of the device for bumpless transfer back to one of the other command sources. (Highest priority command source)
Out-of-Service	The instruction is disabled and has no owner.
Maintenance	Maintenance controls the device and supersedes Operator, Program, External and Override control. Operator commands and settings from the HMI are accepted.
Override	Priority logic controls the device and supersedes Operator, Program and External control. Override Input (Inp_Ovrd) is accepted.
External	External logic (e.g. field pilot control or upstream SCADA) controls the device. External commands (XCmd_) are accepted.
Program locked	Program logic controls the device. Program commands (PCmd_) are accepted. Operator cannot take control from the Program. Override cannot take control from theProgram unless Cfg_OvrdOverLock = 1.
Program	Program logic controls the device. Program commands (PCmd_) are accepted.
Operator locked	The Operator controls the device. Operator commands (OCmd_) from the HMI are accepted. Program cannot take control from Operator. Override cannot take control from Operator unless Cfg_OvrdOverLock = 1.
Operator	The Operator controls the device. Operator commands (OCmd_) from the HMI are accepted. (Lowest priority command source)

The instruction enables or disables these operations:

- Maintenance Out-of-Service exists
- Maintenance exists
- External exists
- Program (unlocked) exists
- Program locked exists
- Operator (unlocked) exists
- Operator locked exists
- PCmd_Lock used as a Level (1 = Lock, 0 = Unlock)

The instruction checks for invalid configurations of control and forces the nearest valid configuration.

The core control model arbitrates the source of the commands and parameters that the receiving function accepts. The core control model determines if the source is:

- A programmatic entity which resides entirely within the processing environment, or

- An external interface entity which issues commands and parameters external and asynchronously to the processing environment.

Locking a control source prevents the other control source from acquiring privilege.

Core Command Source Model

The core control model consists of these control sources:

- Oper
- OperLocked
- Prog
- ProgLocked

The control model defaults to this configuration. Other control sources may be present in the model but act as overriding control sources, acting independent of the base Operator/Program state machine.

Enabling control sources as Configuration

The individual control sources may be enabled or disabled by the user. The default configuration uses the entire base model; upon power-up of the processing environment the control source will be the designated default. Some combinations of enabled control sources are disallowed as they are either unnecessary or could create unintended changes.

Prog Power Up

Configuration allows the user to specify whether Operator or Program will be the power-up default.

Prog Priority

Configuration allows the user to specify whether Operator or Program commands take priority when simultaneously asserted.

Automatic reset of commands

All commands are treated as one-shot-latched. This means that all commands are automatically cleared when the instruction executes and processes them.

Changing Destination States

Under certain configurations the destination command source for some commands may change. This is in keeping with the intent of the command. Example: If the Prog is disabled, the destination of the OCmd_Prog command is directed to the ProgLocked state instead of the Prog state. This maintains the intent of the OCmd_Prog command: the operator entity wishes to place the function in control of the program. If the command was eliminated then there would be no way to accomplish this. This is only done in configurations where it would cause no conflict or race condition, but serves to preserve as much user functionality as is practical.

Higher Priority Command Sources

These Higher priority command sources operate independently within the model:

- External
- Override
- Maintenance
- Out-of-Service
- In-Service
- Hand

Monitor the PDO Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Rung-condition-out is cleared to false. Sts_eSrc is set to 0. Sts_bSrc is set to 0.
Instruction first run	All commands that are automatically cleared on each execution are cleared and ignored. Inp_OvrdCmd is set to 0 (no command). The Program/Operator selection is set based on the configuration (Cfg_ProgPwrUp). The Program or Operator lock selection is set to unlocked. The Maintenance acquired/released state is not modified and persists through a controller powerup or PROG-to-RUN transition. PSet_Owner and Val_Owner are set to 0. If feedback is provided, Out_CVData and Sts_Out are set accordingly or cleared otherwise.
Rung-condition-in is false	Rung-condition-out is cleared to false. The instruction is put Out of Service if Inp_Hand=0. The output is de-energized and all alarm conditions are cleared. Latched alarms are reset. Command source selection processing proceeds except that Program and Operators commands are ignored and cleared and all ownership status bits (Sts_Maint, Sts_Ovrd, Sts_Ext, Sts_Prog and Sts_Oper) are cleared to 0. When rung-condition-in becomes true, the Program/Operator selection is set based on the configuration (Cfg_ProgPwrUp).
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	EnableOut is cleared to false. Sts_eSrc is set to 0. Sts_bSrc is set to 0.
Instruction first run	All commands that are automatically cleared on each execution are cleared and ignored. Inp_OvrdCmd is set to 0 (no command). The Program/Operator selection is set based on the configuration (Cfg_ProgPwrUp). The Program or Operator lock selection is set to unlocked. The Maintenance acquired/released state is not modified and persists through a controller powerup or PROG-to-RUN transition. PSet_Owner and Val_Owner are set to 0. If feedback is provided, Out_CVData and Sts_Out are set accordingly or cleared otherwise.
Instruction first scan	See Instruction first run in the Function Block Diagram table.

Condition/State	Action Taken
EnableIn is false	EnableOut is cleared to false. The instruction is put Out of Service if Inp_Hand=0. The output is de-energized and all alarm conditions are cleared. Latched alarms are reset. Command source selection processing proceeds except that Program and Operators commands are ignored and cleared and all ownership status bits (Sts_Maint, Sts_Ovrd, Sts_Ext, Sts_Prog and Sts_Oper) are cleared to 0. When rung-condition-in becomes true, the Program/Operator selection is set based on the configuration (Cfg_ProgPwrUp).
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

In Structured Text, EnableIn is always true during normal scan. The instruction executes when it is in the control path activated by the logic.

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Instruction first run	See Instruction first run in the Ladder Diagram table.
EnableIn is true	See EnableIn is true in the Function Block Diagram table.
Postscan	See Postscan in the Function Block Diagram table.

Example

Use the PDO instruction to control a heating jacket on a vent filter. The heating jacket is being used in this case to keep the vent filter dry when there is potential for condensate buildup. The vent filter heater jacket does not provide the feedback on its status. In normal operating conditions, the vent filter heater jacket is being commanded on or off by the control sequence configured in the controller. Always command the vent filter off on interlock associated with the status of the controlling sequence of the heating jacket.

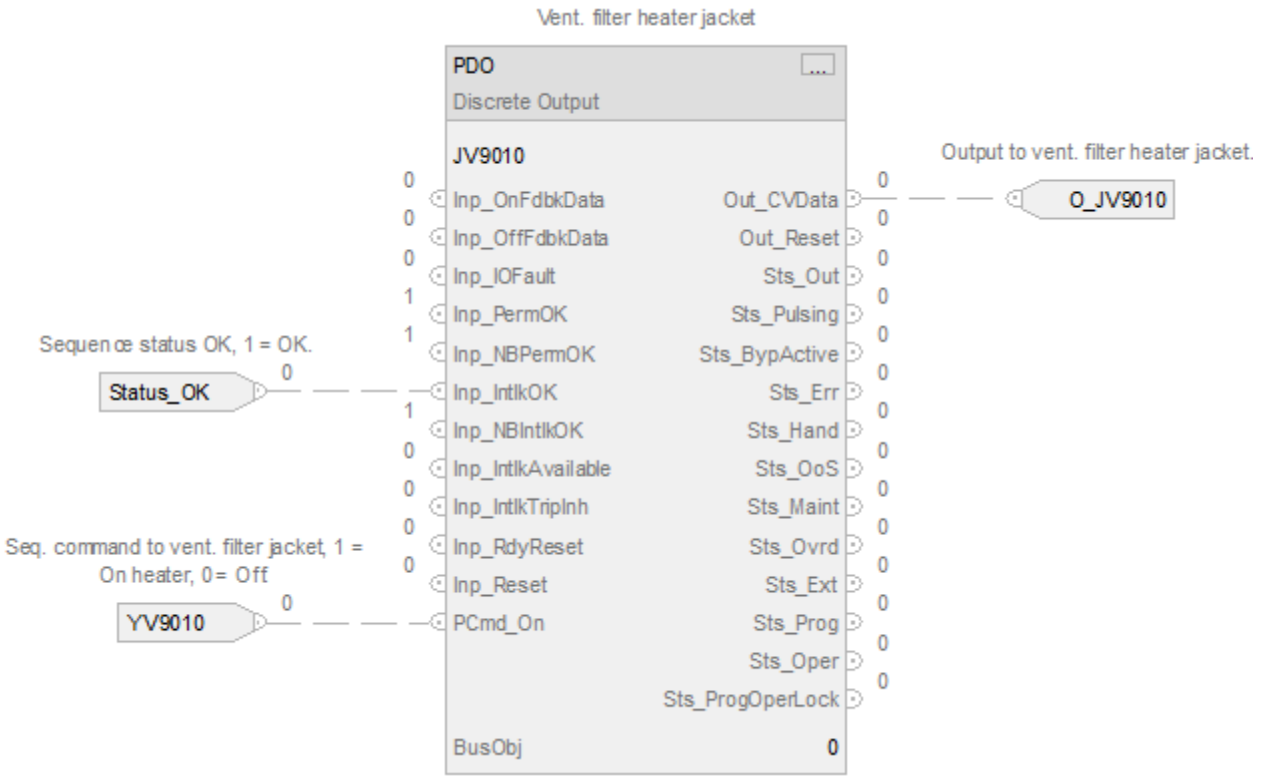
The controlling sequence issues a single bit for the desired state of the vent filter heater. The operand PCmd_On is connected to this bit to command the vent filter heater on and off. The operand Cfg_PCmdOnAsLevel is set to 1, indicating that the instruction acts upon PCmd_On based on value (level) instead of acting only on transition to true (edge) so that PCmd_On can be used to command both the On and Off states. The controlling sequence could be written to set the command bits PCmd_On and PCmd_Off directly (for example, by using structured text within an SFC), in which case Cfg_PCmdOnAsLevel could be left at its default of 0 to cause the instruction to clear the commands once they have been acted upon. The operand Cfg_ProgNormal is set to 1 to indicate that the normal operating state of the controller is Program, meaning it is normally commanded by the control

sequence. The status of the sequence is connected to the Inp_IntlkOK operand so that the output to the vent filter heater jacket is always off when the skid is not operating properly, even if the instruction is not in Program mode. The operands Cfg_HasOnFdbk and Cfg_HasOffFdbk are both set to 0 to indicate that the vent filter heater jacket does not provide feedback on its status.

The example is shown in all three languages.

Ladder Diagram

Function Block Diagram



Structured Text

```
JV9010.Inp_IntlkOK := Status_OK;

JV9010.PCmd_On := YV9010;

PDO(JV9010, 0);

O_JV9010 := JV9010.Out_CVData;
```

See also

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Process Dosing (PDOSE)

This information applies to the ControlLogix 5380P and 5580P controllers.

The Process Dosing (PDOSE) instruction controls an ingredient addition that uses a flow meter to measure the quantity of ingredient added. The flow meter can be an analog flow meter (signal proportional to flow), a pulse generating flow meter (pulse count proportional to quantity delivered), or a

digital flow meter providing flow rate or quantity (totalized flow) information. The instruction also controls an ingredient addition that uses a weigh scale to measure the quantity of ingredient added. The weigh scale can be on the receiving vessel, indicating gain in weight, or on the sourcing vessel, indicating loss in weight. The weigh scale can be connected using an analog input, device network, or other connection.

Use the PDOSE instruction to:

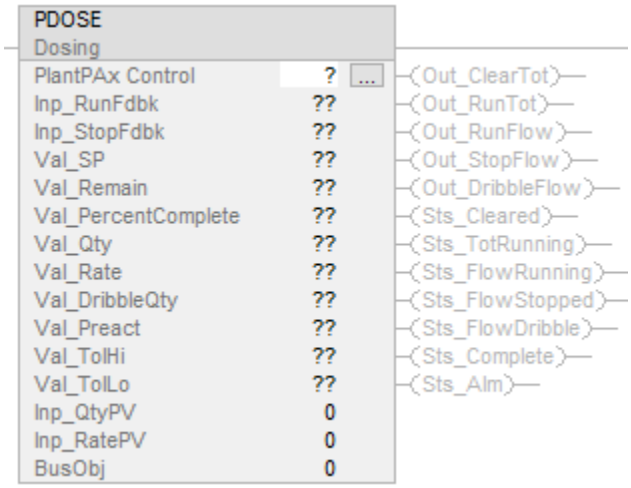
- Provide inputs for rate (flow rate or quantity per time) and quantity (total or pulse count).
- Use a pulse count as the Quantity process variable (PV), with configurable rollover count.
- Totalize the flow rate PV to determine the quantity delivered when the flowmeter provides a rate signal but no quantity.
- Calculate the flow rate given the quantity by differentiating with respect to time when the meter provides a total or pulse count but no rate. If the rate PV is calculated from an input quantity, the PDOSE instruction uses a first-order, lag filter on the calculated rate PV signal to reduce the impact of jitter, scan time, quantization error, or input signal noise.
- Provide a low rate cutoff function, used to ignore flow rate values near zero to deal with noise or zero calibration error in the rate signal.
- Use a flowmeter with built-in totalizer. Forwards the totalizer clear command to the flowmeter and checks that the flowmeter's total was reset.
- Provide outputs to control associated equipment, such as pumps and valves, to start and stop flow.
- Monitor the status of controlled equipment, such as pumps and valves.
- Monitor rate or quantity input communication status and provide indication of uncertain or bad rate PV or quantity PV.
- Provide program or operator entry of a quantity to deliver (setpoint) and calculate the quantity remaining to deliver and percent complete during delivery.
- Provide program or operator entry of high and low tolerance limits. Lets the program or operator initiate a tolerance check after delivery is complete. Provides a warning if under tolerance and lets the operator bump the flow to make up the shortage. The bump can be set up as a timed bump or as an operator jog-like function. Provides an alarm if over tolerance and inhibits further flow.
- Automatically switch to a lower dribble flow rate as the quantity delivered approaches setpoint. Provides operator or program entry of the dribble quantity. Provides run, dribble, and stop outputs to controlled equipment.
- Use a preact value to stop flow to account for material in the pipe, time for equipment to stop, and delays in measurement, scan, communication, and so forth. Provides operator or program entry of

the preact value. Provides an optional automatic preact correction based on the error in delivery when tolerance is checked. The auto correction lets the preact learn the correct value over time.

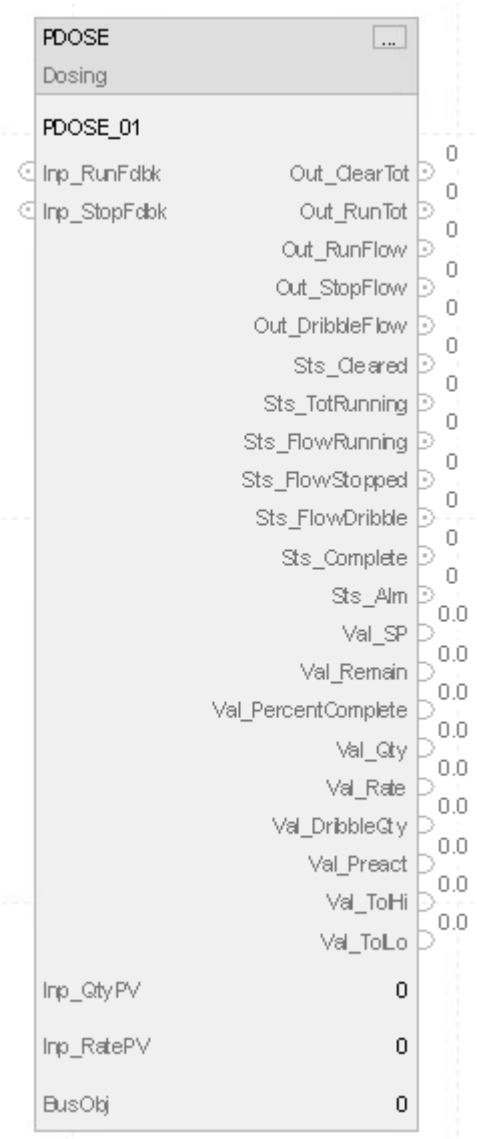
- Use the standard command source (PCMDSRC) instruction to provide ownership for entry of settings and acceptance of commands.
- Provide linear scaling of the input weight value from raw, input card units to engineering, display units.
- Provide a rate of weight change calculation (differentiation with respect to time) to generate an inferred flow rate. The calculated rate is filtered and has a low cutoff, so the rate is reported as zero when the change in weight is only from noise on the input weight signal.
- Provide outputs to control associated equipment, such as pumps and valves to start and stop flow. The operator or the program can start the ingredient addition, then pause and resume as needed.
- Monitor the status of controlled equipment, such as pumps and valves. Flow is stopped and an alarm is raised on an equipment fault or if the equipment fails to respond as commanded.
- Monitor the weight PV input quality and communication status and provides indication of uncertain or bad weight PV. Flow is stopped and an alarm is raised on a bad PV or communication loss.
- Provide a continuous monitoring function which allow continuous monitoring without SP requirement.

Available Languages

Ladder Diagram



Function Block Diagram



Structured Text

PDOSE (PDOSE tag, Inp_QtyPV, Inp_RatePV, BusObj);

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PlantPAx Control	P_DOSING	tag	Data structure required for proper operation of instruction.
Inp_QtyPV	REAL	tag	Quantity from weigh scale or flowmeter (EU or pulse count).
Inp_RatePV	REAL	tag	Flow rate from flowmeter (EU/Time, see Cfg_RateTime).
BusObj	BUS_OBJ	tag	Bus component.

P_DOSING Structure

Public members are standard, visible tag members that are programmatically accessible. Private, hidden members are used in HMI faceplates and are not programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
EnableIn	BOOL	Not Visible	Not Required	Input	Enable Input - System Defined Parameter Default is true.
Inp_InitializeReq	BOOL	Not Visible	Not Required	Input	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. use this request when reinitialization is needed. The instruction clears this operand automatically. Default is true.
Inp_OwnerCmd	DINT	Not Visible	Not Required	Input	Owner device command. 0 = None, Inp_OwnerCmd.10 = Operator lock, Inp_OwnerCmd.11 = Operator unlock, Inp_OwnerCmd.12 = Program lock, Inp_OwnerCmd.13 = Program unlock, Inp_OwnerCmd.14 = acquire Maintenance, Inp_OwnerCmd.15 = release Maintenance, Inp_OwnerCmd.16 = acquire External, Inp_OwnerCmd.17 = release External. Default is 0.
Inp_RatePVBad	BOOL	Not Visible	Not Required	Input	1 = Rate PV input quality = Bad (Fail). Default is false.
Inp_RatePVUncertain	BOOL	Not Visible	Not Required	Input	1 = Rate PV input quality = Uncertain. Default is false.
Inp_QtyPVBad	BOOL	Not Visible	Not Required	Input	1 = Quantity PV input quality = Bad (Fail). Default is false.
Inp_QtyPVUncertain	BOOL	Not Visible	Not Required	Input	1 = Quantity PV input quality = Uncertain. Default is false.
Inp_RunFdbk	BOOL	Visible	Not Required	Input	1 = Controlled equipment is delivering (running). Default is false.
Inp_DribbleFdbk	BOOL	Not Visible	Not Required	Input	1 = Controlled equipment is delivering at dribble. Default is false.
Inp_StopFdbk	BOOL	Visible	Not Required	Input	1 = Controlled equipment is confirmed stopped. Default is false.
Inp_CtrldEqpFault	BOOL	Not Visible	Not Required	Input	Controlled equipment object or I/O status 0 = Ok, 1 = Fail. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Inp_Reset	BOOL	Not Visible	Not Required	Input	1 = Reset shed latches and cleared alarms. Default is false.
Inp_eRatePVSrcQ	SINT	Not Visible	Not Required	Input	Flow rate signal source and quality (enumerated): 0=Good, live, confirmed good 1=Good, live, assumed good 2=Good, no feedback, assumed good 8=Test, virtualized 9=Test, loopback 10=Test, manually entered 16=Uncertain, live, off-spec 17=Uncertain, substituted at device or bus 18=Uncertain, substituted at instruction 19=Uncertain, using last known good 20=Uncertain, using replacement value 32=Bad, signal failure 33=Bad, channel fault 34=Bad, module or communication fault 35=Bad, invalid configuration. Default is 0.
Inp_eRatePVNotify					Rate PV object alarm priority and acknowledgement status. 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged. Default is 0.
Inp_eQtyPVSrcQ	SINT	Not Visible	Not Required	Input	Quantity signal source and quality (enumerated): 0=Good, live, confirmed good 1=Good, live, assumed good 2=Good, no feedback, assumed good 8=Test, virtualized 9=Test, loopback 10=Test, manually entered 16=Uncertain, live, off-spec 17=Uncertain, substituted at device or bus 18=Uncertain, substituted at instruction 19=Uncertain, using last known good 20=Uncertain, using replacement value 32=Bad, signal failure 33=Bad, channel fault 34=Bad, module or communication fault 35=Bad, invalid configuration. Default is 0.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Inp_eQtyPVNotify	SINT	Not Visible	Not Required	Input	Quantity PV object alarm priority and acknowledgement status. 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged. Default is 0.
Inp_Hand	BOOL	Not Visible	Not Required	Input	1 = Acquire hand (typically hardwired local), 0 = Release hand. Default is false.
Inp_Ovrd	BOOL	Not Visible	Not Required	Input	1 = Acquire Override (higher priority Program logic), 0 = Release Override. Default is false.
Inp_ExtInh	BOOL	Not Visible	Not Required	Input	1 = Inhibit External acquisition, 0 = Allow External acquisition. Default is false.
Cfg_AllowDisable	BOOL	Not Visible	Not Required	Input	1 = Allow Operator alarm disable, 0 = Disallow Operator alarm disable. Default is false.
Cfg_AllowShelve	BOOL	Not Visible	Not Required	Input	1 = Allow Operator alarm shelve, 0 = Disallow Operator alarm shelve. Default is false.
Cfg_HasEqpFdbk	BOOL	Not Visible	Not Required	Input	1 = Controlled equipment provides run (dribble if used) and stop feedback. Default is false.
Cfg_UseEqpFdbk	BOOL	Not Visible	Not Required	Input	1 = Use run / dribble / stop feedback, 0 = Assume equipment state. Default is false.
Cfg_HasDribble	BOOL	Not Visible	Not Required	Input	1 = Slow to dribble before complete, 0 = Run full flow until complete. Default is false.
Cfg_HasRatePVNav	BOOL	Not Visible	Not Required	Input	1 = Tells HMI an analog object (pai, etc.) is used for Inp_RatePV. Default is false.
Cfg_HasMonitoring	BOOL	Not Visible	Not Required	Input	1 = Allows continuous monitoring without setpoint requirement. Default is false.
Cfg_HasMoreObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI an object with more info is available. Default is false.
Cfg_HasQtyPVNav	BOOL	Not Visible	Not Required	Input	1 = Tells HMI an analog object (pai, etc.) is used for Inp_QtyPV. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_AutoAdjPreact	BOOL	Not Visible	Not Required	Input	1 = Enable automatic adjustment of preact after each delivery. Default is false.
Cfg_LossInQty	BOOL	Not Visible	Not Required	Input	1 = Flow reduces quantity (Transfer out), 0 = Flow increases quantity (Transfer in). Default is false.
Cfg_SetTrack	BOOL	Not Visible	Not Required	Input	1 = When the owner is Program the operator settings track the program settings. When the owner is Operator the program settings track the operator settings; and the virtual inputs match the output values (transitions are bumpless) 0 = No tracking. Default is true.
Cfg_ShedOnEqpFault	BOOL	Not Visible	Not Required	Input	1 = Stop delivery and alarm on equipment fault; 0 = Alarm only on equipment fault. Default is true.
Cfg_HasOper	BOOL	Not Visible	Not Required	Input	1 = Operator (unlocked) exists, can be selected. Default is true.
Cfg_HasOperLocked	BOOL	Not Visible	Not Required	Input	1 = Operator locked exists, can be selected. Default is true.
Cfg_HasProg	BOOL	Not Visible	Not Required	Input	1 = Program (unlocked) exists, can be selected. Default is true.
Cfg_HasProgLocked	BOOL	Not Visible	Not Required	Input	1 = Program locked exists, can be selected. Default is true.
Cfg_HasExt	BOOL	Not Visible	Not Required	Input	1 = External exists, can be selected. Default is false.
Cfg_HasMaint	BOOL	Not Visible	Not Required	Input	1 = Maintenance exists, can be selected. Default is true.
Cfg_OvrdOverLock	BOOL	Not Visible	Not Required	Input	1 = Override supersedes Program/Operator Lock, 0 = Don't Override lock. Default is true.
Cfg_ExtOverLock	BOOL	Not Visible	Not Required	Input	1 = External supersedes Program/Operator Lock, 0 = Don't Override lock. Default is false.
Cfg_ProgPwrUp	BOOL	Not Visible	Not Required	Input	1 = Power up to Program, 0 = Power up to Operator. Default is false.
Cfg_ProgNormal	BOOL	Not Visible	Not Required	Input	Normal source: 1 = Program if no requests; 0 = Operator if no requests. Default is false.
Cfg_PCmdPriority	BOOL	Not Visible	Not Required	Input	Command priority. 1 = Program commands win, 0 = Operator commands win. Default is false.
Cfg_PCmdProgAsLevel	BOOL	Not Visible	Not Required	Input	1 = PCmd_Prog used as a level. Default is false.
Cfg_PCmdLockAsLevel	BOOL	Not Visible	Not Required	Input	1 = PCmd_Lock used as a level (1 = Lock, 0 = Unlock). Default is false.
Cfg_ExtAcqAsLevel	BOOL	Not Visible	Not Required	Input	1 = XCmd_Acq used as level (1 = Acquire, 0 = Release). Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_AutoAdjPercent	REAL	Not Visible	Not Required	Input	Percentage of delivery error to auto-adjust preact. Valid = 0.0 to 100.0 (%) Default is 10.0.
Cfg_CountsPerEU	REAL	Not Visible	Not Required	Input	Number of counts in Inp_QtyPV which equal 1.0 Engineering Units. Valid = 0.0 to maximum positive float. Default is 1.0.
Cfg_EUQtyMult	REAL	Not Visible	Not Required	Input	Rate to quantity Engineering Units multiplier (e.g., gal to bbl.). Valid = 0.0 to maximum positive float. Default is 1.0.
Cfg_BumpTime	REAL	Not Visible	Not Required	Input	Bump (manual top-off) time. Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_ClearPulseTime	REAL	Not Visible	Not Required	Input	Time to pulse Out_ClearTot to clear External totalizer. Valid = 0.0 to 2147483.0 seconds. Default is 1.0.
Cfg_FaultTime	REAL	Not Visible	Not Required	Input	Time for equipment feedback to follow output before fault. Valid = 0.0 to 2147483.0 seconds. Default is 10.0.
Cfg_HiFlowRateLim	REAL	Not Visible	Not Required	Input	High flow rate alarm limit. Valid = 0.0 to maximum positive float. Default is 3.40E+38.
Cfg_LoFlowRateLim	REAL	Not Visible	Not Required	Input	Low flow rate alarm limit. Valid = 0.0 to maximum negative float. Default is -3.40E+38.
Cfg_LoRateCutoff	REAL	Not Visible	Not Required	Input	Rate below which to report zero flow (Inp_RatePV in engineering unit/time, see Cfg_RateTime). Valid = 0.0 to maximum positive float. Default is 0.0.
Cfg_MaxQty	REAL	Not Visible	Not Required	Input	Maximum allowed quantity to deliver (setpoint).Valid = 0.0 to maximum positive float. Default is 1.50E+38.
Cfg_RateFiltTimeConst	REAL	Not Visible	Not Required	Input	Filter time constant (sec) for calculated rate. Valid = 0.0 to 2147483.0 seconds. Default is 0.1.
Cfg_RateTime	REAL	Not Visible	Not Required	Input	Time factor for rate (60 for /min, 3600 for /hr) (sec). Valid = 0.0 to 2147483.0 seconds. Default is 1.0.
Cfg_Rollover	REAL	Not Visible	Not Required	Input	Quantity rollover (e.g., max count for pulse input). Valid = 0.0 to maximum positive float. Default is 0.0.
Cfg_SettleTime	REAL	Not Visible	Not Required	Input	Time to allow flow to stop before allowing tolerance check.Valid = 0.0 to 2147483.0 seconds. Default is 1.0.
Cfg_VirtualRate	REAL	Not Visible	Not Required	Input	Rate at which to deliver when running in virtual (Engineering Units/rate time). Valid = 0.0 to maximum positive float. Default is 1.0.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_VirtualDribbleRate	REAL	Not Visible	Not Required	Input	Rate at which to dribble when running in virtual (Engineering Units/rate time). Valid = 0.0 to maximum positive float. Default is 0.1.
Cfg_eKeepSP	SINT	Not Visible	Not Required	Input	Ownership of Setpoint (enumeration): 0 = follows CmdSrc, 1 = Operator, 2 = Program, 3 = External. Default is 0.
Cfg_eKeepStart	SINT	Not Visible	Not Required	Input	Ownership of Start commands (enumeration): 0 = follows CmdSrc, 1 = Operator, 2 = Program, 3 = External. Default is 0.
Cfg_eKeepTol	SINT	Not Visible	Not Required	Input	Ownership of Tolerance commands (enumeration): 0 = follows CmdSrc, 1 = Operator, 3 = External. Default is 0.
Cfg_eKeepDribblePreact	SINT	Not Visible	Not Required	Input	Ownership of Dribble\Preact (enumeration): 0 = follows CmdSrc, 1 = Operator, 2 = Program, 3 = External. Default is 0.
Cfg_HasHistTrend	SINT	Not Visible	Not Required	Input	Has Historical Trend. This enables navigation to the Device Historical Trend Faceplate from the HMI. Value of 0 = No External historical trend, 1 = Datalog historical trend, 2 = Historian histrical trend. Default is 0.
Cfg_QtyDecPlcs	SINT	Not Visible	Not Required	Input	Number of decimal places for quantity display. Valid = 0,1,2,3,4,5,6. Default is 2.
Cfg_RateDecPlcs	SINT	Not Visible	Not Required	Input	Number of decimal places for rate display. Valid = 0,1,2,3,4,5,6. Default is 2.
Cfg_CnfrmReqd	SINT	Not Visible	Not Required	Input	Operator Command Confirmation Required. Represents the type of command confirmation required. 0 = None, 1 = Command confirmation required, 2 = Performer e-signature required, 3 = Performer and approver e-signature required. Default is 0.
PCmd_Bump	BOOL	Not Visible	Not Required	Input	Program command to bump delivery for under tolerance. The instruction clears this operand automatically. Default is false.
PCmd_ClearTot	BOOL	Not Visible	Not Required	Input	Program command to clear totalizer quantity. The instruction clears this operand automatically. Default is false.
PCmd_StartTot	BOOL	Not Visible	Not Required	Input	Program command to start totalizer. The instruction clears this operand automatically. Default is false.
PCmd_StopTot	BOOL	Not Visible	Not Required	Input	Program command to stop totalizer. The instruction clears this operand automatically. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
PCmd_StartFlow	BOOL	Not Visible	Not Required	Input	Program command to start delivery. The instruction clears this operand automatically. Default is false.
PCmd_StopFlow	BOOL	Not Visible	Not Required	Input	Program command to stop/pause delivery. The instruction clears this operand automatically. Default is false.
PCmd_CheckTol	BOOL	Not Visible	Not Required	Input	Program command to check tolerances. The instruction clears this operand automatically. Default is false.
PCmd_Lock	BOOL	Not Visible	Not Required	Input	Program command to lock Program (disallow Operator). The instruction clears this operand automatically if Cfg_PCmdLockAsLevel = 0. Default is false.
PCmd_Normal	BOOL	Not Visible	Not Required	Input	Program command to select normal command source (Operator or Program). The instruction clears this operand automatically. Default is false.
PCmd_Oper	BOOL	Not Visible	Not Required	Input	Program command to select Operator (Program to Operator). The instruction clears this operand automatically. Default is false.
PCmd_Prog	BOOL	Not Visible	Not Required	Input	Program command to select Program (Operator to Program). The instruction clears this operand automatically. Default is false.
PCmd_Unlock	BOOL	Not Visible	Not Required	Input	Program command to unlock Program (allow Operator to acquire). The instruction clears this operand automatically. Default is false.
PCmd_Reset	BOOL	Not Visible	Not Required	Input	Program command to reset all alarms requiring reset. The instruction clears this operand automatically. Default is false.
PCmd_Virtual	BOOL	Not Visible	Not Required	Input	Program command to select Virtual (virtualized) device operation. The instruction clears this operand automatically. Default is false.
PCmd_Physical	BOOL	Not Visible	Not Required	Input	Program command to select Physical device operation (not virtualized). The instruction clears this operand automatically. Default is false.
PSet_Owner	DINT	Not Visible	Not Required	Input	Program owner request ID (non-zero) or release (zero). Default is 0.
PSet_DribbleQty	REAL	Not Visible	Not Required	Input	Program setting of quantity to dribble (Engineering Units). Valid = 0.0 to maximum positive float. Default is 0.0.
PSet_Preact	REAL	Not Visible	Not Required	Input	Program setting of amount before total to stop flow (Engineering Units). Valid = 0.0 to maximum positive float. Default is 0.0.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
PSet_SP	REAL	Not Visible	Not Required	Input	Program setting of total quantity to deliver in Engineering Units. Valid = 0.0 to maximum positive float. Default is 0.0.
PSet_TolHi	REAL	Not Visible	Not Required	Input	Program setting of high tolerance limit (ok amount > SP). Valid = 0.0 to maximum positive float. Default is 0.0.
PSet_TolLo	REAL	Not Visible	Not Required	Input	Program setting of low tolerance limit (ok amount < SP). Valid = 0.0 to maximum positive float. Default is 0.0.
XCmd_Bump	BOOL	Not Visible	Not Required	Input	External command to bump delivery for under tolerance.The instruction clears this operand automatically. Default is false.
XCmd_ClearTot	BOOL	Not Visible	Not Required	Input	External command to clear totalizer Quantity. The instruction clears this operand automatically. Default is false.
XCmd_StartTot	BOOL	Not Visible	Not Required	Input	External command to start totalizer.The instruction clears this operand automatically. Default is false.
XCmd_StopTot	BOOL	Not Visible	Not Required	Input	External command to stop totalizer.The instruction clears this operand automatically. Default is false.
XCmd_StartFlow	BOOL	Not Visible	Not Required	Input	External command to start deliver.The instruction clears this operand automatically. Default is false.
XCmd_StopFlow	BOOL	Not Visible	Not Required	Input	External command to stop/pause delivery.The instruction clears this operand automatically. Default is false.
XCmd_CheckTol	BOOL	Not Visible	Not Required	Input	External command to check tolerances.The instruction clears this operand automatically. Default is false.
XCmd_Reset	BOOL	Not Visible	Not Required	Input	External command to clear shed latches and cleared alarms. Default is false.
XCmd_Acq	BOOL	Not Visible	Not Required	Input	External command to Acquire ownership (Operator/Program/Override/Maintenance to External). The instruction clears this operand automatically if Cfg_ExtAcqAsLevel = 0. Default is false.
XCmd_Rel	BOOL	Not Visible	Not Required	Input	External command to release ownership if Cfg_ExtAcqAsLevel = 0 (External to Operator/Program/Override/Maintenance). The instruction clears this operand automatically. Default is false.
XCmd_ResetAckAll	BOOL	Not Visible	Not Required	Input	External command to acknowledge and reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
XSet_DribbleQty	REAL	Not Visible	Not Required	Input	External setting of quantity to dribble (Engineering Units). Default is 0.0.
XSet_Preact	REAL	Not Visible	Not Required	Input	External setting of amount before total to stop flow (Engineering Units). Default is 0.0.
XSet_SP	REAL	Not Visible	Not Required	Input	External setting of total quantity to deliver (Engineering Units). Default is 0.0.
XSet_TolHi	REAL	Not Visible	Not Required	Input	External setting of high tolerance limit (ok amount > SP). Default is 0.0.
XSet_TolLo	REAL	Not Visible	Not Required	Input	External setting of low tolerance limit (ok amount < SP). Default is 0.0.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
EnableOut	BOOL	Not Visible	Not Required	Output	Enable Output - System Defined Parameter
Out_ClearTot	BOOL	Visible	Not Required	Output	1 = Reset external totalizer (e.g. onboard flowmeter).
Out_RunTot	BOOL	Visible	Not Required	Output	1 = Run External totalizer (e.g. onboard flowmeter).
Out_RunFlow	BOOL	Visible	Not Required	Output	1 = Deliver at full (fast) flow.
Out_StopFlow	BOOL	Visible	Not Required	Output	1 = Stop delivery equipment.
Out_DribbleFlow	BOOL	Visible	Not Required	Output	1 = Deliver at dribble (slow) flow.
Out_Reset	BOOL	Not Visible	Not Required	Output	1 = Reset command has been received and accepted.
Out_OwnerSts	DINT	Not Visible	Not Required	Output	Status of command source, owner command handshake and ready status. 0 = None, .10 = Operator lock, .11 = Operator unlock, .12 = Program lock, .13 = Program unlock, .14 = Acquire Maintenance, .15 = Release Maintenance, .16 = Acquire External, .17 = Release External, .18 = Has Maintenance, .19 = External Override lock, .20 = Has External, .21 = Has Operator, .22 = Has Program, .30 = Not ready.
Sts_Initialized	BOOL	Not Visible	Not Required	Output	1 = Instruction is initialized. use Inp_InitializeReq to reinitialize.
Sts_CalcQty	BOOL	Not Visible	Not Required	Output	1 = Integrate Inp_RatePV to get quantity, 0 = Use Inp_QtyPV.
Sts_CalcRate	BOOL	Not Visible	Not Required	Output	1 = Differentiate Inp_QtyPV to get rate, 0 = Use Inp_RatePV.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_Cleared	BOOL	Visible	Not Required	Output	1 = Totalizer clear completed.
Sts_TotRunning	BOOL	Visible	Not Required	Output	1 = Totalizer running, 0 = Totalizer stopped.
Sts_FlowStarting	BOOL	Not Visible	Not Required	Output	1 = Flow is starting (Out_Run is on, feedback not showing run).
Sts_FlowRunning	BOOL	Visible	Not Required	Output	1 = Flow is running (Out_Run is on, feedback shows running).
Sts_FlowStopping	BOOL	Not Visible	Not Required	Output	1 = Flow is stopping (Out_Stop is on, feedback not showing stopped).
Sts_FlowStopped	BOOL	Visible	Not Required	Output	1 = Flow is stopped (Out_Stop is on, feedback shows stopped).
Sts_DribbleStarting	BOOL	Not Visible	Not Required	Output	1 = Dribble starting (Out_Dribble is on, feedback not showing dribble).
Sts_FlowDribble	BOOL	Visible	Not Required	Output	1 = Flow is dribbling (Out_Dribble is on, feedback shows dribble).
Sts_Bumping	BOOL	Not Visible	Not Required	Output	1 = Bump flow is active.
Sts_InTol	BOOL	Not Visible	Not Required	Output	1 = Total delivered is within tolerances.
Sts_Complete	BOOL	Visible	Not Required	Output	1 = Total delivered > (setpoint - preact).
Sts_Virtual	BOOL	Not Visible	Not Required	Output	1 = The instruction treats the device as virtual. The instruction acts as normal but the output is kept de-energized (Out_(x)=0). 0 = The instruction operates the device normally. Sts_Virtual is a copy of Inp_Virtual.
Sts_bSrc	INT	Not Visible	Not Required	Output	Active selection bitmap (for HMI totem pole with command source request selection): Sts_bSrc.0: Hand, Sts_bSrc.1: Programmed out of service (rung false), Sts_bSrc.2: Maintenance out of service, Sts_bSrc.3: Maintenance, Sts_bSrc.4: Override, Sts_bSrc.5: External, Sts_bSrc.6: Program locked, Sts_bSrc.7: Program, Sts_bSrc.8: Operator locked, Sts_bSrc.9: Operator.
Sts_eCmd	SINT	Not Visible	Not Required	Output	Dosing command 0 = None, 1 = Clear Totalizer, 2 = Start Totalizer, 3 = Start Flow, 4 = Start Dribble, 5 = Bump, 6 = Stop Flow, 7 = Stop Totalizer.
Sts_eFdbk	SINT	Not Visible	Not Required	Output	Equipment feedback 0 = None/transition, 1 = Flow stopped, 2 = Flow running, 3 = Flow dribbling, 4 = Virtualized.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_eFault	SINT	Not Visible	Not Required	Output	Object fault status 0 = None, 1 = Error: bad configuration, 15 = Equipment fault.
Sts_eNotifyAll	SINT	Not Visible	Not Required	Output	All alarm status enumerated values including related objects: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotify	SINT	Not Visible	Not Required	Output	All alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyEqpFault	SINT	Not Visible	Not Required	Output	Equipment Fault alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_eNotifyHiFlowRate	SINT	Not Visible	Not Required	Output	Hi Flow Rate alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyLoFlowRate	SINT	Not Visible	Not Required	Output	Lo Flow Rate alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyOverTol	SINT	Not Visible	Not Required	Output	Over Tolerance alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyUnderTol	SINT	Not Visible	Not Required	Output	Under Tolerance alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_eNotifyZeroFault	SINT	Not Visible	Not Required	Output	Zero Fault alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eSrc	INT	Not Visible	Not Required	Output	The current command source enumerated values: 0 = Logic not in use, 4 = Hand, 8 = Maintenance, 16 = Override, 32 = Program, 33 = Program locked, 34 = Program by default (Normal), 64 = Operator, 65 = Operator locked, 66 = Operator by default (Normal), 128 = Maintenance Out of Service, 129 = Programmed Out of Service (rung false), 256 = External.
Sts_eSts	SINT	Not Visible	Not Required	Output	Dosing confirmed Status 0 = ? 1 = Stop, 2 = Totalizer run, 3 = Flow run, 4 = Dribble, 5 = Bump 6 = Start flow, 7 = Start dribble, 8 = Stopping, 15 = Out of Service.
Sts_UnackAlmCount	SINT	Not Visible	Not Required	Output	Count of unacknowledged alarms.
Sts_Available	BOOL	Not Visible	Not Required	Output	1 = Dosing available for control by automation (Program).
Sts_MaintByp	BOOL	Not Visible	Not Required	Output	1 = A Maintenance bypass is active, display icon.
Sts_NotRdy	BOOL	Not Visible	Not Required	Output	1 = Object not ready, see detail bits for reason.
Sts_NrdyCfgErr	BOOL	Not Visible	Not Required	Output	1 = Object not ready: configuration error.
Sts_NrdyEqpFault	BOOL	Not Visible	Not Required	Output	1 = Object not ready: External equipment fault (fault or shed requires reset).
Sts_NrdyOoS	BOOL	Not Visible	Not Required	Output	1 = Object not ready: out of service.
Sts_NrdyPVBad	BOOL	Not Visible	Not Required	Output	1 = Object not ready: PV bad quality or comm failure.
Sts_Err	BOOL	Not Visible	Not Required	Output	1 = Error in configuration: See detail bits (Sts_Errxxx) for reason.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_ErrAlm	BOOL	Not Visible	Not Required	Output	1 = Error in config: alarm minimum on time, shelf time, severity.
Sts_ErrBumpTime	BOOL	Not Visible	Not Required	Output	1 = Error: Cfg_BumpTime invalid.
Sts_ErrClearPulseTime	BOOL	Not Visible	Not Required	Output	1 = Error: Cfg_ClearPulseTime invalid.
Sts_ErrCountsPerEU	BOOL	Not Visible	Not Required	Output	1 = Error: Cfg_CountsPerEU invalid.
Sts_ErrCutoff	BOOL	Not Visible	Not Required	Output	1 = Error: Cfg_Cutoff invalid.
Sts_ErrEUQtyMult	BOOL	Not Visible	Not Required	Output	1 = Error: Cfg_EUQtyMult invalid.
Sts_ErrFaultTime	BOOL	Not Visible	Not Required	Output	1 = Error: Cfg_FaultTime invalid.
Sts_ErrRateTime	BOOL	Not Visible	Not Required	Output	1 = Error: Cfg_RateTime invalid.
Sts_ErrLim	BOOL	Not Visible	Not Required	Output	1 = Error: Cfg_MaxQty invalid.
Sts_ErrRollover	BOOL	Not Visible	Not Required	Output	1 = Error: Cfg_Rollover invalid.
Sts_ErrRateFiltTimeConst	BOOL	Not Visible	Not Required	Output	1 = Error: Cfg_RateFiltTimeConst invalid.
Sts_ErrSettleTime	BOOL	Not Visible	Not Required	Output	1 = Error: Cfg_SettleTime invalid.
Sts_ErrVirtual	BOOL	Not Visible	Not Required	Output	1 = Error: Cfg_VirtualDribbleRate or Cfg_VirtualRate invalid.
Sts_Hand	BOOL	Not Visible	Not Required	Output	1 = Hand is selected (supersedes Out of Service, Maintenance, Override, External, Program, Operator).
Sts_OoS	BOOL	Not Visible	Not Required	Output	1 = Out of service is selected (supersedes Maintenance, Override, External, Program, Operator).
Sts_Maint	BOOL	Not Visible	Not Required	Output	1 = Maintenance is selected (supersedes Override, External, Program, Operator).
Sts_Ovrd	BOOL	Not Visible	Not Required	Output	1 = Override is selected (supersedes External, Program, Operator).
Sts_Ext	BOOL	Not Visible	Not Required	Output	1 = External is selected (supersedes Program and Operator).
Sts_Prog	BOOL	Not Visible	Not Required	Output	1 = Program is selected.
Sts_ProgLocked	BOOL	Not Visible	Not Required	Output	1 = Program is selected and locked.
Sts_Oper	BOOL	Not Visible	Not Required	Output	1 = Operator is selected.
Sts_OperLocked	BOOL	Not Visible	Not Required	Output	1 = Operator is selected and locked.
Sts_ProgOperSel	BOOL	Not Visible	Not Required	Output	Program/Operator selection (latch) state: 1 = Program, 0 = Operator.
Sts_ProgOperLock	BOOL	Not Visible	Not Required	Output	Program/Operator lock (latch) state, 1 = Locked, 0 = Unlocked.
Sts_Normal	BOOL	Not Visible	Not Required	Output	1 = Selection equals the normal (Program or Operator).
Sts_ExtReqInh	BOOL	Not Visible	Not Required	Output	1 = External request inhibited, cannot get to External from current state.
Sts_ProgReqInh	BOOL	Not Visible	Not Required	Output	1 = Program request inhibited, cannot get to Program from current state.
Sts_MAcqRcvd	BOOL	Not Visible	Not Required	Output	1 = Maintenance acquire command received this scan.
Sts_Alm	BOOL	Visible	Not Required	Output	1 = An alarm is active.
Sts_AlmInh	BOOL	Not Visible	Not Required	Output	1 = One or more alarms shelved, disabled or suppressed.
Sts_EqpFault	BOOL	Not Visible	Not Required	Output	1 = Equipment fault detected.
Sts_HiFlowRate	BOOL	Not Visible	Not Required	Output	1 = Hi flow rate alarm.
Sts_LoFlowRate	BOOL	Not Visible	Not Required	Output	1 = Lo flow rate alarm.
Sts_LoRateCutoff	BOOL	Not Visible	Not Required	Output	1 = Rate PV below low rate cutoff, flow assumed to be zero.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_OverTol	BOOL	Not Visible	Not Required	Output	1 = Delivery out of tolerance high.
Sts_UnderTol	BOOL	Not Visible	Not Required	Output	1 = Delivery out of tolerance low.
Sts_ZeroFault	BOOL	Not Visible	Not Required	Output	1 = Totalizer did not clear or unexpected flow.
Sts_QtyBad	BOOL	Not Visible	Not Required	Output	1 = Quantity value is Bad (PV Fail).
Sts_QtyUncertain	BOOL	Not Visible	Not Required	Output	1 = Quantity value is Uncertain.
Sts_RateBad	BOOL	Not Visible	Not Required	Output	1 = Rate value is Bad (PV Fail).
Sts_RateUncertain	BOOL	Not Visible	Not Required	Output	1 = Rate value is Uncertain.
Sts_RdyAck	BOOL	Not Visible	Not Required	Output	1 = An alarm is ready to be acknowledged.
Sts_RdyReset	BOOL	Not Visible	Not Required	Output	1 = A latched alarm or shed condition is ready to be reset.
Val_SP	REAL	Visible	Not Required	Output	Amount to be delivered (setpoint) (Engineering Units). Valid = 0.0 to maximum positive float.
Val_Remain	REAL	Visible	Not Required	Output	Amount yet to deliver to reach setpoint (Engineering Units). Valid = 0.0 to maximum positive float.
Val_PercentComplete	REAL	Visible	Not Required	Output	Percent complete (for progress bar on HMI) Valid = 0.0 to 100.0.
Val_QtyPV	REAL	Not Visible	Not Required	Output	Quantity from weigh scale or flowmeter (Engineering Units or pulse count). Valid = 0.0 to maximum positive float.
Val_Qty	REAL	Visible	Not Required	Output	Quantity actually delivered (totalizer output) (Engineering Units). Valid = 0.0 to maximum positive float.
Val_RatePV	REAL	Not Visible	Not Required	Output	Flow rate from flowmeter (Engineering Units/time, see Cfg_RateTime). Valid = 0.0 to maximum positive float.
Val_Rate	REAL	Visible	Not Required	Output	Current delivery rate (Engineering Units/time) (see Cfg_RateTime). Valid = 0.0 to maximum positive float.
Val_DribbleQty	REAL	Visible	Not Required	Output	Amount to be delivered at slow rate (Engineering Units). Valid = 0.0 to maximum positive float.
Val_Preact	REAL	Visible	Not Required	Output	Amount before setpoit at which flow will be stopped (Engineering Units). Valid = 0.0 to maximum positive float.
Val_TolHi	REAL	Visible	Not Required	Output	Allowed amount > Setpoint (Engineering Units). Valid = 0.0 to maximum positive float.
Val_TolLo	REAL	Visible	Not Required	Output	Allowed amount < Setpoint (Engineering Units). Valid = 0.0 to maximum positive float.
Val_Owner	DINT	Not Visible	Not Required	Output	Current object owner ID (0 = Not owned).

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
SrcQ	SINT	Not Visible	Not Required	Output	Source and quality of primary value or status (enumerated): 0=Good, live, confirmed good 1=Good, live, assumed good 2=Good, no feedback, assumed good 8=Test, virtualized 9=Test, loopback 10=Test, manually entered 16=Uncertain, live, off-spec 17=Uncertain, substituted at device or bus 18=Uncertain, substituted at instruction 19=Uncertain, using last known good 20=Uncertain, using replacement value 32=Bad, signal failure 33=Bad, channel fault 34=Bad, module or communication fault 35=Bad, invalid configuration
SrcQ_IO	SINT	Not Visible	Not Required	Output	Source and quality of primary input or output (enumerated): 0=Good, live, confirmed good 1=Good, live, assumed good 2=Good, no feedback, assumed good 8=Test, virtualized 9=Test, loopback 10=Test, manually entered 16=Uncertain, live, off-spec 17=Uncertain, substituted at device or bus 18=Uncertain, substituted at instruction 19=Uncertain, using last known good 20=Uncertain, using replacement value 32=Bad, signal failure 33=Bad, channel fault 34=Bad, module or communication fault 35=Bad, invalid configuration
XRdy_Acq	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Acq, enable HMI button.
XRdy_Rel	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Rel, enable HMI button.
XRdy_Reset	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Reset, enable HMI button.
XRdy_ResetAckAll	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_ResetAckAll, enable HMI button.

Private Input Members	Data Type	Description
HMI_BusObjIndex	DINT	HMI bus object index Default is 0.
MCmd_Acq	BOOL	Maintenance command to acquire ownership (Operator/Program/External/Override to Maintenance). The instruction clears this operand automatically. Default is false.
MCmd_IS	BOOL	Maintenance command to select in service. The instruction clears this operand automatically. Default is false.

Private Input Members	Data Type	Description
MCmd_OoS	BOOL	Maintenance command to select out of service. The instruction clears this operand automatically. Default is false.
MCmd_Physical	BOOL	Maintenance command to select Physical device operation (not virtualized) Default is false.
MCmd_Rel	BOOL	Maintenance command to release ownership (Maintenance to Operator/Program/External/Override). The instruction clears this operand automatically. Default is false.
MCmd_Virtual	BOOL	Maintenance command to select Virtual (virtualized) device operation Default is false.
OCmd_Bump	BOOL	Operator command to bump delivery for under tolerance. The instruction clears this operand automatically. Default is false.
OCmd_CheckTol	BOOL	Operator command to check tolerances. The instruction clears this operand automatically. Default is false.
OCmd_ClearTot	BOOL	Operator command to clear totalizer quantity. The instruction clears this operand automatically. Default is false.
OCmd_Lock	BOOL	Operator command to lock Operator (disallow Program). The instruction clears this operand automatically. Default is false.
OCmd_Normal	BOOL	Operator command to select normal (Operator or Program). The instruction clears this operand automatically. Default is false.
OCmd_Oper	BOOL	Operator command to select Operator (Program to Operator). The instruction clears this operand automatically. Default is false.
OCmd_Prog	BOOL	Operator command to select Program (Operator to Program). The instruction clears this operand automatically. Default is false.
OCmd_Reset	BOOL	Operator command to reset all alarms requiring reset. The instruction clears this operand automatically. Default is false.
OCmd_ResetAckAll	BOOL	Operator command to reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.
OCmd_StartFlow	BOOL	Operator command to start deliver. The instruction clears this operand automatically. Default is false.
OCmd_StartTot	BOOL	Operator command to start totalizer. The instruction clears this operand automatically. Default is false.
OCmd_StopFlow	BOOL	Operator command to stop/pause delivery. The instruction clears this operand automatically. Default is false.
OCmd_StopTot	BOOL	Operator command to stop totalizer. The instruction clears this operand automatically. Default is false.
OCmd_Unlock	BOOL	Operator command to unlock / release (allow Program to acquire) ownership. The instruction clears this operand automatically. Default is false.

Private Input Members	Data Type	Description
OSet_DribbleQty	REAL	Operator setting of quantity to dribble (Engineering Units). Valid = 0.0 to maximum positive float. Default is 0.0.
OSet_Preact	REAL	Operator setting of amount before total to stop flow (Engineering Units). Valid = 0.0 to maximum positive float. Default is 0.0.
OSet_SP	REAL	Operator setting of total quantity to deliver (Engineering Units). Valid = 0.0 to maximum positive float. Default is 0.0.
OSet_TolHi	REAL	Operator setting of high tolerance limit (ok amount > SP). Valid = 0.0 to maximum positive float. Default is 0.0.
OSet_TolLo	REAL	Operator setting of low tolerance limit (ok amount < SP). Valid = 0.0 to maximum positive float. Default is 0.0.

Private Output Members	Data Type	Description
MRdy_Acq	BOOL	1 = Ready for MCmd_Acq, enable HMI button.
MRdy_IS	BOOL	1 = Ready for MCmd_IS, enable HMI button.
MRdy_OoS	BOOL	1 = Ready for MCmd_OoS, enable HMI button.
MRdy_Physical	BOOL	1 = Ready for MCmd_Physical, enable HMI button.
MRdy_Rel	BOOL	1 = Ready for MCmd_Rel, enable HMI button.
MRdy_Virtual	BOOL	1 = Ready for MCmd_Virtual, enable HMI button.
ORdy_Bump	BOOL	1 = Ready for OCmd_Bump (enables HMI button).
ORdy_CheckTol	BOOL	1 = Ready for OCmd_CheckTol (enables HMI button).
ORdy_ClearTot	BOOL	1 = Ready for OCmd_ClearTot (enables HMI button).
ORdy_DribblePreact	BOOL	1 = Ready for OSet_DribblePreact (enables HMI entry field).
ORdy_Lock	BOOL	1 = Ready for OCmd_Lock, enable HMI button.
ORdy_Normal	BOOL	1 = Ready for OCmd_Normal, enable HMI button.
ORdy_Oper	BOOL	1 = Ready for OCmd_Oper, enable HMI button.
ORdy_Prog	BOOL	1 = Ready for OCmd_Prog, enable HMI button.
ORdy_Reset	BOOL	1 = Ready for OCmd_Reset (enables HMI button).
ORdy_ResetAckAll	BOOL	1 = Ready for OCmd_ResetAckAll (enables HMI button).
ORdy_SP	BOOL	1 = Ready for OSet_SP (enables HMI entry field).
ORdy_StartFlow	BOOL	1 = Ready for OCmd_StartFlow (enables HMI button).
ORdy_StartTot	BOOL	1 = Ready for OCmd_StartTot (enables HMI button).
ORdy_StopFlow	BOOL	1 = Ready for OCmd_StopFlow (enables HMI button).
ORdy_StopTot	BOOL	1 = Ready for OCmd_StopTot (enables HMI button).
ORdy_Tol	BOOL	1 = Ready for OSet_Tol (enables HMI entry field).
ORdy_Unlock	BOOL	1 = Ready for OCmd_Unlock, enable HMI button.
Sts_ErrInpSrc	BOOL	1 = Error: Input Source invalid. No Rate or Quantity PV connected

Public InOut Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Inp_QtyPV	REAL	Visible	Required	InOut	Quantity from weigh scale or flowmeter (Engineering Units or pulse count). Valid = 0.0 to maximum positive float.
Inp_RatePV	REAL	Visible	Required	InOut	Flow rate from flowmeter (Engineering Units/time, see Cfg_RateTime). Valid = 0.0 to 2147483.0 seconds.
BusObj	BUS_OBJ	Visible	Required	InOut	Bus component

BUS_OBJ Structure

Use InOut parameters to link the Instruction to external tags that contain necessary data for the instruction to operate. These external tags must be of the data type shown, or NULL. All public InOut parameters for this instruction may be NULL.

Members	Data Type	Description
Inp_Cmd	DINT	Input to assert commands
Out_Cmd	DINT	Resultant commands
Inp_CmdLLH	DINT	Input for level normally high commands
Out_CmdLLH	DINT	Resultant line level high commands
Inp_Sts	DINT	Input to assert status
Out_Sts	DINT	Resultant status
Inp_CmdAck	DINT	Input to assert a command acknowledgement
Out_CmdAck	DINT	Resultant command acknowledgements
Inp_SeverityMax	DINT	Input: maximum alarm severity
Out_SeverityMax	DINT	Resultant of maximum alarm severity
Cfg_CmdMask	DINT	Propagation mask for commands
Cfg_CmdLLHMask	DINT	Propagation mask for line level high commands
Cfg_StsMask	DINT	Propagation mask for status
Ref_Index	DINT	Bus array index

Alarms

Discrete tag-based alarms are defined for these members.

Member	Alarm Name	Description
Sts_EqpFault	Alm_EqpFault	Equipment fault. Raised when the Inp_CtrldEqpFault input is true, or when equipment feedback signals fail to track the commanded state of the equipment within the configured time. If an equipment fault is configured as a shed fault, the flow is stopped and a reset is required to resume flow.
Sts_HiFlowRate	Alm_HiFlowRate	Above high limit. Raised when the flow rate exceeds the High Flow Rate limit, for a configured period of time.
Sts_LoFlowRate	Alm_LoFlowRate	Below low limit. Raised when the flow rate falls short of the Low Flow Rate limit, for a configured period of time.
Sts_OverTol	Alm_OverTol	Above over tolerance limit. Raised when the tolerance check is performed and the quantity delivered exceeds the setpoint by more than the High Tolerance threshold.
Sts_UnderTol	Alm_UnderTol	Below under tolerance limit. Raised when the tolerance check is performed and the quantity delivered falls short of the setpoint by more than the Low Tolerance threshold.
Sts_ZeroFault	Alm_ZeroFault	Zero fault. Raised if the dosing fails to clear, or if the dosing is cleared but then registers flow before flow is commanded to start.

Mark the alarm as used or unused and set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format:

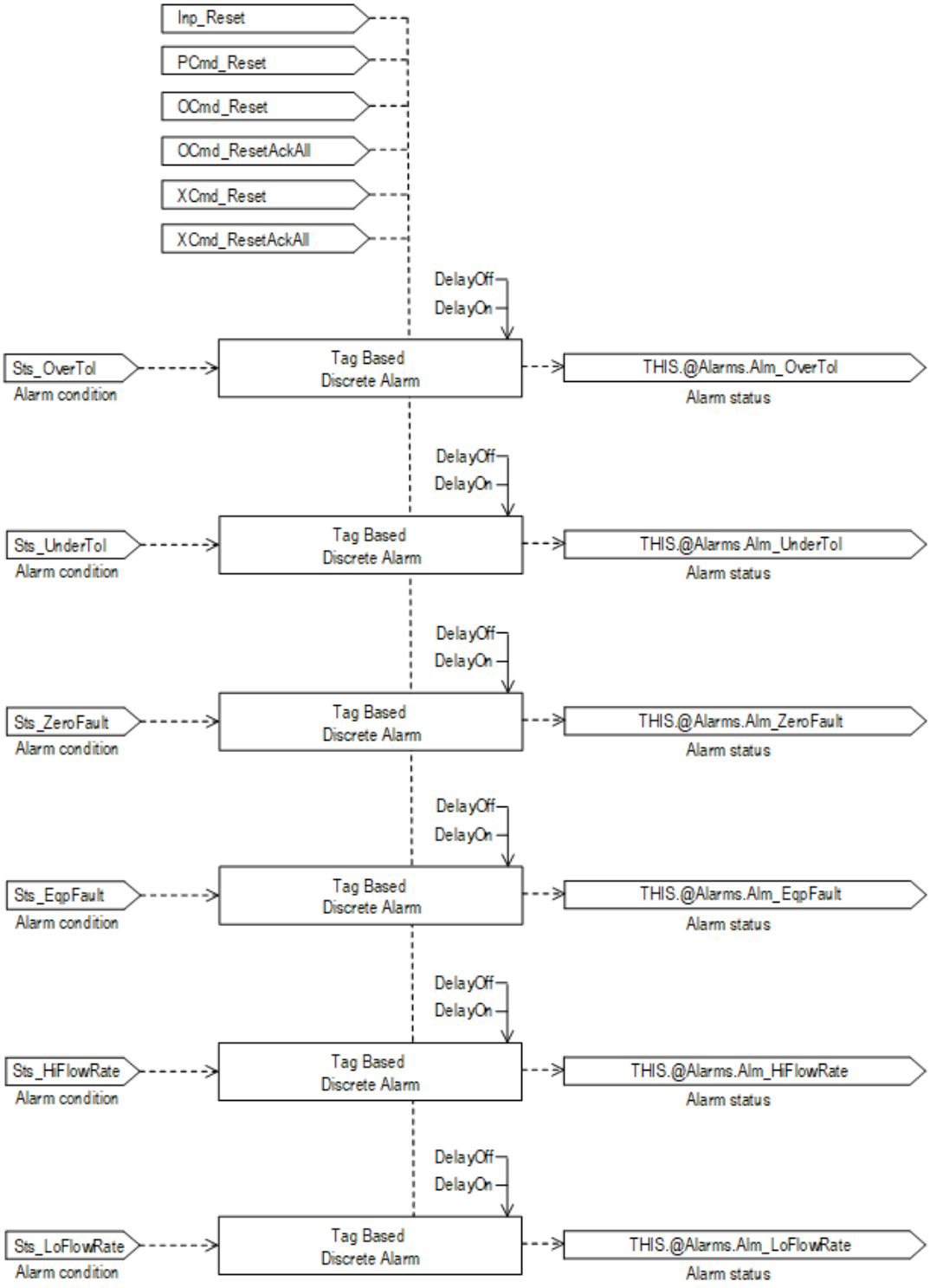
Tag.@Alarms.AlarmName.AlarmElement

PDOSETag.@Alarms.AlarmName.AlarmElement

The PDOSE instruction handles alarm conditions described by these use cases and conditions:

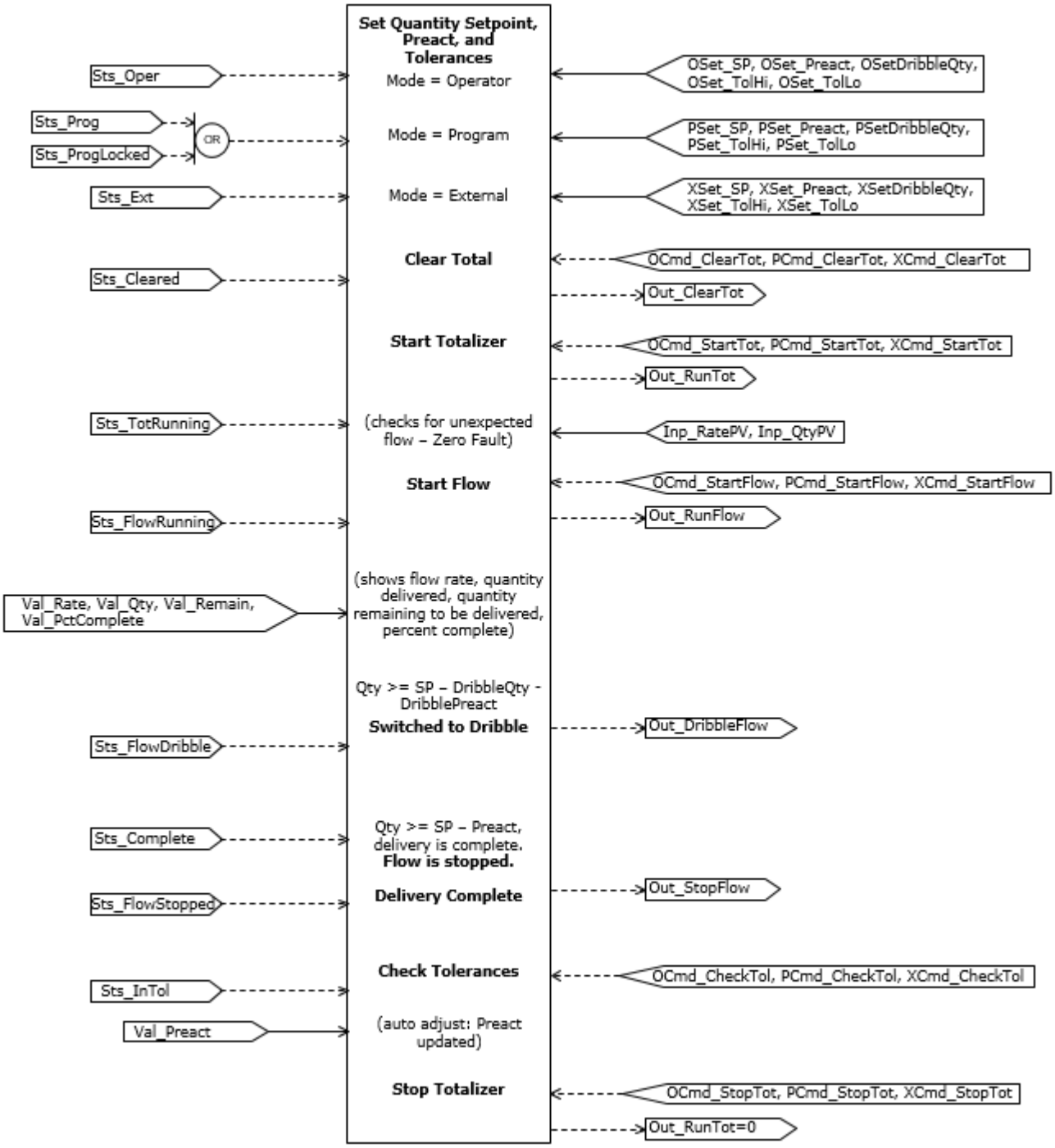
- Equipment fault - raised when the Inp_CtrldEqpFault input is true, or when equipment feedback signals fail to track the commanded state of the equipment within the configured time. If an equipment fault is configured as a shed fault, the flow is stopped and a reset is required to resume flow.
- Above high limit - raised when the flow rate exceeds the High Flow Rate limit, for a configured period of time.
- Below low limit - raised when the flow rate falls short of the Low Flow Rate limit, for a configured period of time.
- Above over tolerance limit - raised when the tolerance check is performed and the quantity delivered exceeds the setpoint by more than the High Tolerance threshold.
- Below under tolerance limit - raised when the tolerance check is performed and the quantity delivered falls short of the setpoint by more than the Low Tolerance threshold.
- Zero fault - raised if the dosing fails to clear, or if the dosing is cleared but then registers flow before flow is commanded to start.

There are Program, Operator, and External commands that enable the Reset and Reset & Acknowledge of all alarms of the instruction (Alarm Set) at the same time. This diagram shows how the commands interact with the PDOSE instruction.



Operation

This diagram illustrates the functionality of the PDOSE instruction:



Operator command request confirmation

The PDOSE instruction enables these operator command requests:

- OCmd _Bump
- OCmd_CheckTol
- OCmd_ClearTot
- OCmd_StartFlow
- OCmd_StartTot

- OCmd_StopFlow
- OCmd_StopTot
- OSet_DribbleQty
- OSet_Preact
- OSet_SP
- OSet_TolHi
- OSetTolLo

Enforced security might require the request to be confirmed or canceled before the selected command executes. The instruction checks the security rules inspecting Cfg_CnfrmReqd. If Cfg_CnfrmReqd=0 no confirmation is required and the request executes immediately. If Cfg_CnfrmReqd=1 the instruction waits for confirmation OCmd_CmdCnfrm=1 and/or cancellation For Cfg_CnfrmReqd=2 or 3 eSignature is needed before the confirmation and cancellation is enabled.

Virtualization

Use virtualization for instruction testing and operator training. Use PCmd_Virtual or MCmd_Virtual to enable virtualization. After finishing virtualization, use PCmd_Physical or MCmd_Physical to return to normal (physical device) operation.

When Virtualization is active, the instruction treats the object as virtual. The instruction acts as normal but the output is kept de-energized. The instruction can emulate a rate at which to dribble and a rate at which to deliver when running in virtual.

Initialization

The instruction is normally initialized in the instruction first run. Request re-initialization by setting Inp_InitializeReq = 1. For proper initialization, when adding the instruction while performing an online edit of the code, make sure that Inp_InitializeReq = 1, the default value.

Configuration of Strings for HMI

Configure strings for HMI faceplates (FactoryTalk View) and for Logix Designer configuration dialog. The strings are set to extended properties of tag items. Configure the strings in Logix Designer only.

- Description
- Label for graphic symbol
- Display Library for HMI Faceplate call-up
- Instruction name
- Area name

- URL link
- More Information
- Input Quantity PV
- Input Rate PV
- Quantity PV Units
- Rate PV Units
- Alarm Equipment Fault Text
- Alarm Hi Flow Rate Text
- Alarm Lo Flow Rate Text
- Alarm Over Tolerance Text
- Alarm Under Tolerance Text
- Alarm Zero Fault Text

Command Source

The instruction uses these command sources. The command sources are prioritized in order from highest to lowest in this table.

Command Source	Description
Hand	Hardwired logic or other logic outside the instruction controls the device. The instruction tracks the state of the device for bumpless transfer back to one of the other command sources. This is the highest priority command source.
Out-of-Service	The instruction is disabled and has no owner.
Maintenance	Maintenance controls the device and supersedes Operator, Program, External and Override control. Operator commands and settings from the HMI are accepted.
Override	Priority logic controls the device and supersedes Operator, Program and External control. Override Input (Inp_Ovrd) is accepted.
External	External logic (e.g. field pilot control or upstream SCADA) controls the device. External commands (XCmd_) are accepted.
Program locked	Program logic controls the device. Program commands (PCmd_) are accepted. Operator cannot take control from the Program. Override cannot take control from theProgram unless Cfg_OvrdOverLock = 1.
Program	Program logic controls the device. Program commands (PCmd_) are accepted.
Operator locked	The Operator controls the device. Operator commands (OCmd_) from the HMI are accepted. Program cannot take control from Operator. Override cannot take control from Operator unless Cfg_OvrdOverLock = 1.
Operator	The Operator controls the device. Operator commands (OCmd_) from the HMI are accepted. This is the lowest priority command source.

The instruction enables or disables these operations:

- Maintenance Out-of-Service exists
- Maintenance exists
- External exists
- Program (unlocked) exists
- Program locked exists
- Operator (unlocked) exists
- Operator locked exists
- PCmd_Lock used as a Level (1 = Lock, 0 = Unlock)

The instruction checks for invalid configurations of control and forces the nearest valid configuration.

The core control model arbitrates the source of the commands and parameters that the receiving function accepts. The core control model determines if the source is:

- A programmatic entity which resides entirely within the processing environment, or
- An external interface entity which issues commands and parameters external and asynchronously to the processing environment.

Locking a control source prevents the other control source from acquiring privilege.

Core Command Source Model

The core control model consists of these control sources:

- Oper
- OperLocked
- Prog
- ProgLocked

The control model defaults to this configuration. Other control sources may be present in the model but act as overriding control sources, acting independent of the base Operator/Program state machine.

Enabling control sources as Configuration

The individual control sources may be enabled or disabled by the user. The default configuration uses the entire base model; upon power-up of the processing environment the control source will be the designated default. Some combinations of enabled control sources are disallowed as they are either unnecessary or could create unintended changes.

Prog Power Up

Configuration allows the user to specify whether Operator or Program is the power-up default.

Prog Priority

Configuration allows the user to specify whether Operator or Program commands take priority when simultaneously asserted.

Automatic reset of commands

All commands are treated as one-shot-latched. This means that all commands are automatically cleared when the instruction executes and processes them.

Changing Destination States

Under certain configurations the destination command source for some commands may change. This is in keeping with the intent of the command. For example, if the Prog is disabled, the destination of the OCmd_Prog command is directed to the ProgLocked state instead of the Prog state. This maintains the intent of the OCmd_Prog command: the operator entity wishes to place the function in control of the program. If the command was eliminated there would be no way to accomplish this. This is only done in configurations where it would cause no conflict or race condition, but serves to preserve as much user functionality as is practical.

Higher Priority Command Sources

These Higher priority command sources operate independently within the model:

- External
- Override
- Maintenance
- Out-of-Service
- In-Service
- Hand

Monitor the PDOSE Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Rung-condition-out is cleared to false. Any commands that are received before first scan are discarded.
Instruction first run	Any commands received before first scan are discarded.
Rung-condition-in is false	Any commands that are received are discarded. All alarms are cleared. The command source is reported as Program Out of Service. The displayed rate is zeroed. Outputs to controlled equipment are de-energized. Other output parameters (values and status) hold their last value.
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	EnableOut is cleared to false. Any commands that are received before first scan are discarded.
Instruction first run	Any commands received before first scan are discarded.
Instruction first scan	See Instruction first run in the Function Block Diagram table.
EnableIn is false	EnableOut is cleared to false. Latched alarms are reset. Command source selection processing proceeds except that Program and Operators commands are ignored and cleared and all ownership status bits (Sts_Maint, Sts_Ovrd, Sts_Ext, Sts_Prog and Sts_Oper) are cleared to 0. When rung-condition-in becomes true, the Program/Operator selection is set based on the configuration (Cfg_ProgPwrUp).
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

In Structured Text, EnableIn is always true during normal scan. The instruction executes when it is in the control path activated by the logic.

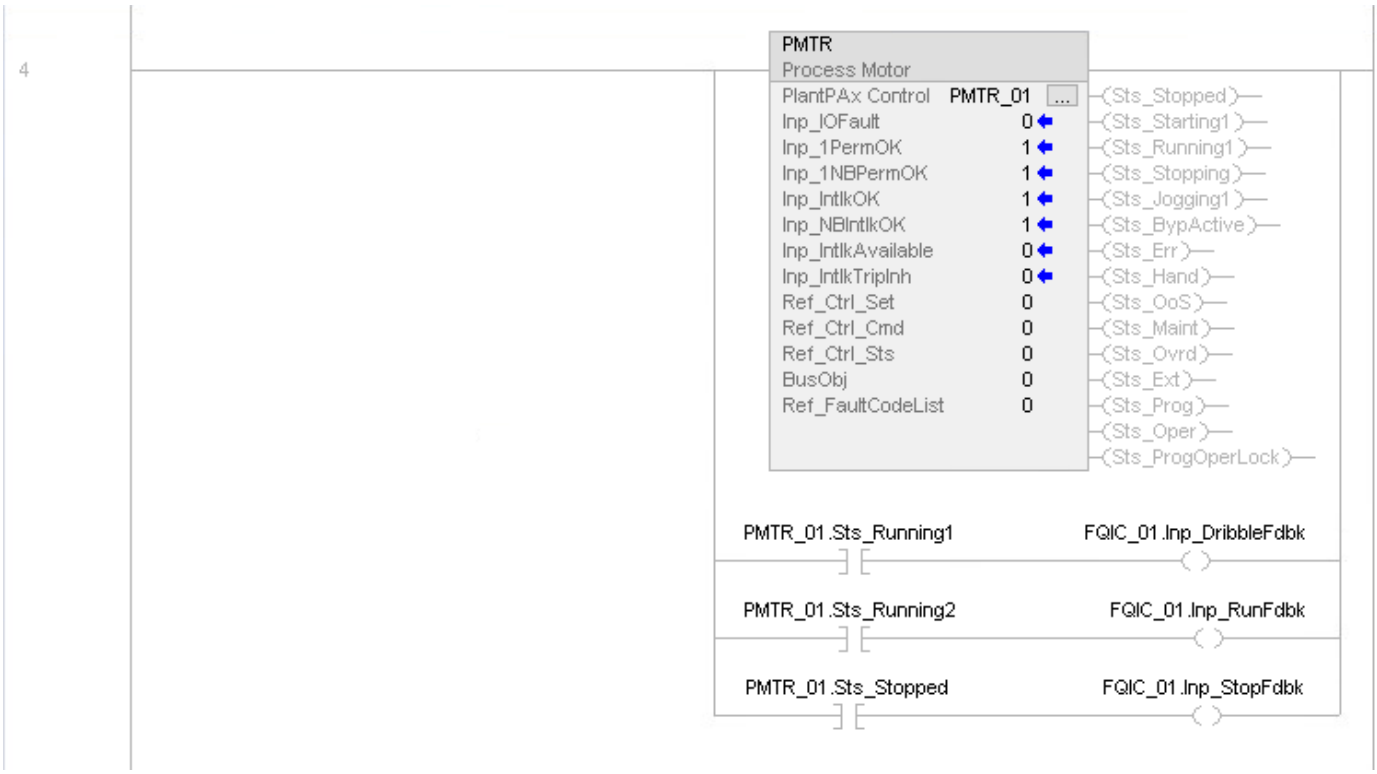
Condition/State	Action Taken
Prescan	See Prescan in the Function Block Diagram table.
Instruction first run	See Instruction first run in the Function Block Diagram table.
EnableIn is true	See EnableIn is true in the Function Block Diagram table.

Condition/State	Action Taken
Postscan	See Postscan in the Function Block Diagram table.

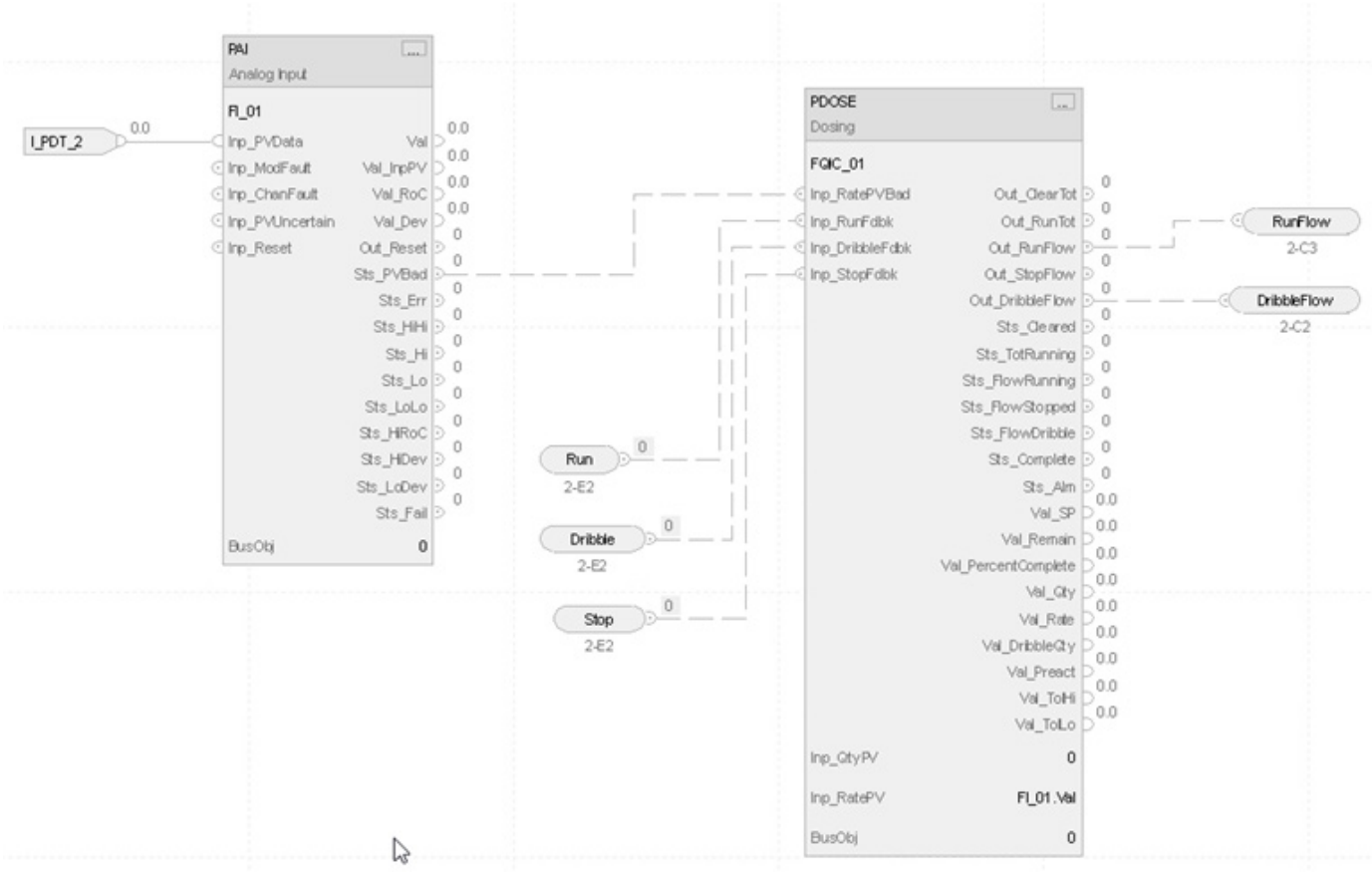
Example

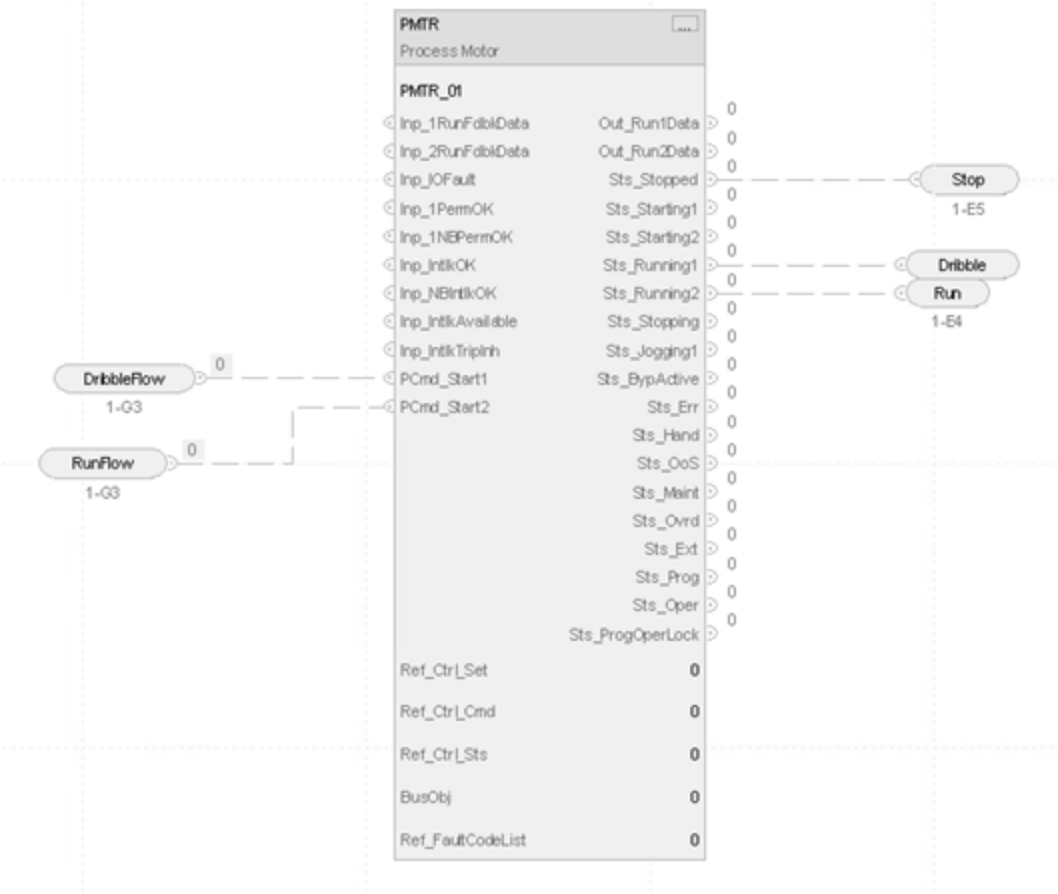
This example shows an input from a flowmeter (I_PDT_2) connected to a P_AInAdv block for the conversion of differential pressure to flow. The PV representing flow (Val from P_AInAdv) is the input for the PDOSE instruction (Inp_RatePV). The Sts_PVBad for the flow value is also used by the PDOSE instruction (Inp_RatePVBad). The outputs of the PDOSE instruction (Out_RunFlow and Out_DribbleFLow) are used as inputs to a two-speed motor (P_Motor2Spd). RunFlow and DribbleFlow are connected to PCmd_RunFast and PCmd_RunSlow, respectively. The status outputs of the motor for stopped (Sts_Stopped), running slow (Sts_RunningSlow), and running fast (Sts_RunningFast) are connected back to the PDOSE block as inputs Inp_StopFdbk, Inp_DribbleFdbk, and Inp_RunFdbk.

Ladder Diagram



Function Block Diagram





Structured Text

```
FI_01.Inp_PVData:=I_PDT_2;

PAI(FI_01,0);

FQIC_01.Inp_RatePVBad:=FI_01.Sts_PVBad;

PDOSE(FQIC_01, 0, FI_01.Val, 0);

PMTR_01.PCmd_Start1:=FQIC_01.Out_DribbleFlow;

PMTR_01.PCmd_Start2:=FQIC_01.Out_RunFlow;

PMTR(PMTR_01, 0, 0, 0, 0, 0);

FQIC_01.Inp_DribbleFdbk:=PMTR_01.Sts_Running1;

FQIC_01.Inp_RunFdbk:=PMTR_01.Sts_Running2;

FQIC_01.Inp_StopFdbk:=PMTR_01.Sts_Stopped;
```

See also

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Process Analog Fanout (PFO)

This information applies to the ControlLogix 5380P and 5580P controllers.

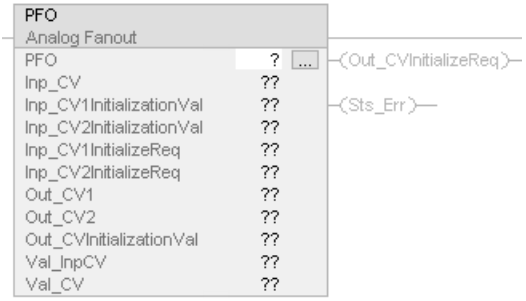
The Analog Fanout (PFO) instruction sends one primary analog output signal to multiple secondary users or devices. Each secondary output has configurable gain, offset, and clamping limits.

The Process Analog Fanout (PFO) instruction:

- Receives an input-controlled variable (CV) from a primary PID loop or analog output.
- Applies rate-of-change limiting to the input signal.
- Calculates outputs for up to eight secondary devices. Each secondary output has its own ratio (slope) and offset (intercept) from the rate-limited primary input. The ratios and offsets are configured values.
- Applies minimum and maximum clamping limits to each output (secondary) CV.
- Provides for initialization of each of its secondary CV outputs based on a request bit and a requested value from the secondary. When a particular output CV comes out of initialization, it is ramped from the initialization value to its calculated value using a configured takeup rate.
- Provides for initialization of the primary when all secondaries have requested initialization. The initialization value sent to the primary can be a fixed, configured value or a calculated value based on the CV1 (Output 1) requested initialization value, accounting for the CV1 gain and offset. Thus CV1 is the priority output.

Available Languages

Ladder Diagram



Function Block Diagram

Structured Text

PFO (PFO tag);

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PlantPAx Control	P_ANALOG_FANOUT	tag	Data structure required for proper operation of instruction.

P_ANALOG_FANOUT Structure

Public members are standard, visible tag members that are programmatically accessible. Private, hidden members are used in HMI faceplates and are not programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	Description
EnableIn	BOOL	Enable input. Ladder Diagram: Corresponds to the rung condition. Default is true.
Inp_InitializeReq	BOOL	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request when reinitialization is needed. The instruction clears this operand automatically. Default is true.
Inp_CV	REAL	Input CV from upstream block's output (engineering units). Default is 0.0.

Public Input Members	Data Type	Description
Inp_CV1InitializationVal	REAL	Initialization value from downstream block #1 (Out 1 engineering units). Valid = any float. Default is 0.0.
Inp_CV2InitializationVal	REAL	Initialization value from downstream block #2 (Out 2 engineering units). Valid = any float. Default is 0.0.
Inp_CV3InitializationVal	REAL	Initialization value from downstream block #3 (Out 3 engineering units). Valid = any float. Default is 0.0.
Inp_CV4InitializationVal	REAL	Initialization value from downstream block #4 (Out 4 engineering units). Valid = any float. Default is 0.0.
Inp_CV5InitializationVal	REAL	Initialization value from downstream block #5 (Out 5 engineering units). Valid = any float. Default is 0.0.
Inp_CV6InitializationVal	REAL	Initialization value from downstream block #6 (Out 6 engineering units). Valid = any float. Default is 0.0.
Inp_CV7InitializationVal	REAL	Initialization value from downstream block #7 (Out 7 engineering units). Valid = any float. Default is 0.0.
Inp_CV8InitializationVal	REAL	Initialization value from downstream block #8 (Out 8 engineering units). Valid = any float. Default is 0.0.
Inp_CV1InitializeReq	BOOL	Initialize request from downstream block #1 1 = set Out_CV1 to Inp_CV1InitializationVal. Default is false.
Inp_CV2InitializeReq	BOOL	Initialize request from downstream block #2 1 = set Out_CV2 to Inp_CV2InitializationVal. Default is false.
Inp_CV3InitializeReq	BOOL	Initialize request from downstream block #3 1 = set Out_CV3 to Inp_CV3InitializationVal. Default is false.
Inp_CV4InitializeReq	BOOL	Initialize request from downstream block #4 1 = set Out_CV4 to Inp_CV4InitializationVal. Default is false.
Inp_CV5InitializeReq	BOOL	Initialize request from downstream block #5 1 = set Out_CV5 to Inp_CV5InitializationVal. Default is false.
Inp_CV6InitializeReq	BOOL	Initialize request from downstream block #6 1 = set Out_CV6 to Inp_CV6InitializationVal. Default is false.
Inp_CV7InitializeReq	BOOL	Initialize request from downstream block #7 1 = set Out_CV7 to Inp_CV7InitializationVal. Default is false.
Inp_CV8InitializeReq	BOOL	Initialize request from downstream block #8 1 = set Out_CV8 to Inp_CV8InitializationVal. Default is false.
Cfg_HasCV2	BOOL	1 = Output CV #2 is connected. Default is true.
Cfg_HasCV3	BOOL	1 = Output CV #3 is connected. Default is false.
Cfg_HasCV4	BOOL	1 = Output CV #4 is connected. Default is false.
Cfg_HasCV5	BOOL	1 = Output CV #5 is connected. Default is false.
Cfg_HasCV6	BOOL	1 = Output CV #6 is connected. Default is false.

Public Input Members	Data Type	Description
Cfg_HasCV7	BOOL	1 = Output CV #7 is connected. Default is false.
Cfg_HasCV8	BOOL	1 = Output CV #8 is connected. Default is false.
Cfg_FixedInitializationVal	REAL	Fixed initialization value (in Inp_CV engineering units), used if Cfg_UseFixedInitialization = 1. Valid = any float. Default is 0.0.
Cfg_UseFixedInitialization	BOOL	1 = Use Cfg_UseFixedInitialization to initialize primary, 0 = use Inp_CV1InitializationVal. Default is false.
Cfg_ShedHold	BOOL	1 = Hold output on Inf / NaN input, 0 = copy Inf / NaN through. Default is false.
Cfg_HasMoreObj	BOOL	1 = Tells HMI an object with more information is available. Default is false.
Cfg_HasCVNav	BOOL	1 = Tells HMI to enable navigation to a connected CV object. Default is false.
Cfg_HasNav	SINT	Set bits indicate which navigation buttons are enabled .0=CV1, .1=CV2, ...7=CV8. Default is 0.
Cfg_CVEUMin	REAL	Input CV minimum in engineering units (for scaling). Valid = 0.0 to maximum positive float, Cfg_CVEUMin <> Cfg_CVEUMax. Default is 0.0.
Cfg_CVEUMax	REAL	Input CV maximum in engineering units (for scaling). Valid = 0.0 to maximum positive float, Cfg_CVEUMin <> Cfg_CVEUMax. Default is 100.0.
Cfg_CVLoLim	REAL	Input CV minimum, Lo clamp (Inp engineering units). Valid = any float, Cfg_CVLoLim<= Cfg_CVHiLim. Default is 0.0.
Cfg_CVHiLim	REAL	Input CV maximum, Hi clamp (Inp engineering units). Valid = any float, Cfg_CVHiLim>= Cfg_CVLoLim. Default is 100.0.
Cfg_CVRoCLim	REAL	Program setting for input CV rate of change limit, increase or decrease (Inp engineering units/seconds). Valid = 0.0 to maximum positive float. Default is 0.0.
Cfg_CV1Ratio	REAL	Configuration for CV1 ratio (m in mx+b). Valid = any float. Default is -2.0.
Cfg_CV1Offset	REAL	Configuration for CV1 offset (b in mx+b). Valid = any float. Default is 100.0.
Cfg_CV1LoLim	REAL	Output CV #1 minimum in engineering units (for clamping). Valid = any float, Cfg_CV1LoLim<= Cfg_CV1HiLim. Default is 0.0.
Cfg_CV1HiLim	REAL	Output CV #1 maximum in engineering units (for clamping). Valid = any float, Cfg_CV1HiLim>= Cfg_CV1LoLim. Default is 100.0.
Cfg_CV1TakeupRate	REAL	Rate (engineering units/seconds) at which CV1 bias is taken up after Inp_CV1InitializeReq = 0. Valid = 0.0 to maximum positive float. Default is 1.0.
Cfg_CV2Ratio	REAL	Configuration for CV2 ratio (m in mx+b). Valid = any float. Default is 2.0.
Cfg_CV2Offset	REAL	Configuration for CV2 offset (b in mx+b). Valid = any float. Default is -100.0.

Public Input Members	Data Type	Description
Cfg_CV2LoLim	REAL	Output CV #2 minimum in engineering units (for clamping). Valid = any float, Cfg_CV2LoLim<= Cfg_CV2HiLim. Default is 0.0.
Cfg_CV2HiLim	REAL	Output CV #2 maximum in engineering units (for clamping). Valid = any float, Cfg_CV2HiLim>= Cfg_CV2LoLim. Default is 100.0.
Cfg_CV2TakeupRate	REAL	Rate (engineering units/seconds) at which CV2 bias is taken up after Inp_CV2InitializeReq = 0. Valid = 0.0 to maximum positive float. Default is 1.0.
Cfg_CV3Ratio	REAL	Configuration for CV3 ratio (m in mx+b). Valid = any float. Default is 1.0.
Cfg_CV3Offset	REAL	Configuration for CV3 offset (b in mx+b). Valid = any float. Default is 0.0.
Cfg_CV3LoLim	REAL	Output CV #3 minimum in engineering units (for clamping). Valid = any float, Cfg_CV3LoLim<= Cfg_CV3HiLim. Default is 0.0.
Cfg_CV3HiLim	REAL	Output CV #3 maximum in engineering units (for clamping). Valid = any float, Cfg_CV3HiLim>= Cfg_CV3LoLim. Default is 100.0.
Cfg_CV3TakeupRate	REAL	Rate (engineering units/seconds) at which CV3 bias is taken up after Inp_CV3InitializeReq = 0. Valid = 0.0 to maximum positive float. Default is 1.0.
Cfg_CV4Ratio	REAL	Configuration for CV4 ratio (m in mx+b). Valid = any float. Default is 1.0.
Cfg_CV4Offset	REAL	Configuration for CV4 offset (b in mx+b). Valid = any float. Default is 0.0.
Cfg_CV4LoLim	REAL	Output CV #4 minimum in engineering units (for clamping). Valid = any float, Cfg_CV4LoLim<= Cfg_CV4HiLim. Default is 0.0.
Cfg_CV4HiLim	REAL	Output CV #4 maximum in engineering units (for clamping). Valid = any float, Cfg_CV4HiLim>= Cfg_CV4LoLim. Default is 100.0.
Cfg_CV4TakeupRate	REAL	Rate (engineering units/seconds) at which CV4 bias is taken up after Inp_CV4InitializeReq = 0. Valid = 0.0 to maximum positive float. Default is 1.0.
Cfg_CV5Ratio	REAL	Configuration for CV5 ratio (m in mx+b). Valid = any float. Default is 1.0.
Cfg_CV5Offset	REAL	Configuration for CV5 offset (b in mx+b). Valid = any float. Default is 0.0.
Cfg_CV5LoLim	REAL	Output CV #5 minimum in engineering units (for clamping). Valid = any float, Cfg_CV5LoLim<= Cfg_CV5HiLim. Default is 0.0.
Cfg_CV5HiLim	REAL	Output CV #5 maximum in engineering units (for clamping). Valid = any float, Cfg_CV5HiLim>= Cfg_CV5LoLim. Default is 100.0.
Cfg_CV5TakeupRate	REAL	Rate (engineering units/seconds) at which CV5 bias is taken up after Inp_CV5InitializeReq = 0. Valid = 0.0 to maximum positive float. Default is 1.0.
Cfg_CV6Ratio	REAL	Configuration for CV6 ratio (m in mx+b). Valid = any float. Default is 1.0.
Cfg_CV6Offset	REAL	Configuration for CV6 offset (b in mx+b). Valid = any float. Default is 0.0.

Public Input Members	Data Type	Description
Cfg_CV6LoLim	REAL	Output CV #6 minimum in engineering units (for clamping). Valid = any float, Cfg_CV6LoLim<= Cfg_CV6HiLim. Default is 0.0.
Cfg_CV6HiLim	REAL	Output CV #6 maximum in engineering units (for clamping). Valid = any float, Cfg_CV6HiLim>= Cfg_CV6LoLim. Default is 100.0.
Cfg_CV6TakeupRate	REAL	Rate (engineering units/seconds) at which CV6 bias is taken up after Inp_CV6InitializeReq = 0. Valid = 0.0 to maximum positive float. Default is 1.0.
Cfg_CV7Ratio	REAL	Configuration for CV7 ratio (m in mx+b). Valid = any float. Default is 1.0.
Cfg_CV7Offset	REAL	Configuration for CV7 offset (b in mx+b). Valid = any float. Default is 0.0.
Cfg_CV7LoLim	REAL	Output CV #7 minimum in engineering units (for clamping). Valid = any float, Cfg_CV7LoLim<= Cfg_CV7HiLim. Default is 0.0.
Cfg_CV7HiLim	REAL	Output CV #7 maximum in engineering units (for clamping). Valid = any float, Cfg_CV7HiLim>= Cfg_CV7LoLim. Default is 100.0.
Cfg_CV7TakeupRate	REAL	Rate (engineering units/seconds) at which CV7 bias is taken up after Inp_CV7InitializeReq = 0. Valid = 0.0 to maximum positive float. Default is 1.0.
Cfg_CV8Ratio	REAL	Configuration for CV8 ratio (m in mx+b). Valid = any float. Default is 1.0.
Cfg_CV8Offset	REAL	Configuration for CV8 offset (b in mx+b). Valid = any float. Default is 0.0.
Cfg_CV8LoLim	REAL	Output CV #8 minimum in engineering units (for clamping). Valid = any float, Cfg_CV8LoLim<= Cfg_CV8HiLim. Default is 0.0.
Cfg_CV8HiLim	REAL	Output CV #8 maximum in engineering units (for clamping). Valid = any float, Cfg_CV8HiLim>= Cfg_CV8LoLim. Default is 100.0.
Cfg_CV8TakeupRate	REAL	Rate (engineering units/seconds) at which CV8 bias is taken up after Inp_CV8InitializeReq = 0. Valid = 0.0 to maximum positive float. Default is 1.0.
Cfg_CVDecPlcs	SINT	Number of decimal places for CV display (0...6). Default is 0.

Public Output Members	Data Type	Description
EnableOut	BOOL	Enable output. This output state always reflects EnableIn input state.
Out_CV1	REAL	Output to downstream block #1 (out 1 engineering units).
Out_CV2	REAL	Output to downstream block #2 (out 2 engineering units).
Out_CV3	REAL	Output to downstream block #3 (out 3 engineering units).
Out_CV4	REAL	Output to downstream block #4 (out 4 engineering units).
Out_CV5	REAL	Output to downstream block #5 (out 5 engineering units).
Out_CV6	REAL	Output to downstream block #6 (out 6 engineering units).
Out_CV7	REAL	Output to downstream block #7 (out 7 engineering units).
Out_CV8	REAL	Output to downstream block #8 (out 8 engineering units).
Out_CVInitializationVal	REAL	Initialization value to upstream block (Inp_CV engineering units).
Out_CVInitializeReq	BOOL	Initialization request to upstream block (1 = initialize).

Public Output Members	Data Type	Description
Val_CVEUMin	REAL	Minimum of scaled range = minimum (Cfg_CVEUMin, Cfg_CVEUMax).
Val_CVEUMax	REAL	Maximum of scaled range = maximum (Cfg_CVEUMin, Cfg_CVEUMax).
Val_InpCV	REAL	Value of Inp_CV, not clamped or ramped (engineering units).
Val_CV	REAL	Value of CV after clamping and ramping (engineering units).
Val_MinCVIn1	REAL	Input CV at minimum of CV1 output (for HMI use).
Val_MaxCVIn1	REAL	Input CV at maximum of CV1 output (for HMI use).
Val_MinCVIn2	REAL	Input CV at minimum of CV2 output (for HMI use).
Val_MaxCVIn2	REAL	Input CV at maximum of CV2 output (for HMI use).
Val_MinCVIn3	REAL	Input CV at minimum of CV3 output (for HMI use).
Val_MaxCVIn3	REAL	Input CV at maximum of CV3 output (for HMI use).
Val_MinCVIn4	REAL	Input CV at minimum of CV4 output (for HMI use).
Val_MaxCVIn4	REAL	Input CV at maximum of CV4 output (for HMI use).
Val_MinCVIn5	REAL	Input CV at minimum of CV5 output (for HMI use).
Val_MaxCVIn5	REAL	Input CV at maximum of CV5 output (for HMI use).
Val_MinCVIn6	REAL	Input CV at minimum of CV6 output (for HMI use).
Val_MaxCVIn6	REAL	Input CV at maximum of CV6 output (for HMI use).
Val_MinCVIn7	REAL	Input CV at minimum of CV7 output (for HMI use).
Val_MaxCVIn7	REAL	Input CV at maximum of CV7 output (for HMI use).
Val_MinCVIn8	REAL	Input CV at minimum of CV8 output (for HMI use).
Val_MaxCVIn8	REAL	Input CV at maximum of CV8 output (for HMI use).
Sts_Initialized	BOOL	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.
Sts_CVInfNaN	BOOL	1 = Inp_CV is infinite or not a number (1.$, 1.#NaN).
Sts_CVLimited	BOOL	1 = Output CV clamped at configured maximum/minimum.
Sts_CV1InitializationInfNaN	BOOL	1 = Inp_CV1InitializationVal is infinite or not a number (1.$, 1.#NaN).
Sts_CV1Limited	BOOL	1 = Output CV1 clamped at configured maximum/minimum.
Sts_CV2InitializationInfNaN	BOOL	1 = Inp_CV2InitializationVal is infinite or not a number (1.$, 1.#NaN).
Sts_CV2Limited	BOOL	1 = Output CV2 clamped at configured maximum/minimum.
Sts_CV3InitializationInfNaN	BOOL	1 = Inp_CV3InitializationVal is infinite or not a number (1.$, 1.#NaN).
Sts_CV3Limited	BOOL	1 = Output CV3 clamped at configured maximum/minimum.
Sts_CV4InitializationInfNaN	BOOL	1 = Inp_CV4InitializationVal is infinite or not a number (1.$, 1.#NaN).
Sts_CV4Limited	BOOL	1 = Output CV4 clamped at configured maximum/minimum.
Sts_CV5InitializationInfNaN	BOOL	1 = Inp_CV5InitializationVal is infinite or not a number (1.$, 1.#NaN).
Sts_CV5Limited	BOOL	1 = Output CV5 clamped at configured maximum/minimum.
Sts_CV6InitializationInfNaN	BOOL	1 = Inp_CV6InitializationVal is infinite or not a number (1.$, 1.#NaN).
Sts_CV6Limited	BOOL	1 = Output CV6 clamped at configured maximum/minimum.
Sts_CV7InitializationInfNaN	BOOL	1 = Inp_CV7InitializationVal is infinite or not a number (1.$, 1.#NaN).
Sts_CV7Limited	BOOL	1 = Output CV7 clamped at configured maximum/minimum.
Sts_CV8InitializationInfNaN	BOOL	1 = Inp_CV8InitializationVal is infinite or not a number (1.$, 1.#NaN).
Sts_CV8Limited	BOOL	1 = Output CV8 clamped at configured maximum/minimum.
Sts_Err	BOOL	1 = Error in configuration: See detail bits (Sts_Errxxx) for reason.
Sts_ErrLim	BOOL	1 = Error in configuration: CV clamp limits crossed (maximum<minimum).
Sts_ErrEU	BOOL	1 = Error in configuration: CV scale EU minimum = maximum.
Sts_ErrCV1Lim	BOOL	1 = Error in configuration: CV1 clamp limits crossed (maximum<minimum).
Sts_ErrCV2Lim	BOOL	1 = Error in configuration: CV2 clamp limits crossed (maximum<minimum).
Sts_ErrCV3Lim	BOOL	1 = Error in configuration: CV3 clamp limits crossed (maximum<minimum).
Sts_ErrCV4Lim	BOOL	1 = Error in configuration: CV4 clamp limits crossed (maximum<minimum).
Sts_ErrCV5Lim	BOOL	1 = Error in configuration: CV5 clamp limits crossed (maximum<minimum).
Sts_ErrCV6Lim	BOOL	1 = Error in configuration: CV6 clamp limits crossed (maximum<minimum).
Sts_ErrCV7Lim	BOOL	1 = Error in configuration: CV7 clamp limits crossed (maximum<minimum).
Sts_ErrCV8Lim	BOOL	1 = Error in configuration: CV8 clamp limits crossed (maximum<minimum).

Initialization

The instruction is normally initialized in the instruction first run. Request re-initialization by setting Inp_InitializeReq = 1. For proper initialization, when adding the instruction while performing an online edit of the code, make sure that Inp_InitializeReq = 1, the default value.

Configuration of Strings for HMI

Configure strings for HMI faceplates (FT View) and for the Logix Designer configuration dialog box. The strings are set to extended properties of tag items. Configure the strings in Logix Designer only.

- Description
- Label for graphic symbol
- Display Library for HMI Faceplate call-up
- Instruction name
- Area name
- URL link
- CV Units
- Output CV Label CV1
- Output CV Label CV2
- Output CV Label CV3
- Output CV Label CV4
- Output CV Label CV5
- Output CV Label CV6
- Output CV Label CV7
- Output CV Label CV8
- CV1 EU (Engineering Units)
- CV2 EU
- CV3 EU
- CV4 EU
- CV5 EU
- CV6 EU
- CV7 EU
- CV8 EU
- Allow Navigation Object Tag Name Input CV
- Allow Navigation Object Tag Name CV1
- Allow Navigation Object Tag Name CV2
- Allow Navigation Object Tag Name CV3
- Allow Navigation Object Tag Name CV4
- Allow Navigation Object Tag Name CV5
- Allow Navigation Object Tag Name CV6
- Allow Navigation Object Tag Name CV7
- Allow Navigation Object Tag Name CV8

Monitor the PFO Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Rung-condition-out is cleared to false.
Instruction first run	The instruction executes normally.
Rung-condition-in is false	Rung-condition-out is cleared to false.
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	EnableOut is cleared to false.
Instruction first run	The instruction executes normally.
Instruction first scan	See Instruction first run in the Function Block Diagram table.
EnableIn is false	EnableOut is cleared to false.
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

In Structured Text, EnableIn is always true during normal scan. The instruction executes when it is in the control path activated by the logic.

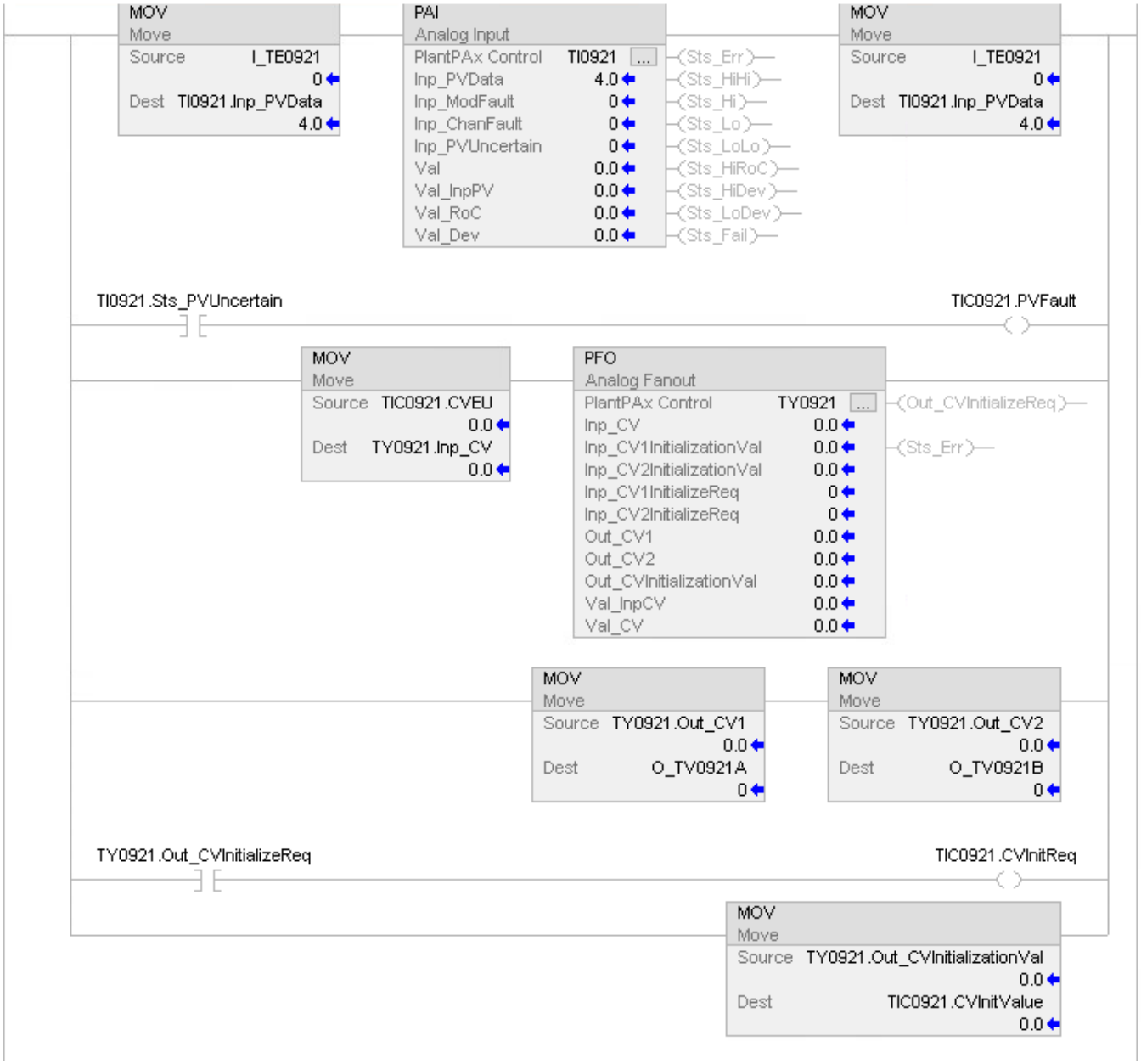
Condition/State	Action Taken
Prescan	See Prescan in the Function Block Diagram table.

Instruction first run	See Instruction first run in the Function Block Diagram table.
EnableIn is true	See EnableIn is true in the Function Block Diagram table.
Postscan	See Postscan in the Function Block Diagram table.

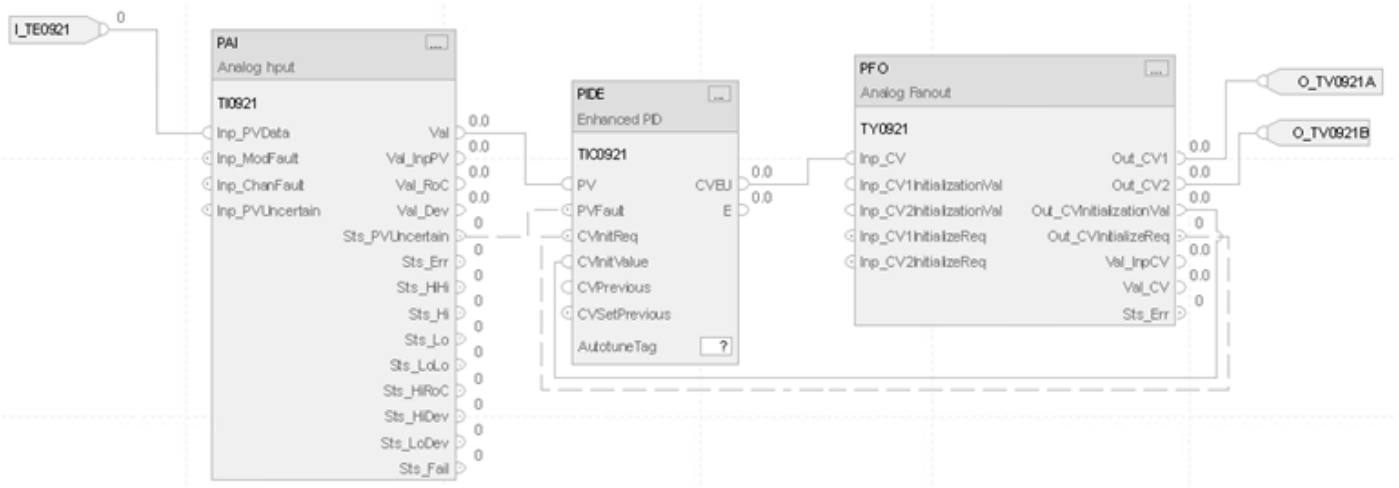
Example

In this example, the PFO instruction to implement a split range PID control strategy to control temperature of a processing vessel. The heat exchanger to the vessel jacket is fed by a steam valve to heat or a glycol valve to cool. One PID controls the temperature. The example assumes that the relative process gain between each valve and the temperature is the same.

Ladder Diagram



Function Block Diagram



Structured Text

TI0921.Inp_PVData:=I_TE0921;

PAI(TI0921);

TIC0921.PV:= TI0921.Val;

TIC0921.PVFault:=TI0921.Sts_PVUncertain;

PIDE(TIC0921);

TY0921.Inp_CV:=TIC0921.CVEU;

TIC0921.CVInitReq:=TY0921.Out_CVInitializeReq;

TIC0921.CVInitValue:=TY0921.Out_CVInitializationVal;

O_TV0921A:=TY0921.Out_CV1;

O_TV0921B:=TY0921.Out_CV2;

PFO(TY0921);

See also

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Process High or Low Selector (PHLS)

This information applies to the ControlLogix 5380P and 5580P controllers.

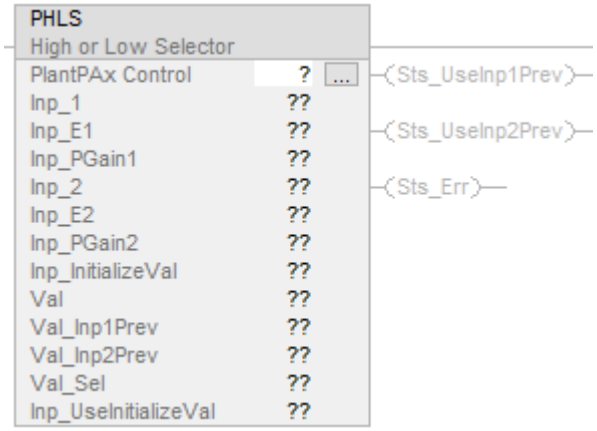
The Process High or Low Selector (PHLS) instruction selects the highest or the lowest of up to six incoming controlled variables (CVs). The instruction sends the selected CV as output and flags the unselected CVs to track the selected CV.

To avoid problems with ever-decreasing or ever-increasing output, offset the tracking value by an amount equal to the upstream PID/PIDE gain time's error value.

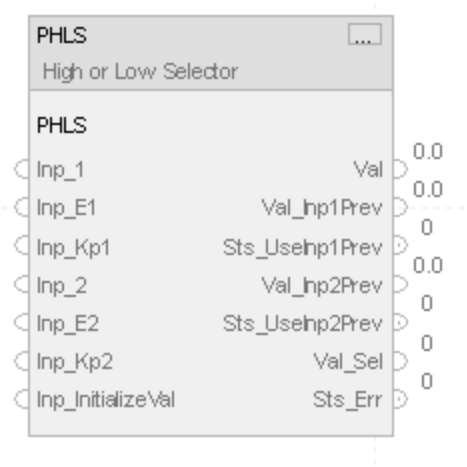
Use the PHLS instruction to implement an Override Select control strategy. An Override Select strategy provides control of a primary process variable while allowing other process variables to override the output on the final control element, which avoids exceeding constraints.

Available Languages

Ladder Diagram



Function Block Diagram



Structured Text

PHLS(PHLSTag);

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PHLS	P_HIGH_LOW_SELECT	tag	PHLS structure

PHLS Input Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. Ladder Diagram: Corresponds to the rung-condition-in. Default is true.
Inp_InitializeReq	BOOL	Use this request when reinitializing. Default is false.
Inp_1	REAL	Input #1. Valid = Any float. Default is 0.0.

Input Members	Data Type	Description
Inp_E1	REAL	Loop error from primary #1 (optional, used for offset calculation). Valid = Any float. Default is 0.0.
Inp_PGain1	REAL	Proportional gain from primary #1 (optional, used for offset calculation). Valid = Any float. Default is 0.0.
Inp_2	REAL	Input #2. Valid = Any float. Default is 0.0.
Inp_E2	REAL	Loop error from primary #2 (optional, used for offset calculation). Valid = Any float. Default is 0.0.
Inp_PGain2	REAL	Proportional gain from primary #2 (optional, used for offset calculation). Valid = Any float. Default is 0.0.
Inp_3	REAL	Input #3. Valid = Any float. Default is 0.0.
Inp_E3	REAL	Loop error from primary #3 (optional, used for offset calculation). Valid = Any float. Default is 0.0.
Inp_PGain3	REAL	Proportional gain from primary #3 (optional, used for offset calculation). Valid = Any float. Default is 0.0.
Inp_4	REAL	Input #4. Valid = Any float. Default is 0.0.
Inp_E4	REAL	Loop error from primary #4 (optional, used for offset calculation). Valid = Any float. Default is 0.0.
Inp_PGain4	REAL	Proportional gain from primary #4 (optional, used for offset calculation). Valid = Any float. Default is 0.0.
Inp_5	REAL	Input #5. Valid = Any float. Default is 0.0.
Inp_E5	REAL	Loop error from primary #5 (optional, used for offset calculation). Valid = Any float. Default is 0.0.
Inp_PGain5	REAL	Proportional gain from primary #5 (optional, used for offset calculation). Valid = Any float. Default is 0.0.
Inp_6	REAL	Input #6. Valid = Any float. Default is 0.0.
Inp_E6	REAL	Loop error from primary #6 (optional, used for offset calculation). Valid = Any float. Default is 0.0.
Inp_PGain6	REAL	Proportional gain from primary #6 (optional, used for offset calculation). Valid = Any float. Default is 0.0.
Inp_InitializeVal	REAL	Initialization value from downstream block. Valid = Any float. Default is 0.0.
Cfg_HiLoSel	BOOL	Selection: 1 = High - Select, 0 = Low - Select. Default is false.
Cfg_HasInp1	BOOL	1 = Inp_1 is connected. Default is true.
Cfg_UseInp1	BOOL	1 = Inp_1 is included in selection (for maintenance use). Default is true.
Cfg_Inp1Offset	BOOL	1 = Offset Inp_1, Offset = Val +/- Inp_PGain1 * Inp_E1. Default is false.
Cfg_HasInp2	BOOL	1 = Inp_2 is connected. Default is true.
Cfg_UseInp2	BOOL	1 = Inp_2 is included in selection (for maintenance use). Default is true.
Cfg_Inp2Offset	BOOL	1 = Offset Inp_2, Offset = Val +/- Inp_PGain2 * Inp_E2. Default is false.

Input Members	Data Type	Description
Cfg_HasInp3	BOOL	1 = Inp_3 is connected. Default is false.
Cfg_UseInp3	BOOL	1 = Inp_3 is included in selection (for maintenance use). Default is false.
Cfg_Inp3Offset	BOOL	1 = Offset Inp_3, Offset = Val +/- Inp_PGain3 * Inp_E3. Default is false.
Cfg_HasInp4	BOOL	1 = Inp_4 is connected. Default is false.
Cfg_UseInp4	BOOL	1 = Inp_4 is included in selection (for maintenance use). Default is false.
Cfg_Inp4Offset	BOOL	1 = Offset Inp_4, Offset = Val +/- Inp_PGain4 * Inp_E4. Default is false.
Cfg_HasInp5	BOOL	1 = Inp_5 is connected. Default is false.
Cfg_UseInp5	BOOL	1 = Inp_5 is included in selection (for maintenance use). Default is false.
Cfg_Inp5Offset	BOOL	1 = Offset Inp_5, Offset = Val +/- Inp_PGain5 * Inp_E5. Default is false.
Cfg_HasInp6	BOOL	1 = Inp_6 is connected. Default is false.
Cfg_UseInp6	BOOL	1 = Inp_6 is included in selection (for maintenance use). Default is false.
Cfg_Inp6Offset	BOOL	1 = Offset Inp_6, Offset = Val +/- Inp_PGain6 * Inp_E6. Default is false.
Cfg_DecPlcs	SINT	Number of decimal places for display. Valid = 0 to 6. Default is 2.
Cfg_HasOutNav	BOOL	1 = Tells HMI to enable navigation to a connected output object. Default is false.
Cfg_HasNav	SINT	Set bits to indicate which navigation buttons are enabled: Cfg_HasNav.0 = Inp_1, Cfg_HasNav.1 = Inp_2, Cfg_HasNav.2 = Inp_3, Cfg_HasNav.3 = Inp_4, Cfg_HasNav.4 = Inp_5, Cfg_HasNav.5 = Inp_6. Default is 0.
Cfg_OutLoLim	REAL	Output low clamping (engineering units). Default is 0.0.
Cfg_OutHiLim	REAL	Output high clamping (engineering units). Default is 100.0.
Cfg_HasMoreObj	BOOL	1 = Tells HMI an object with more info is available. Default is false.

Output Members	Data Type	Description
EnableOut	BOOL	This output state always reflects EnableIn input state.
Val	REAL	Output value (selected minimum or maximum) for downstream block.
Val_Inp1Prev	REAL	Previous (Feedback) input value for primary #1.
Sts_UseInp1Prev	BOOL	Request for primary #1 to use feedback Val_Inp1Prev.

Output Members	Data Type	Description
Val_Inp2Prev	REAL	Previous (Feedback) input value for primary #2.
Sts_UseInp2Prev	BOOL	Request for primary #2 to use feedback Val_Inp2Prev.
Val_Inp3Prev	REAL	Previous (Feedback) input value for primary #3.
Sts_UseInp3Prev	BOOL	Request for primary #3 to use feedback Val_Inp3Prev.
Val_Inp4Prev	REAL	Previous (Feedback) input value for primary #4.
Sts_UseInp4Prev	BOOL	Request for primary #4 to use feedback Val_Inp4Prev.
Val_Inp5Prev	REAL	Previous (Feedback) input value for primary #5.
Sts_UseInp5Prev	BOOL	Request for primary #5 to use feedback Val_Inp5Prev.
Val_Inp6Prev	REAL	Previous (Feedback) input value for primary #6.
Sts_UseInp6Prev	BOOL	Request for primary #6 to use feedback Val_Inp6Prev.
Val_Out	REAL	Output value (selected minimum or maximum Input) for HMI.
Val_Sel	DINT	Selected input: 0 = Minimum, 1 = Inp_1, 2 = Inp_2, 3 = Inp_3, 4 = Inp_4, 5 = Inp_5, 6 = Inp_6, 7 = Maximum, 8 = Inp_InitializeVal.
Sts_Initialized	BOOL	1 = Instruction is initialized.
Sts_MaintByp	BOOL	1 = A maintenance bypass is active (display icon on HMI)
Sts_Err	BOOL	1 = Error in configuration, check Sts_ErrHas or Sts_ErrLim for reason.
Sts_ErrHas	BOOL	1 = Configuration error: must HAVE at least one Input.
Sts_ErrLim	BOOL	1 = Configuration error: Cfg_OutHiLim less than or equal to Cfg_OutLoLim.

Alarms

The PHLS instruction does not have any alarms.

Virtualization

The PHLS instruction does not have any virtualization capability.

Configuration of Strings for HMI

Configure strings for HMI faceplates (FT View) and for the Logix Designer configuration dialog box. The strings are set to extended properties of tag items. Configure the strings in Logix Designer only.

- Description - Description of PHLS Tag
- Label for graphic symbol – Label metadata of PHLS tag
- Display Library for HMI Faceplate call-up - Library metadata of PHLS tag

- Instruction name - Instruction metadata of PHLS tag
- Area name - Area metadata of PHLS tag
- URL link - URL metadata of PHLS tag
- Output Units – Engineering Unit metadata of PHLS member Val_Out tag
- Input Label Input1 – Description metadata of PHLS member Inp_1 tag
- Input Label Input2 – Description metadata of PHLS member Inp_2 tag
- Input Label Input3 – Description metadata of PHLS member Inp_3 tag
- Input Label Input4 – Description metadata of PHLS member Inp_4 tag
- Input Label Input5 – Description metadata of PHLS member Inp_5 tag
- Input Label Input6 – Description metadata of PHLS member Inp_6 tag
- Input1 EU – Engineering Unit metadata of PHLS member Inp_1 tag
- Input2 EU – Engineering Unit metadata of PHLS member Inp_2 tag
- Input3 EU – Engineering Unit metadata of PHLS member Inp_3 tag
- Input4 EU – Engineering Unit metadata of PHLS member Inp_4 tag
- Input5 EU – Engineering Unit metadata of PHLS member Inp_5 tag
- Input6 EU – Engineering Unit metadata of PHLS member Inp_6 tag
- Allow Navigation Object Tag Name Output - Navigation metadata of PHLS member Out_Val tag
- Allow Navigation Object Tag Name Input1 - Navigation metadata of PHLS member Inp_1 tag
- Allow Navigation Object Tag Name Input2 - Navigation metadata of PHLS member Inp_2 tag
- Allow Navigation Object Tag Name Input3 - Navigation metadata of PHLS member Inp_3 tag
- Allow Navigation Object Tag Name Input4 - Navigation metadata of PHLS member Inp_4 tag
- Allow Navigation Object Tag Name Input5 - Navigation metadata of PHLS member Inp_5 tag
- Allow Navigation Object Tag Name Input6 - Navigation metadata of PHLS member Inp_6 tag

Monitor the PHLS Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Rung-condition-out is cleared to false.
Instruction first run	All configurations has input, use input and input offset are cleared. All previous feedback, loop error, gain values is set to 0. The instruction executes normally.
Rung-condition-in is false	Rung-condition-out is cleared to false. All configurations has input, use input and input offset are cleared. All previous feedback, loop error, gain values is set to 0. The instruction is kept in its last state.
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	EnableOut is cleared to false.
Instruction first run	All configurations has input, use input and input offset are cleared. All previous feedback, loop error, gain values is set to 0. The instruction executes normally.
Instruction first scan	See Instruction first run in the Function Block Diagram table.
EnableIn is false	EnableOut is cleared to false. The instruction will hold its last selection state and output value (Val) is not updated (holds last value).
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

In Structured Text, EnableIn is always true during normal scan. The instruction executes when it is in the control path activated by the logic.

Condition/State	Action Taken
Prescan	See Prescan in the Function Block Diagram table.
Instruction first run	See Instruction first run in the Function Block Diagram table.
EnableIn is true	See EnableIn is true in the Function Block Diagram table.
Postscan	See Postscan in the Function Block Diagram table.

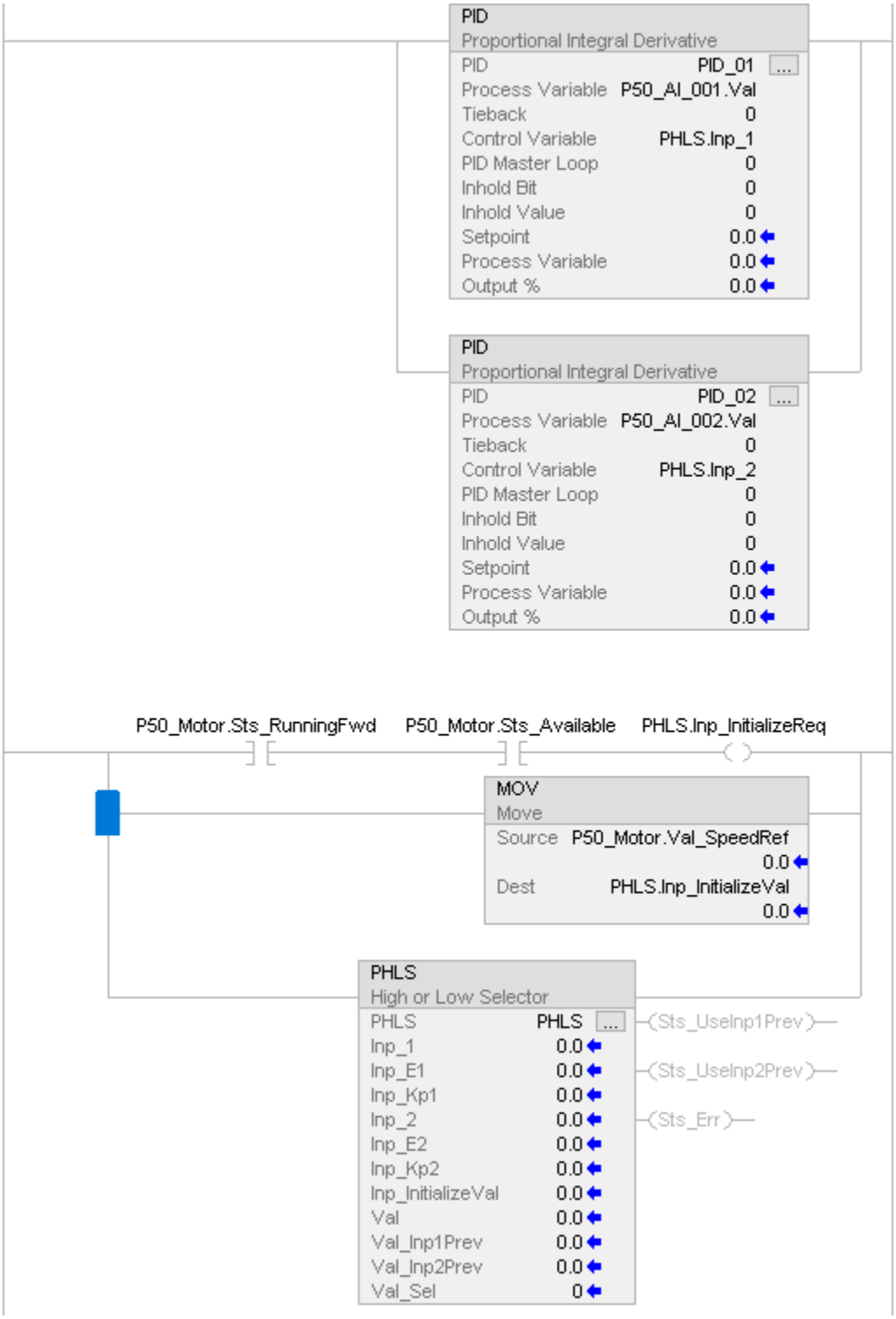
Example

The example uses the PHLS instruction to implements part of the pressure control strategy. In this case, two PIDE instructions are used as inputs. The PIDE instructions are for Suction Pressure Override Control and Discharge Pressure Control. The PIDE output values CV (CV to final control element) and E (Loop Error) are used as inputs to the PHLS instruction.

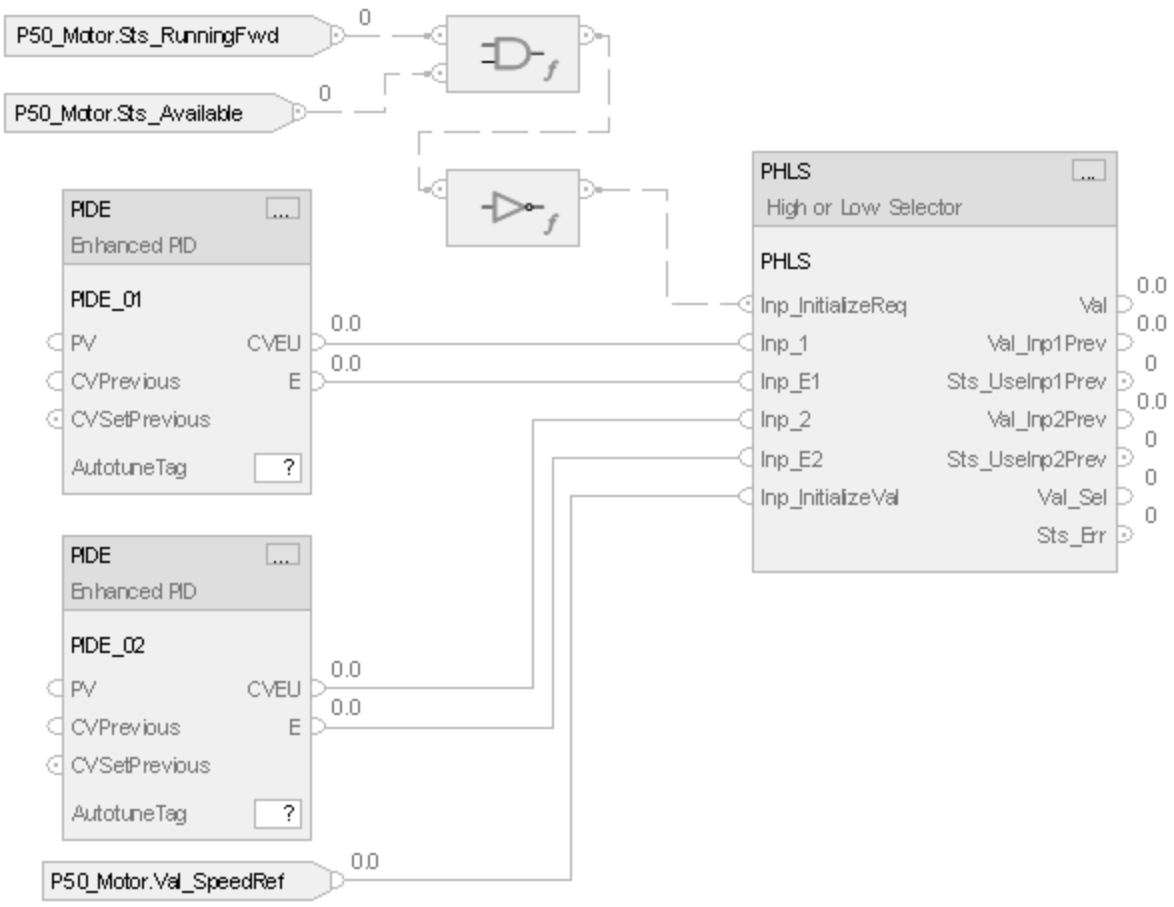
This example also shows PHLS inputs for Initial Value (Inp_InitializeVal) and initialization request (Inp_InitializeReq). In this case, the Initial Value is taken from the speed reference to the pump motor drive. The Initialization

In this example the instruction initialization request flag is set based on the motor's running and availability status.

Ladder Diagram



Function Block Diagram



Structured text

PIDE(PIDE_01);

PIDE(PIDE_02);

PHLS.Inp_1 := PIDE_01.CV;

PHLS.Inp_E1 := PIDE_01.E;

PHLS.Inp_2 := PIDE_02.CV;

PHLS.Inp_E2 := PIDE_02.E;

PHLS.Inp_InitializeVal := P50_Motor.Val_SpeedRef;

PHLS.Inp_InitializeReq := NOT(P50_Motor.Sts_RunningFwd AND P50_Motor.Sts_Available);

PHLS(PHLS);

See also

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Process Interlocks (PINTLK)

This information applies to the ControlLogix 5380P and 5580P controllers.

The Process Interlocks (PINTLK) instruction collects, or sums up, the interlock conditions that stop or de-energize a running or energized piece of equipment. This instruction can also help prevent equipment from starting or being energized. Interlocks are always evaluated to de-energize equipment. For permissive conditions that must be made to start the equipment, but are ignored once the equipment is running, use the Process Permissive (PPERM) instruction.

The PINTLK instruction provides:

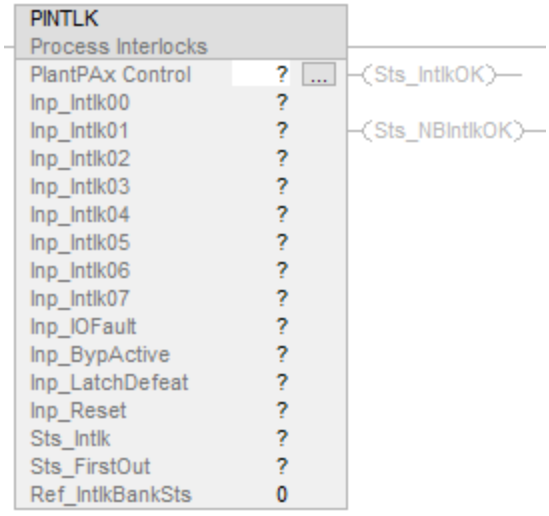
- Interlock input OK check: Each input is compared with its configured OK state. If the input is not in its OK state, it raises an interlock condition unless bypassed.
- Interlock Condition Latching: If the input is configured as latched, the interlock condition is latched until reset, unless the latch defeat input is true. If the input is not configured as latched, the interlock condition clears when the input returns to it's OK state.
- Interlock Bypass: If the input is configured as able to be bypassed and interlocks are bypassed, the input does not raise an interlock condition, even if it is not in its OK state. If the input is configured as not able to be bypassed or if interlocks are not bypassed, the input raises an interlock condition. Engineering configures which interlocks are allowed to be bypassed. Maintenance chooses which inputs to bypass from the interlocks that are allowed by engineering.
- First Out: If no interlock conditions are raised (OK to run), the first interlock condition to be raised is marked as the first out. If multiple interlock conditions are raised in the same scan, they are all marked as first out.
- Latch Defeat: A latch defeat function reduces the number of operator actions that are required to start equipment. The latch defeat input is set when the equipment is not running. When the latch defeat input is true, the latched configuration of inputs is ignored, and all interlock conditions clear when their corresponding inputs are in their OK states. This action saves the operator from having to reset before starting the equipment. When the equipment starts, the latch defeat

input is turned off. Then, if an interlock condition configured as latched shuts down the equipment, it remains latched until reset.

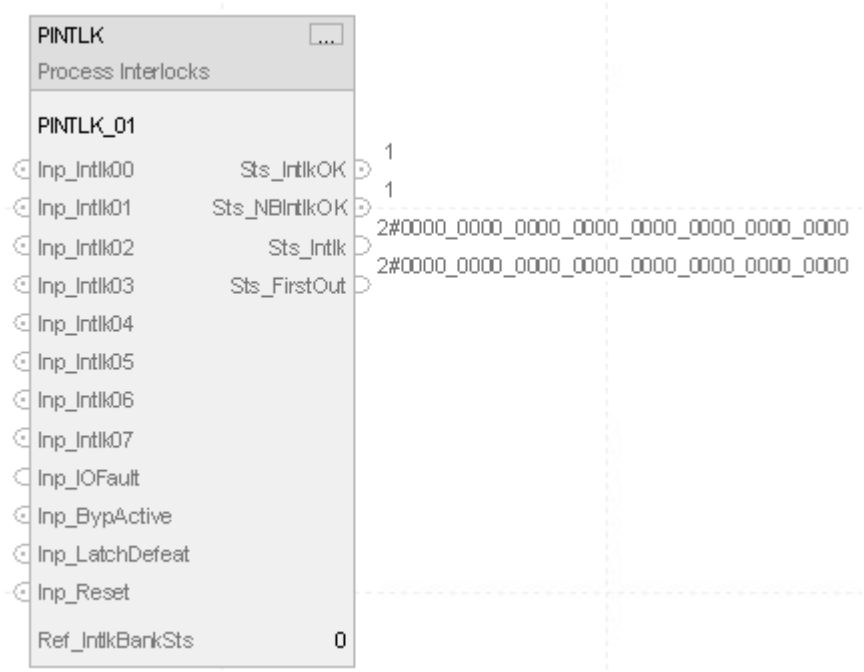
- Summary Status: Summarizes its 32 interlock input conditions into two primary status bits:
  - Sts_IntlkOK. Indicates all interlock conditions are clear and ready to run.
  - Sts_NBIntlkOK. Indicates all interlock conditions that cannot be bypassed are clear and ready to run if interlocks are bypassed.

Available Languages

Ladder Diagram



Function Block Diagram



Structured Text

PINTLK (PINTLK tag, Ref_IntlkBankSts);

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PlantPAx	P_INTERLOCK	tag	Data structure required for proper operation of instruction.
Ref_IntlkBankSts	P_INTERLOCK_BANK_STATUS	tag	Reference interlock bank status.

P_INTERLOCK Structure

Public members are standard, visible tag members that are programmatically accessible. Private, hidden members are used in HMI faceplates and are not programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	Description
EnableIn	BOOL	Enable Input - System Defined Parameter Default is true.
Inp_InitializeReq	BOOL	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request when reinitialization is needed. The instruction clears this operand automatically. Default is true.
Inp_Intlk00	BOOL	Interlock condition 00, de-energize if not in configured OK state. Default is false.
Inp_Intlk01	BOOL	Interlock condition 01, de-energize if not in configured OK state. Default is false.
Inp_Intlk02	BOOL	Interlock condition 02, de-energize if not in configured OK state. Default is false.
Inp_Intlk03	BOOL	Interlock condition 03, de-energize if not in configured OK state. Default is false.
Inp_Intlk04	BOOL	Interlock condition 04, de-energize if not in configured OK state. Default is false.
Inp_Intlk05	BOOL	Interlock condition 05, de-energize if not in configured OK state. Default is false.
Inp_Intlk06	BOOL	Interlock condition 06, de-energize if not in configured OK state. Default is false.
Inp_Intlk07	BOOL	Interlock condition 07, de-energize if not in configured OK state. Default is false.
Inp_Intlk08	BOOL	Interlock condition 08, de-energize if not in configured OK state. Default is false.
Inp_Intlk09	BOOL	Interlock condition 09, de-energize if not in configured OK state. Default is false.
Inp_Intlk10	BOOL	Interlock condition 10, de-energize if not in configured OK state. Default is false.
Inp_Intlk11	BOOL	Interlock condition 11, de-energize if not in configured OK state. Default is false.
Inp_Intlk12	BOOL	Interlock condition 12, de-energize if not in configured OK state. Default is false.
Inp_Intlk13	BOOL	Interlock condition 13, de-energize if not in configured OK state. Default is false.
Inp_Intlk14	BOOL	Interlock condition 14, de-energize if not in configured OK state. Default is false.
Inp_Intlk15	BOOL	Interlock condition 15, de-energize if not in configured OK state. Default is false.
Inp_Intlk16	BOOL	Interlock condition 16 , de-energize if not in configured OK state. Default is false.
Inp_Intlk17	BOOL	Interlock condition 17 , de-energize if not in configured OK state. Default is false.
Inp_Intlk18	BOOL	Interlock condition 18 , de-energize if not in configured OK state. Default is false.
Inp_Intlk19	BOOL	Interlock condition 19 , de-energize if not in configured OK state. Default is false.
Inp_Intlk20	BOOL	Interlock condition 20 , de-energize if not in configured OK state. Default is false.
Inp_Intlk21	BOOL	Interlock condition 21 , de-energize if not in configured OK state. Default is false.
Inp_Intlk22	BOOL	Interlock condition 22 , de-energize if not in configured OK state. Default is false.

Public Input Members	Data Type	Description
Inp_Intlk23	BOOL	Interlock condition 23 , de-energize if not in configured OK state. Default is false.
Inp_Intlk24	BOOL	Interlock condition 24 , de-energize if not in configured OK state. Default is false.
Inp_Intlk25	BOOL	Interlock condition 25 , de-energize if not in configured OK state. Default is false.
Inp_Intlk26	BOOL	Interlock condition 26 , de-energize if not in configured OK state. Default is false.
Inp_Intlk27	BOOL	Interlock condition 27 , de-energize if not in configured OK state. Default is false.
Inp_Intlk28	BOOL	Interlock condition 28 , de-energize if not in configured OK state. Default is false.
Inp_Intlk29	BOOL	Interlock condition 29 , de-energize if not in configured OK state. Default is false.
Inp_Intlk30	BOOL	Interlock condition 30 , de-energize if not in configured OK state. Default is false.
Inp_Intlk31	BOOL	Interlock condition 31 , de-energize if not in configured OK state. Default is false.
Inp_IOFault	DINT	Input register for IO fault logic. Default is 2#0000_0000_0000_0000_0000_0000_0000_0000.
Inp_Available	BOOL	External availability input. Default is true.
Inp_BypActive	BOOL	1 = Interlock Bypassing is currently active. Default is false.
Inp_LatchDefeat	BOOL	Set when device is de-energized. 1 =Do not latch inputs, even if configured for latching, and do not capture a new first-out. Default is false.
Inp_Reset	BOOL	1 = Reset latched interlocks and first-out. Default is false.
Cfg_OKState	DINT	Bits indicate which state (0 or 1) of each input is OK to run. Default is 2#0000_0000_0000_0000_0000_0000_0000_0000.
Cfg_Latched	DINT	Set bits indicate which conditions are latched (sealed in). Default is 2#0000_0000_0000_0000_0000_0000_0000_0000.
Cfg_StopOnly	DINT	Set bits indicate which conditions cause a stop - do not trip. Default is 2#0000_0000_0000_0000_0000_0000_0000_0000.
Cfg_Bypassable	DINT	Set bits indicate which conditions can be bypassed. Default is 2#0000_0000_0000_0000_0000_0000_0000_0000.
Cfg_HasNav	DINT	Set bits indicate which navigation buttons are enabled. Default is 2#0000_0000_0000_0000_0000_0000_0000_0000.
Cfg_eType00	SINT	Enumerated type of Interlock 0. Default is 6.
Cfg_eType01	SINT	Enumerated type of Interlock 1. Default is 6.
Cfg_eType02	SINT	Enumerated type of Interlock 2. Default is 6.
Cfg_eType03	SINT	Enumerated type of Interlock 3. Default is 6.
Cfg_eType04	SINT	Enumerated type of Interlock 4. Default is 6.
Cfg_eType05	SINT	Enumerated type of Interlock 5. Default is 6.
Cfg_eType06	SINT	Enumerated type of Interlock 6. Default is 6.

Public Input Members	Data Type	Description
Cfg_eType07	SINT	Enumerated type of Interlock 7. Default is 6.
Cfg_eType08	SINT	Enumerated type of Interlock 8. Default is 6.
Cfg_eType09	SINT	Enumerated type of Interlock 9. Default is 6.
Cfg_eType10	SINT	Enumerated type of Interlock 10. Default is 6.
Cfg_eType11	SINT	Enumerated type of Interlock 11. Default is 6.
Cfg_eType12	SINT	Enumerated type of Interlock 12. Default is 6.
Cfg_eType13	SINT	Enumerated type of Interlock 13. Default is 6.
Cfg_eType14	SINT	Enumerated type of Interlock 14. Default is 6.
Cfg_eType15	SINT	Enumerated type of Interlock 15. Default is 6.
Cfg_eType16	SINT	Enumerated type of Interlock 16. Default is 6.
Cfg_eType17	SINT	Enumerated type of Interlock 17. Default is 6.
Cfg_eType18	SINT	Enumerated type of Interlock 18. Default is 6.
Cfg_eType19	SINT	Enumerated type of Interlock 19. Default is 6.
Cfg_eType20	SINT	Enumerated type of Interlock 20. Default is 6.
Cfg_eType21	SINT	Enumerated type of Interlock 21. Default is 6.
Cfg_eType22	SINT	Enumerated type of Interlock 22. Default is 6.
Cfg_eType23	SINT	Enumerated type of Interlock 23. Default is 6.
Cfg_eType24	SINT	Enumerated type of Interlock 24. Default is 6.
Cfg_eType25	SINT	Enumerated type of Interlock 25. Default is 6.
Cfg_eType26	SINT	Enumerated type of Interlock 26. Default is 6.
Cfg_eType27	SINT	Enumerated type of Interlock 27. Default is 6.
Cfg_eType28	SINT	Enumerated type of Interlock 28. Default is 6.
Cfg_eType29	SINT	Enumerated type of Interlock 29. Default is 6.
Cfg_eType30	SINT	Enumerated type of Interlock 30. Default is 6.
Cfg_eType31	SINT	Enumerated type of Interlock 31. Default is 6.
Cfg_HasType	SINT	Enable selection for types from HMI. Default is 2#1111_1111.

Public Input Members	Data Type	Description
Cfg_TypeDesc00	BOOL	Interlock type 00 description. Default is false.
Cfg_TypeDesc01	BOOL	Interlock type 01 description. Default is false.
Cfg_TypeDesc02	BOOL	Interlock type 02 description. Default is false.
Cfg_TypeDesc03	BOOL	Interlock type 03 description. Default is false.
Cfg_TypeDesc04	BOOL	Interlock type 04 description. Default is false.
Cfg_TypeDesc05	BOOL	Interlock type 05 description. Default is false.
Cfg_TypeDesc06	BOOL	Interlock type 06 description. Default is false.
Cfg_TypeDesc07	BOOL	Interlock type 07description. Default is false.
Cfg_BankID	INT	Bank ID for use with multiple banks (0 to 7). Default is 0.
Cfg_HasMoreObj	BOOL	1 = Tells HMI an object with more information is available. Default is false.
Cfg_CnfrmReqd	SINT	Operator command confirmation required. Represents the type of command confirmation required. 0 = None, 1 = Command confirmation required, 2 = Performer e-signature required, 3 = Performer and approver e-signature required. Default is 0.
PCmd_Reset	BOOL	Program command to reset latched interlocks. Default is false.

Public Output Members	Data Type	Description
EnableOut	BOOL	Enable Output - System Defined Parameter
Out_Reset	BOOL	1 = Reset external devices.
Val_FirstUpBankID	INT	Bank ID number of first up interlock.
Val_FirstUpIndex	INT	Index number of first up interlock.
Sts_Initialized	BOOL	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.
Sts_IntlkOK	BOOL	1 = OK to run, 0 = Stop.
Sts_NBIntlkOK	BOOL	1 = All non-bypassable interlocks OK to run.
Sts_Available	BOOL	1 = Available.
Sts_IntlkTripInh	BOOL	1 = Interlock trip inhibit - stops equipment but does not trip.
Sts_BypActive	BOOL	1 = Interlock bypassing is active (ignore bypassable interlocks).
Sts_FirstUpDetect	BOOL	1 = First up interlock detected.
Sts_BankIDError	BOOL	1 = Error in bank ID's, each bank ID must be unique.
Sts_LatchDefeat	BOOL	1 = Do not latch inputs even if configured for latching.
Sts_RdyReset	BOOL	1 = A latched interlock (returned to OK) is ready to be reset.
Sts_LatchMask	DINT	Latch mask- always latch based on type.
Sts_BypassMask	DINT	Bypass mask- bypass based on type.
Sts_Intlk	DINT	Individual interlock status (1 = stop, 0 = OK).
Sts_FirstOut	DINT	Interlock first out status (bit 1 is first not-OK condition).

Private Input Members	Data Type	Description
HMI_Tab	SINT	Tab to display (FTView ME). Default is 0.
MSet_Bypass	DINT	Individual condition maintenance bypass toggles. Default is 2#0000_0000_0000_0000_0000_0000_0000_0000.
OCmd_Reset	BOOL	Operator command to reset latched interlocks. Default is false.

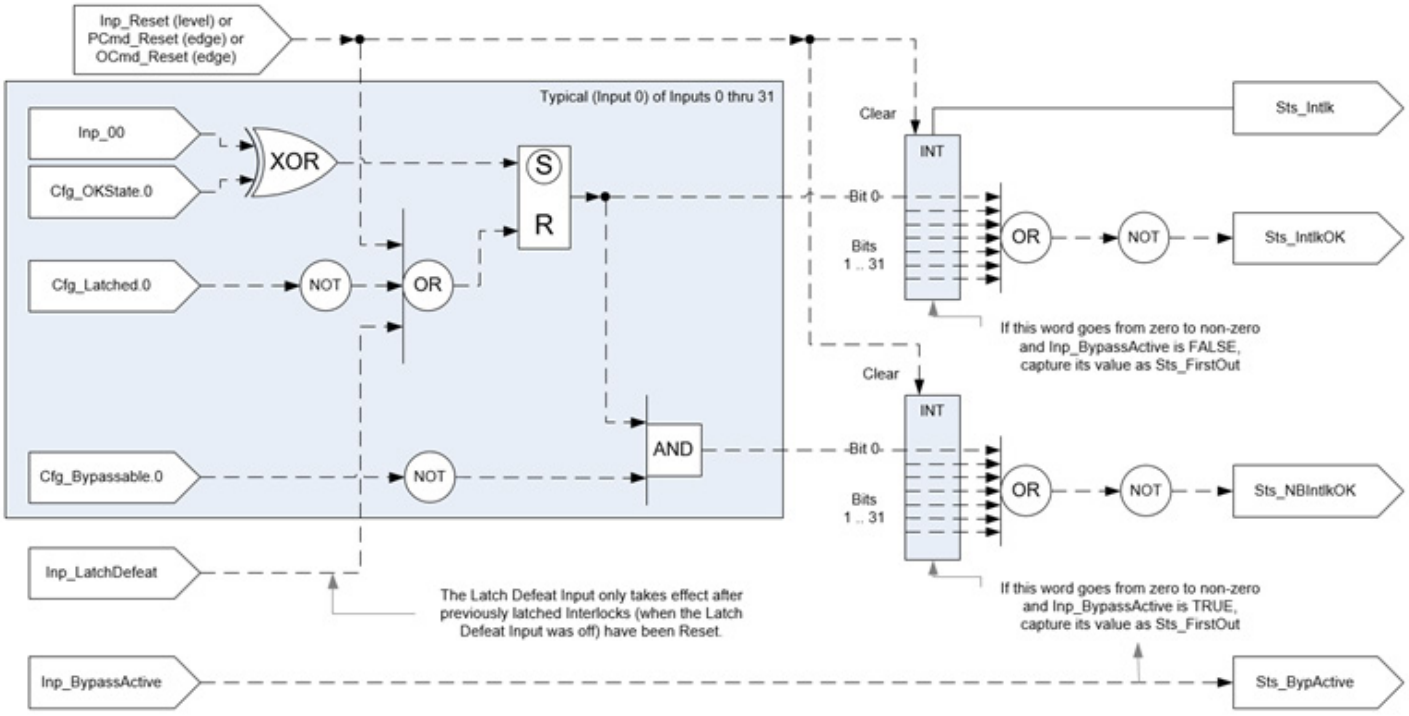
Private Output Members	Data Type	Description
ORdy_Reset	BOOL	1 = Ready for OCmd_Reset (enables HMI button).
Val_BankMap	DINT	Map of interlock banks detected.
Val_BankSts	DINT	Map of interlock banks statuses.

P_INTERLOCK_BANK_STATUS Structure

Members	Data Type	Description
Val_FirstUpIndex	INT	Index number of first up interlock.
Val_FirstUpBankID	INT	Bank ID number of first up interlock.
Val_BankMap	DINT	Map of interlock banks detected.
Val_BankSts	DINT	Map of interlock banks statuses.
Inp_Reset	BOOL	1 = Reset trip and first up.
Inp_BypActive	BOOL	1 = Interlock bypassing is currently active.
Inp_LatchDefeat	BOOL	1 = Do not latch.
Inp_Available	BOOL	1 = Available from preceding equipment.
Sts_BankIDError	BOOL	1 = Duplicate or invalid bank ID.
Sts_IntlkOK	BOOL	Interlocks bypassable interlock status (1 = all interlocks OK to energize).
Sts_NBIntlkOK	BOOL	Interlocks non-bypassable interlock status (1=all non-bypassable interlocks OK to energize).
Sts_IntlkTripInh	BOOL	1 = Interlock trip inhibit – stops equipment but does not trip.
Sts_Available	BOOL	Availability status.(1 = Available).
Sts_FirstUpDetect	BOOL	1=First up interlock detected.
Sts_RdyReset	BOOL	1=A latched interlock (returned to OK) is ready to be reset.
Sts_PrevIntlkOK	BOOL	Previous interlocks bypassable interlock status (1 = all interlocks OK to energize).
Sts_PrevNBIntlkOK	BOOL	Previous interlocks non-bypassable interlock status (1 = all non-bypassable interlocks OK to energize).

Operation

This diagram illustrates the functionality of the PINTLK instruction:



Configuration of Strings for HMI

Configure strings for HMI faceplates (FT View) and for the Logix Designer configuration dialog box. The strings are set to extended properties of tag items. Configure the strings in Logix Designer only.

- Description
- Label for graphic symbol
- Display Library for HMI Faceplate call-up
- Instruction name
- Area name
- URL link
- Input Conditional Text
- Navigation Path
- Interlock Type
- More Information

Monitor the PINTLK Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Rung-condition-out is cleared to false.
Instruction first run	The instruction executes normally.
Rung-condition-in is false	Rung-condition-out is cleared to false. If this instruction is off-scan, then set the summary interlock OK status bits to false. Only set individual interlock bypasses for conditions that are configured for bypassing. All the MSets for inputs that are NOT bypassable will be cleared.
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	EnableOut is cleared to false.
Instruction first run	The instruction executes normally.
Instruction first scan	See Instruction first run in the Function Block Diagram table.
EnableIn is false	EnableOut is cleared to false. If this instruction is off-scan, then set the summary interlock OK status bits to false. Only set individual interlock bypasses for conditions that are configured for bypassing. All the MSets for inputs that are NOT bypassable will be cleared.
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

In Structured Text, EnableIn is always true during normal scan. The instruction executes when it is in the control path activated by the logic.

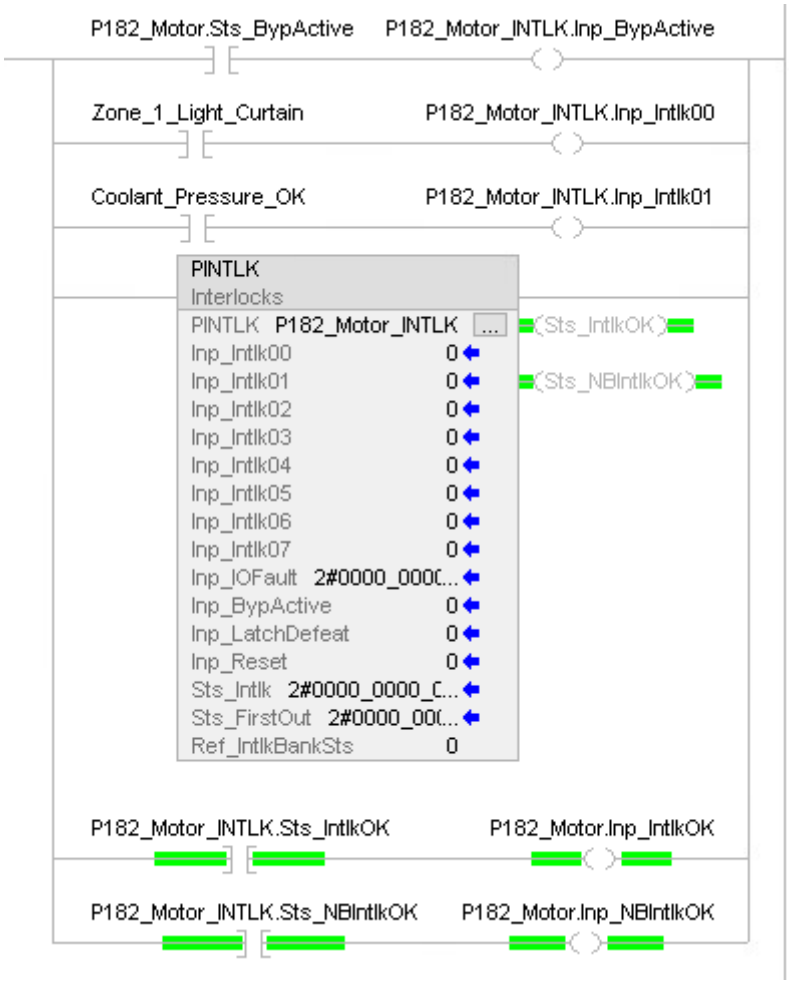
Condition/State	Action Taken
Prescan	See Prescan in the Function Block Diagram table.
Instruction first run	See Instruction first run in the Function Block Diagram table.
EnableIn is true	See EnableIn is true in the Function Block Diagram table.

Condition/State	Action Taken
Postscan	See Postscan in the Function Block Diagram table.

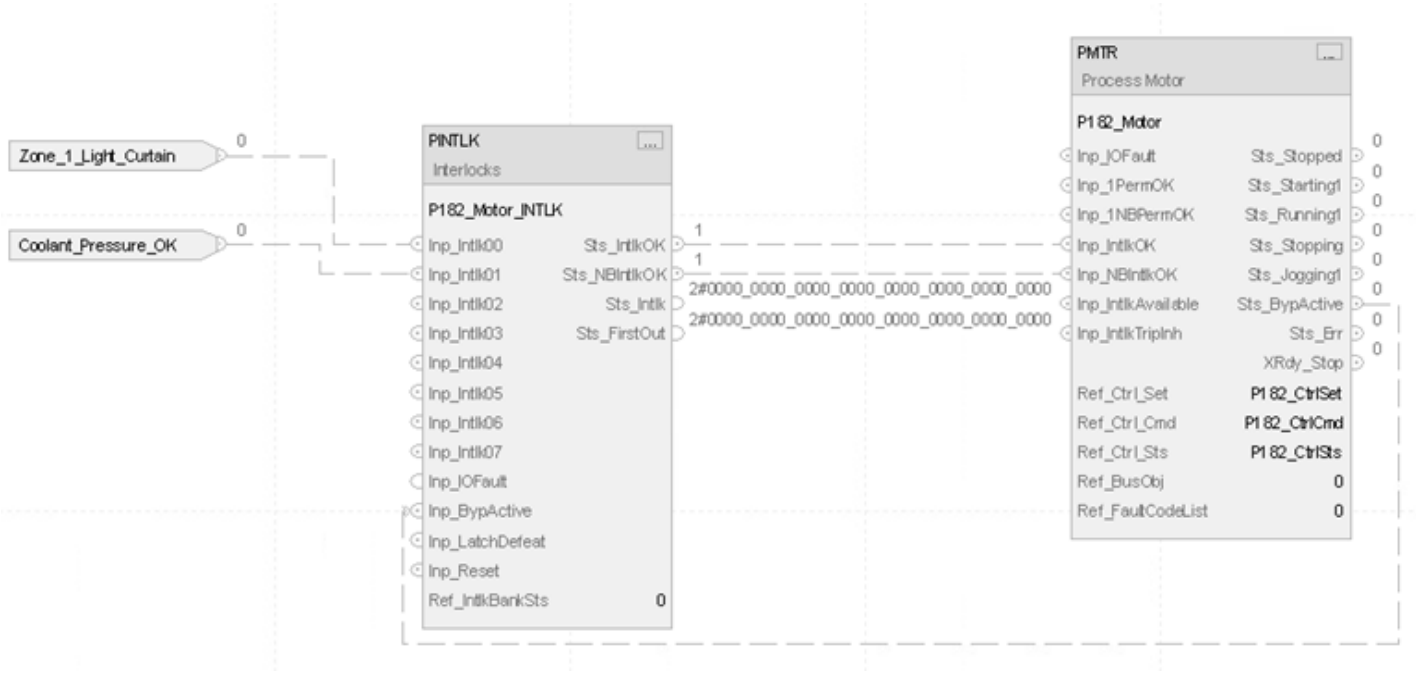
Example

This example uses the PINTLK instruction to concentrate the interlock conditions that allow the functioning of the refiner plates that are used for grinding wood as part of the pulp manufacturing process.

Ladder Diagram



Function Block Diagram



Structured Text

```
P182_Motor_INTLK.Inp_BypActive := P182_Motor.Sts_BypActive;
P182_Motor_INTLK.Inp_Intlk00 := Zone_1_Light_Curtain;
P182_Motor_INTLK.Inp_Intlk01 := Coolant_Pressure_OK;
PINTLK(P182_Motor_INTLK, 0);
P182_Motor.Inp_IntlkOK := P182_Motor_INTLK.Sts_IntlkOK;
P182_Motor.Inp_NBIntlkOK := P182_Motor_INTLK.Sts_NBIntlkOK;
PMTR(P182_Motor,P182_CtrlSet,P182_CtrlCmd,P182_CtrlSts, 0, 0);
```

See also

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Process Lead Lag Standby Motor Group (PLLS)

This information applies to the ControlLogix 5380P and 5580P controllers.

The Process Lead Lag Standby Motor Group (PLLS) instruction provides control of a parallel group of motors, such as a set of pumps with a common intake source and discharge destination. The number of motors to run

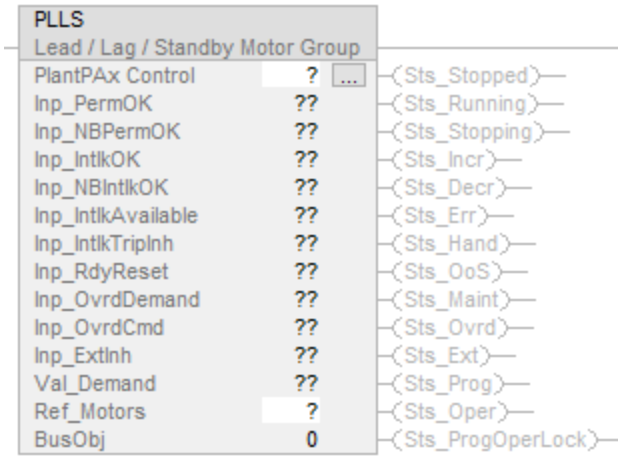
depends on the demand on the system. The group can be configured to consist of as few as two or as many as 30 motors. The minimum demand can be set as low as 0, so that all motors are stopped at minimum demand. The maximum demand can be set as high as the number of pumps in the group.

Use the PLLS instruction to:

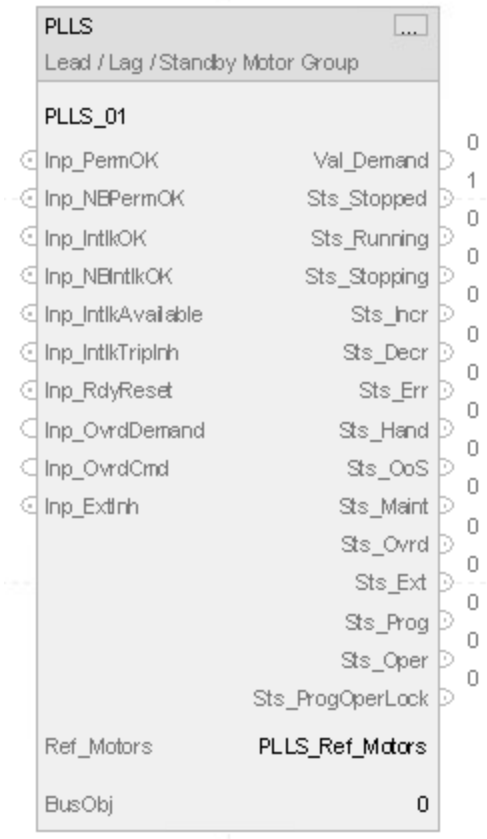
- Control and monitor a group of 2 to 30 motors.
- Start and stop a group using Operator, Program, and Override capability.
- Allow the Operator or Program to enter a demand (the number of motors to run).
- Configure maximum demand (1 to number of motors in group).
- Configure minimum demand (0 to maximum demand).
- Configure stopping the last started motor or the first started motor (first-on-last-off or last-on-last-off).
- Configure delay between starts and configure delay between stops.
- Use start and stop commands to start or stop the motors as a group. The delay between starts or stops can be configured to sequence the motors.
- Start or stop motors as required to meet the entered demand.
- Identify (and optionally alarm) when there are not enough motors available to start (in Program Mode and ready to run) to meet the given demand.
- Identify (and optionally alarm) when there are not enough motors available to stop (in Program Mode and ready to stop) to meet the given demand.
- Ability to rotate the list of motors (demote the lead, promote the others).
- Monitor Permissive conditions to allow starting the motor group.
- Monitor Interlock conditions to stop or prevent starting the motor group.
- Alarm if interlock conditions cause the group to be stopped.
- Use HMI breadcrumbs for Alarm Inhibited, Bad Configuration, Not Ready, and Maintenance Bypass Active.
- Use Available status in automation logic to determine whether the motor group can be controlled by other objects.

Available Languages

Ladder Diagram



Function Block Diagram



Structured Text

PLLS (PLLS tag, Ref_Motors tag, BusObj tag);

Operands

IMPORTANT Unexpected operation may occur if:
- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

This table describes the PLLS configuration operands.

Operand	Type	Format	Description
PlantPAx Control	P_LEAD_LAG_STANDBY	tag	Data structure required for proper operation of instruction.
Ref_Motors	P_LEAD_LAG_STANDBY_MOTOR	tag	Motor interface array.
BusObj	BUS_OBJ	tag	Bus component.

P_LEAD_LAG_STANDBY Structure

Use InOut parameters to link the instruction to external tags that contain necessary data for the instruction to operate. These external tags must be of the data type shown.

Public members are standard, visible tag members that are programmatically accessible. Private, hidden members are used in HMI faceplates and are not programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
EnableIn	BOOL	Not Visible	Not Required	Input	Enable Input. Ladder Diagram. Corresponds to the rung-condition-in. Default is true.
Inp_InitializeReq	BOOL	Not Visible	Not Required	Input	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request when reinitialization is needed. The instruction clears this operand automatically. Default is true.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Inp_OwnerCmd	DINT	Not Visible	Not Required	Input	Owner device command. 0 = None, Inp_OwnerCmd.10 = Operator Lock, Inp_OwnerCmd.11 = Operator Unlock, Inp_OwnerCmd.12 = Program Lock, Inp_OwnerCmd.13 = Program Unlock, Inp_OwnerCmd.14 = Acquire Maintenance, Inp_OwnerCmd.15 = Release Maintenance, Inp_OwnerCmd.16 = Acquire External, Inp_OwnerCmd.17 = Release External. Default is 0.
Inp_PermOK	BOOL	Visible	Not Required	Input	1 = Start permissives OK, group can start. Default is true.
Inp_NBPermOK	BOOL	Visible	Not Required	Input	1 = Non-bypassable start permissives OK, group can start. Default is true.
Inp_IntlkOK	BOOL	Visible	Not Required	Input	1 = Interlocks OK, group can start/run. Default is true.
Inp_NBIntlkOK	BOOL	Visible	Not Required	Input	1 = Non-bypassable interlocks OK, group can start/run. Default is true.
Inp_IntlkAvailable	BOOL	Visible	Not Required	Input	1 = Interlock availability OK. Default is false.
Inp_IntlkTripInh	BOOL	Visible	Not Required	Input	1 = Inhibit interlock trip status. Default is false.
Inp_RdyReset	BOOL	Visible	Not Required	Input	1 = Related object, reset by this object, is ready to be reset. Default is false.
Inp_Hand	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is false.
Inp_Ovrd	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is false.
Inp_OvrdDemand	DINT	Visible	Not Required	Input	Override Mode setting for number of motors to run (MinDemand..MaxDemand). Default is 0.
Inp_OvrdCmd	DINT	Visible	Not Required	Input	Override Mode Command: 0 = None, 1 = Stop Group, 2 = Start Group, 3 = Rotate Assignments. Default is 0.
Inp_ExtInh	BOOL	Visible	Not Required	Input	Control / command source selection. Default is false.
Inp_Reset	BOOL	Not Visible	Not Required	Input	1 = Reset all fault conditions and latched alarms. Default is false.
Cfg_AllowDisable	BOOL	Not Visible	Not Required	Input	1 = Allow maintenance to disable alarms. Default is true.
Cfg_AllowShelve	BOOL	Not Visible	Not Required	Input	1 = Allow operator to shelve alarms. Default is true.
Cfg_NumMotors	DINT	Not Visible	Not Required	Input	Number of motors in this Lead / Lag / Standby Group. Valid = 2 to 30. Default is 3.
Cfg_MaxDemand	DINT	Not Visible	Not Required	Input	Maximum number of motors to run. Valid = 1 to Cfg_NumMotors. Default is 2.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_MinDemand	DINT	Not Visible	Not Required	Input	Minimum number of motors to run. Valid = 0 to Cfg_MaxDemand. Default is 0.
Cfg_StartDly	REAL	Not Visible	Not Required	Input	Time (seconds) after start or stop until next start is allowed (0..2M seconds). Valid = 0.0 to 2147483.0. Default is 10.0.
Cfg_StopDly	REAL	Not Visible	Not Required	Input	Time (seconds) after start or stop until next stop is allowed (0..2M seconds). Valid = 0.0 to 2147483.0. Default is 10.0.
Cfg_FirstOnFirstOff	BOOL	Not Visible	Not Required	Input	1 = First started is first stopped, 0 = First started is last stopped. Default is false.
Cfg_AllowRotate	BOOL	Not Visible	Not Required	Input	1 = Allow rotate (cycle lead) command to rotate motor assignments. Default is true.
Cfg_RotateOnStop	BOOL	Not Visible	Not Required	Input	1 = Rotate (cycle lead to end of list) upon stopping all motors. Default is true.
Cfg_HasPermObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI an object is connected to permissive inputs. Default is false.
Cfg_HasIntlkObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI an object is connected to interlock inputs. Default is false.
Cfg_HasMoreObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI an object with more information is available. Default is false.
Cfg_HasNav01	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #1. Default is false.
Cfg_HasNav02	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #2. Default is false.
Cfg_HasNav03	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #3. Default is false.
Cfg_HasNav04	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #4. Default is false.
Cfg_HasNav05	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #5. Default is false.
Cfg_HasNav06	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #6. Default is false.
Cfg_HasNav07	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #7. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_HasNav08	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #8. Default is false.
Cfg_HasNav09	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #9. Default is false.
Cfg_HasNav10	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #10. Default is false.
Cfg_HasNav11	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #11. Default is false.
Cfg_HasNav12	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #12. Default is false.
Cfg_HasNav13	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #13. Default is false.
Cfg_HasNav14	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #14. Default is false.
Cfg_HasNav15	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #15. Default is false.
Cfg_HasNav16	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #16. Default is false.
Cfg_HasNav17	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #17. Default is false.
Cfg_HasNav18	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #18. Default is false.
Cfg_HasNav19	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #19. Default is false.
Cfg_HasNav20	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #20. Default is false.
Cfg_HasNav21	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #21. Default is false.
Cfg_HasNav22	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #22. Default is false.
Cfg_HasNav23	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #23. Default is false.
Cfg_HasNav24	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #24. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_HasNav25	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #25. Default is false.
Cfg_HasNav26	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #26. Default is false.
Cfg_HasNav27	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #27. Default is false.
Cfg_HasNav28	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #28. Default is false.
Cfg_HasNav29	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #29. Default is false.
Cfg_HasNav30	BOOL	Not Visible	Not Required	Input	1 = enable a button on the HMI that is used to call up the faceplate for motor #30. Default is false.
Cfg_SetTrack	BOOL	Not Visible	Not Required	Input	1 = When the owner is program the operator settings track the program settings. When the owner is operator the program settings track the operator settings, and the virtual inputs match the output values (transitions are bumpless), 0 = No tracking. Default is false.
Cfg_SetTrackOvrdHand	BOOL	Not Visible	Not Required	Input	1 = program/operator settings track override/hand speed reference. Default is false.
Cfg_OperStopPrio	BOOL	Not Visible	Not Required	Input	1 = OCmd_Stop any time, 0 = OCmd_Stop only when operator selected. Default is false.
Cfg_ExtStopPrio	BOOL	Not Visible	Not Required	Input	1 = XCmd_Stop any time, 0 = XCmd_Stop only when external selected. Default is false.
Cfg_OCmdResets	BOOL	Not Visible	Not Required	Input	1 = New group OCmd resets shed latches and cleared alarms, 0 = OCmdReset required. Default is false.
Cfg_XCmdResets	BOOL	Not Visible	Not Required	Input	1 = New group XCmd resets shed latches and cleared alarms, 0 = OCmdReset required. Default is false.
Cfg_OvrdPermIntlk	BOOL	Not Visible	Not Required	Input	1 = Override ignores bypassable permissives/interlocks, 0 = Always use permissives/interlocks. Default is false.
Cfg_CnfrmReqd	SINT	Not Visible	Not Required	Input	Operator command confirmation required. Represents the type of command confirmation required. 0 = None, 1 = Command confirmation required, 2 = Performer e-signature required, 3 = Performer and approver e-signature required. Default is 0.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
PSet_Demand	DINT	Not Visible	Not Required	Input	Program setting for number of motors to run (MinDemand...MaxDemand). Default is 0.
PSet_Owner	DINT	Not Visible	Not Required	Input	Program owner request ID (non-zero) or release (zero). Default is 0.
XSet_Demand	DINT	Not Visible	Not Required	Input	External setting for number of motors to run (MinDemand...MaxDemand). Default is 0.
PCmd_Start	BOOL	Not Visible	Not Required	Input	Program command to start motor group. Default is false.
PCmd_Stop	BOOL	Not Visible	Not Required	Input	Program command to stop motor group. Default is false.
PCmd_Rotate	BOOL	Not Visible	Not Required	Input	Program command to rotate assignments (cycle lead to end of list). Default is false.
PCmd_Prog	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is false.
PCmd_Oper	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is false.
PCmd_Lock	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is false.
PCmd_Unlock	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is false.
PCmd_Normal	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is false.
PCmd_Reset	BOOL	Not Visible	Not Required	Input	Program command to reset all alarms requiring reset. Default is false.
XCmd_Start	BOOL	Not Visible	Not Required	Input	External command to start motor group. Default is false.
XCmd_Stop	BOOL	Not Visible	Not Required	Input	External command to stop motor group. Default is false.
XCmd_Rotate	BOOL	Not Visible	Not Required	Input	External command to rotate assignments (cycle lead to end of list). Default is false.
XCmd_Reset	BOOL	Not Visible	Not Required	Input	External command to reset all alarms requiring reset. The instruction clears this operand automatically. Default is false.
XCmd_ResetAckAll	BOOL	Not Visible	Not Required	Input	External command to acknowledge and reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.
Cfg_HasOper	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is true.
Cfg_HasOperLocked	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is true.
Cfg_HasProg	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is true.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_HasProgLocked	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is true.
Cfg_HasExt	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is false.
Cfg_HasMaint	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is true.
Cfg_HasMaintOoS	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is true.
Cfg_OvrdOverLock	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is true.
Cfg_ExtOverLock	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is false.
Cfg_ProgPwrUp	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is false.
Cfg_ProgNormal	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is false.
Cfg_PCmdPriority	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is false.
Cfg_PCmdProgAsLevel	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is false.
Cfg_PCmdLockAsLevel	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is false.
Cfg_ExtAcqAsLevel	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is false.
XCmd_Acq	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is false.
XCmd_Rel	BOOL	Not Visible	Not Required	Input	Control / command source selection. Default is false.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
EnableOut	BOOL	Not Visible	Not Required	Output	Enable Output. This output state always reflects EnableIn input state.
Val_Demand	DINT	Visible	Not Required	Output	Number of motors requested to run.
Val_RotateRank	DINT	Not Visible	Not Required	Output	Motor rank (0 = Lead, etc.) which will be demoted on rotate.
Val_RotateID	DINT	Not Visible	Not Required	Output	Motor number which will be demoted on rotate.
Sts_eCmd	SINT	Not Visible	Not Required	Output	Group command 0 = None, 1 = Stop, 2 = Start.
Sts_Fdbk	SINT	Not Visible	Not Required	Output	Group Feedback 0...31 = Number of motors actually running.
Sts_eSts	INT	Not Visible	Not Required	Output	Group confirmed status: 0 = ?, 1 = Stopped, 2 = Running, 3 = Stopping, 4 = Decreasing, 5 = Increasing.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_eFault	INT	Not Visible	Not Required	Output	Group fault status: 0 = None, 1 = Configuration error, 12 = Fail to start, 13 = Fail to stop.
Sts_eNotifyAll	SINT	Not Visible	Not Required	Output	Highest alarm priority and acknowledge status this object + motors (enumeration).
Sts_Initialized	BOOL	Not Visible	Not Required	Output	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.
Sts_Stopped	BOOL	Visible	Not Required	Output	1 = Motor group requested to stop and all motors confirmed stopped.
Sts_Running	BOOL	Visible	Not Required	Output	1 = Motor group requested to run.
Sts_Stopping	BOOL	Visible	Not Required	Output	1 = Motor group requested to stop and not all motors confirmed stopped.
Sts_Incr	BOOL	Visible	Not Required	Output	1 = Group is starting motors in sequence to get up to demand.
Sts_Decr	BOOL	Visible	Not Required	Output	1 = Group is stopping motors in sequence to get down to demand.
Sts_Available	BOOL	Not Visible	Not Required	Output	1 = Group available for control by automation (program).
Sts_IntlkAvailable	BOOL	Not Visible	Not Required	Output	1 = Device can be acquired by program and is available for start/stop control when interlocks are OK.
Sts_Bypass	BOOL	Not Visible	Not Required	Output	1 = Bypassable interlocks and permissives are bypassed.
Sts_BypActive	BOOL	Not Visible	Not Required	Output	1 = Interlock bypassing active (bypassed or maintenance).
Sts_NotRdy	BOOL	Not Visible	Not Required	Output	1 = Group is not ready, for HMI use hidden detail bits (Sts_Nrdyxxx) for reason.
Sts_NrdyCfgErr	BOOL	Not Visible	Not Required	Output	1 = Group is not ready: Configuration error.
Sts_NrdyIntlk	BOOL	Not Visible	Not Required	Output	1 = Group is not ready: Interlock not OK.
Sts_NrdyOoS	BOOL	Not Visible	Not Required	Output	1 = Group is not ready: Group is out of service.
Sts_NrdyPrioStop	BOOL	Not Visible	Not Required	Output	1 = Group is not ready: Operator/external priority stop requires reset.
Sts_NrdyPerm	BOOL	Not Visible	Not Required	Output	1 = Group is not ready: Permissive not OK.
Sts_MaintByp	BOOL	Not Visible	Not Required	Output	1 = A maintenance bypass function is active.
Sts_Alm	BOOL	Not Visible	Not Required	Output	1 = An alarm is active.
Sts_AlmInh	BOOL	Not Visible	Not Required	Output	1 = One or more alarms shelved, disabled, or suppressed.
Sts_Err	BOOL	Visible	Not Required	Output	1 = Error in configuration: See detail bits (Sts_Errxxx) for reason.
Sts_ErrStartDly	BOOL	Not Visible	Not Required	Output	1 = Error in configuration: Start check timer preset (use 0.0 to 2147483.0).
Sts_ErrStopDly	BOOL	Not Visible	Not Required	Output	1 = Error in configuration: Stop check timer preset (use 0.0 to 2147483.0).
Sts_ErrAlm	BOOL	Not Visible	Not Required	Output	1 = Error in configuration: Alarm throttle time or severity.
Val_Owner	DINT	Not Visible	Not Required	Output	Current object owner ID (0 = not owned).
Sts_MotorAvailable	DINT	Not Visible	Not Required	Output	Set bits indicate which motors are available for program control.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_MotorStopped	DINT	Not Visible	Not Required	Output	Set bits indicate which motors are confirmed stopped.
Sts_MotorStarting	DINT	Not Visible	Not Required	Output	Set bits indicate which motors are starting.
Sts_MotorRunning	DINT	Not Visible	Not Required	Output	Set bits indicate which motors are confirmed running.
Sts_MotorStopping	DINT	Not Visible	Not Required	Output	Set bits indicate which motors are confirmed stopped.
Sts_Hand	BOOL	Visible	Not Required	Output	Control / command source selection.
Sts_OoS	BOOL	Visible	Not Required	Output	Control / command source selection.
Sts_Maint	BOOL	Visible	Not Required	Output	Control / command source selection.
Sts_Ovrd	BOOL	Visible	Not Required	Output	Control / command source selection.
Sts_Ext	BOOL	Visible	Not Required	Output	Control / command source selection.
Sts_Prog	BOOL	Visible	Not Required	Output	Control / command source selection.
Sts_ProgLocked	BOOL	Not Visible	Not Required	Output	Control / command source selection.
Sts_Oper	BOOL	Visible	Not Required	Output	Control / command source selection.
Sts_OperLocked	BOOL	Not Visible	Not Required	Output	Control / command source selection.
Sts_Normal	BOOL	Not Visible	Not Required	Output	Control / command source selection.
Sts_ExtReqInh	BOOL	Not Visible	Not Required	Output	Control / command source selection.
Sts_ProgReqInh	BOOL	Not Visible	Not Required	Output	Control / command source selection.
Sts_MAcqRcvd	BOOL	Not Visible	Not Required	Output	Control / command source selection.
Sts_CantStart	BOOL	Not Visible	Not Required	Output	1 = Motor failed to start (one-shot).
Sts_CantStop	BOOL	Not Visible	Not Required	Output	1 = Motor failed to stop.
Sts_IntlkTrip	BOOL	Not Visible	Not Required	Output	1 = Group stopped by an interlock NOT OK (one-shot).
Sts_RdyReset	BOOL	Not Visible	Not Required	Output	1 = A latched alarm or shed condition is ready to be reset.
Sts_RdyAck	BOOL	Not Visible	Not Required	Output	1 = An alarm is ready to be acknowledged.
Sts_UnackAlmCount	DINT	Not Visible	Not Required	Output	Count of unacknowledged alarms.
Out_Reset	BOOL	Not Visible	Not Required	Output	1 = Reset command has been received and accepted.
Out_OwnerSts	DINT	Not Visible	Not Required	Output	Status of command source, owner command handshake and ready status. 0 = None, .10 = Operator Lock, .11 = Operator Unlock, .12 = Program Lock, .13 = Program Unlock, .14 = Acquire Maintenance, .15 = Release Maintenance, .16 = Acquire External, .17 = Release External, .18 = Has Maintenance, .19 = External Override Lock, .20 = Has External, .21 = Has Operator, .22 = Has Program, .30 = Not Ready.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_eSrc	INT	Not Visible	Not Required	Output	The current command source enumerated values: 0 = Logic not in use, 4 = Hand, 8 = Maintenance, 16 = Override, 32 = Program, 33 = Program locked, 34 = Program by default (Normal), 64 = Operator, 65 = Operator locked, 66 = Operator by default (Normal), 128 = Maintenance Out of Service, 129 = Programmed Out of Service (rung false), 256 = External.
Sts_bSrc	INT	Not Visible	Not Required	Output	Control / command source selection.
Sts_ProgOperSel	BOOL	Not Visible	Not Required	Output	Control / command source selection.
Sts_ProgOperLock	BOOL	Visible	Not Required	Output	Control / command source selection.
XRdy_Acq	BOOL	Not Visible	Not Required	Output	Control / command source selection.
XRdy_Rel	BOOL	Not Visible	Not Required	Output	Control / command source selection.
XRdy_Reset	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Reset, enable HMI button.
XRdy_ResetAckAll	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_ResetAckAll, enable HMI button.
XRdy_Stop	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Stop, enable HMI button.
XRdy_Start	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Start, enable HMI button.
XRdy_Rotate	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Rotate, enable HMI button.

Private Input Members	Data Type	Description
HMI_BusObjIndex	DINT	HMI bus object index. Default is 0.
MCmd_Acq	BOOL	Maintenance command to acquire ownership (Operator/Program/External/Override to Maintenance). The instruction clears this operand automatically. Default is false.
MCmd_Bypass	BOOL	Maintenance command to bypass all bypassable interlocks and permissives. Default is false.
MCmd_Check	BOOL	Maintenance command to check (not bypass) all interlocks and permissives. Default is false.
MCmd_IS	BOOL	Maintenance command to select In Service. The instruction clears this operand automatically. Default is false.
MCmd_OoS	BOOL	Maintenance command to select Out of Service. The instruction clears this operand automatically. Default is false.
MCmd_Rel	BOOL	Maintenance command to release ownership (Maintenance to Operator/Program/External/Override). The instruction clears this operand automatically. Default is false.
MSet_MotorOoS	DINT	Set bits indicate which motors have been taken out of service by maintenance. Default is 2#0000_0000_0000_0000_0000_0000_0000_0000.

Private Input Members	Data Type	Description
OCmd_CmdCncl	BOOL	Operator command to cancel command request. The instruction clears this operand automatically. Default is false.
OCmd_Lock	BOOL	Operator command to lock Operator (disallow Program). The instruction clears this operand automatically. Default is false.
OCmd_Normal	BOOL	Operator command to select Normal (Operator or Program). The instruction clears this operand automatically. Default is false.
OCmd_Oper	BOOL	Operator command to select Operator (Program to Operator). The instruction clears this operand automatically. Default is false.
OCmd_Prog	BOOL	Operator command to select Program (Operator to Program). The instruction clears this operand automatically. Default is false.
OCmd_Reset	BOOL	Operator command to reset all alarms requiring reset. Default is false.
OCmd_ResetAckAll	BOOL	Operator command to reset all alarms and latched shed conditions. Default is false.
OCmd_ResetPrefs	BOOL	Operator command to reset all motor preferences to 0. Default is false.
OCmd_Rotate	BOOL	Operator command to rotate assignments (cycle lead to end of list). Default is false.
OCmd_SetPrefs	BOOL	Operator command to set motor preferences. Default is false.
OCmd_Start	BOOL	Operator command to start motor group. Default is false.
OCmd_Stop	BOOL	Operator command to stop motor group. Default is false.
OCmd_Unlock	BOOL	Operator command to unlock / release (allow Program to acquire) ownership. The instruction clears this operand automatically. Default is false.
OSet_Demand	DINT	Operator setting for number of motors to run (MinDemand.. MaxDemand). Default is 0.

Private Output Members	Data Type	Description
MRdy_Acq	BOOL	1 = Ready for MCmd_Acq, enable HMI button.
MRdy_Bypass	BOOL	1 = Ready for MCmd_Bypass (enables HMI button).
MRdy_Check	BOOL	1 = Ready for MCmd_Check (enables HMI button).
MRdy_IS	BOOL	1 = Ready for MCmd_IS, enable HMI button.
MRdy_OoS	BOOL	1 = Ready for MCmd_OoS, enable HMI button.
MRdy_Rel	BOOL	1 = Ready for MCmd_Rel, enable HMI button.
ORdy_Demand	BOOL	1 = Ready for OSet_Demand (enables numeric entry).
ORdy_Lock	BOOL	1 = Ready for OCmd_Lock, enable HMI button.
ORdy_Normal	BOOL	1 = Ready for OCmd_Normal, enable HMI button.
ORdy_Oper	BOOL	1 = Ready for OCmd_Oper, enable HMI button.
ORdy_Prog	BOOL	1 = Ready for OCmd_Prog, enable HMI button.
ORdy_Reset	BOOL	1 = Ready for OCmd_Reset (enables HMI button).
ORdy_ResetAckAll	BOOL	1 = Ready for OCmd_ResetAckAll (enables HMI button).
ORdy_Rotate	BOOL	1 = Ready for OCmd_Rotate (enables HMI button).
ORdy_Start	BOOL	1 = Ready for OCmd_Start (enables HMI button).

Private Output Members	Data Type	Description
ORdy_Stop	BOOL	1 = Ready for OCmd_Stop (enables HMI button).
ORdy_Unlock	BOOL	1 = Ready for OCmd_Unlock, enable HMI button.
Sts_bStsList	SINT[32]	Rank list of motor status: [0]=lead, [1]=lag, etc.; .0 = Avaiable, .1 = Stopped, .2 = Starting, .3 = Running, .4 = Stopping, .5 = Out of Service (Maint).
Sts_eNotify	SINT	Current alarm level and acknowledgement (enumeration).
Sts_eNotifyCantStart	SINT	Current alarm level and acknowledgement (enumeration).
Sts_eNotifyCantStop	SINT	Current alarm level and acknowledgement (enumeration).
Sts_eNotifyIntlkTrip	SINT	Current alarm level and acknowledgement (enumeration).
Val_PrefList	SINT[32]	Rank list of motor preferences: [0] = lead, [1] = lag, etc...
Val_PrioList	SINT[32]	Rank list of motor priorities: [0] = lead, [1] = lag, etc...
Val_RankList	SINT[32]	Rank list of motor numbers: [0] = lead, [1] = lag, etc...
Val_UsrList	INT[32]	Rank list of user sort criteria: [0] = lead, [1] = lag, etc...

Public InOut Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Ref_Motors	P_LEAD_LAG_STANDBY_MOTOR[30]	Visible	Required	InOut	Motor interface array (link to 2 to 30 motors).
BusObj	BUS_OBJ	Visible	Optional	InOut	Bus component.

Alarms

Discrete tag-based alarms are defined for these members.

Member	Alarm Name	Description
Sts_CantStart	Alm_CantStart	Motor can't start alarm. Raised when there are not enough motors available to start to satisfy the entered demand. Too many motors are faulted or stopped in a mode other than program.
Sts_CantStop	Alm_CantStop	Motor can't stop alarm. Raised when there are not enough motors available to stop to satisfy the entered demand. Too many motors are running in a mode other than program.
Sts_IntlkTrip	Alm_IntlkTrip	Interlock Trip alarm. Raised when the motor is running and an interlock not-OK condition causes the motor to stop. If interlocks are not bypassed, a bypassable interlock or a non-bypassable interlock not-OK condition initiates an interlock trip. If interlocks are bypassed, only a non-bypassable interlock not-OK condition initiates an interlock trip.

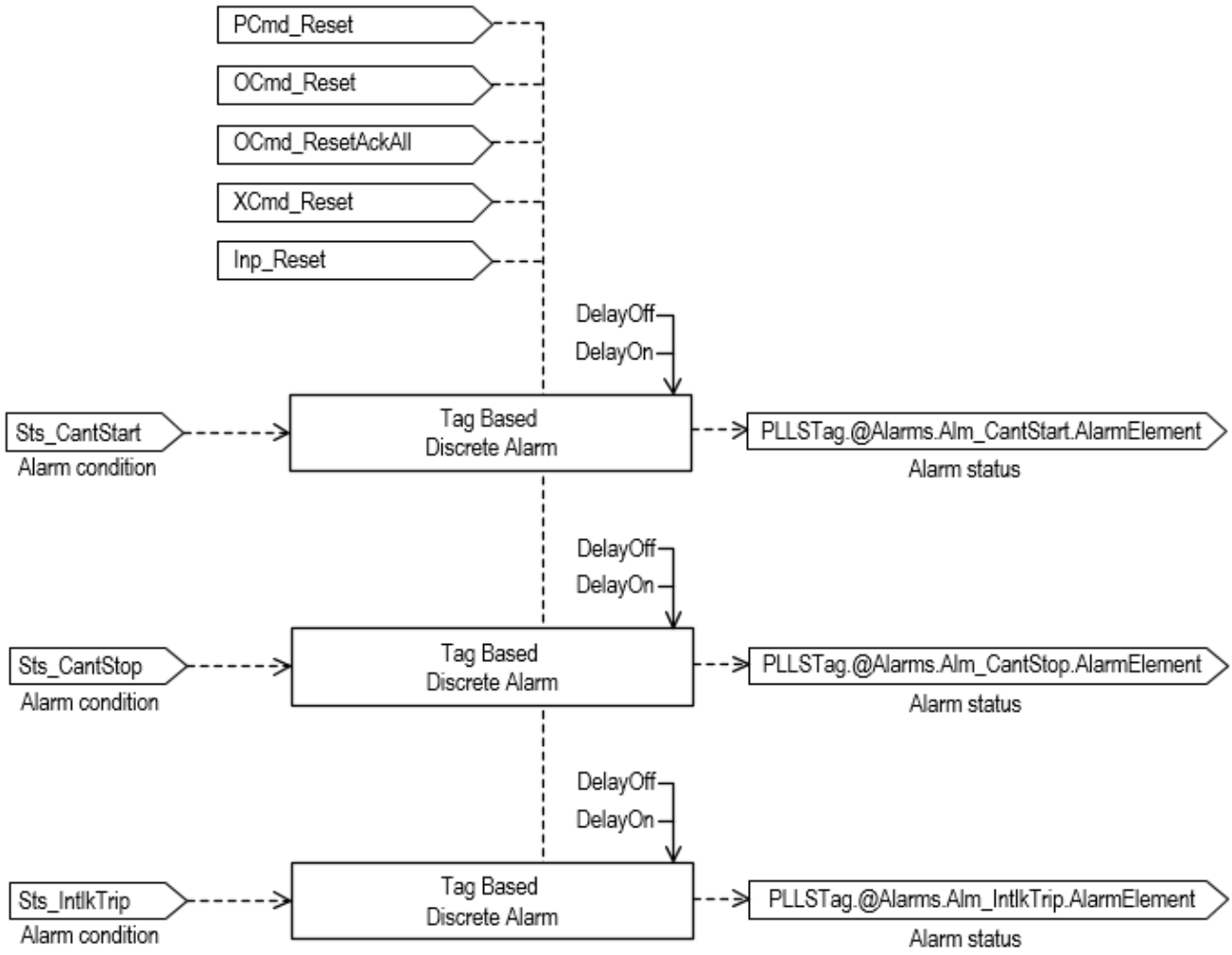
Mark an alarm as used or unused and set standard configuration members of the discrete tag-based alarm. Use this format to access alarm elements:

Tag.@Alarms.AlarmName.AlarmElement

There are Program commands for each Alarm that are available to Acknowledge, Suppress, Unsupress and Unshelve the Alarm. These

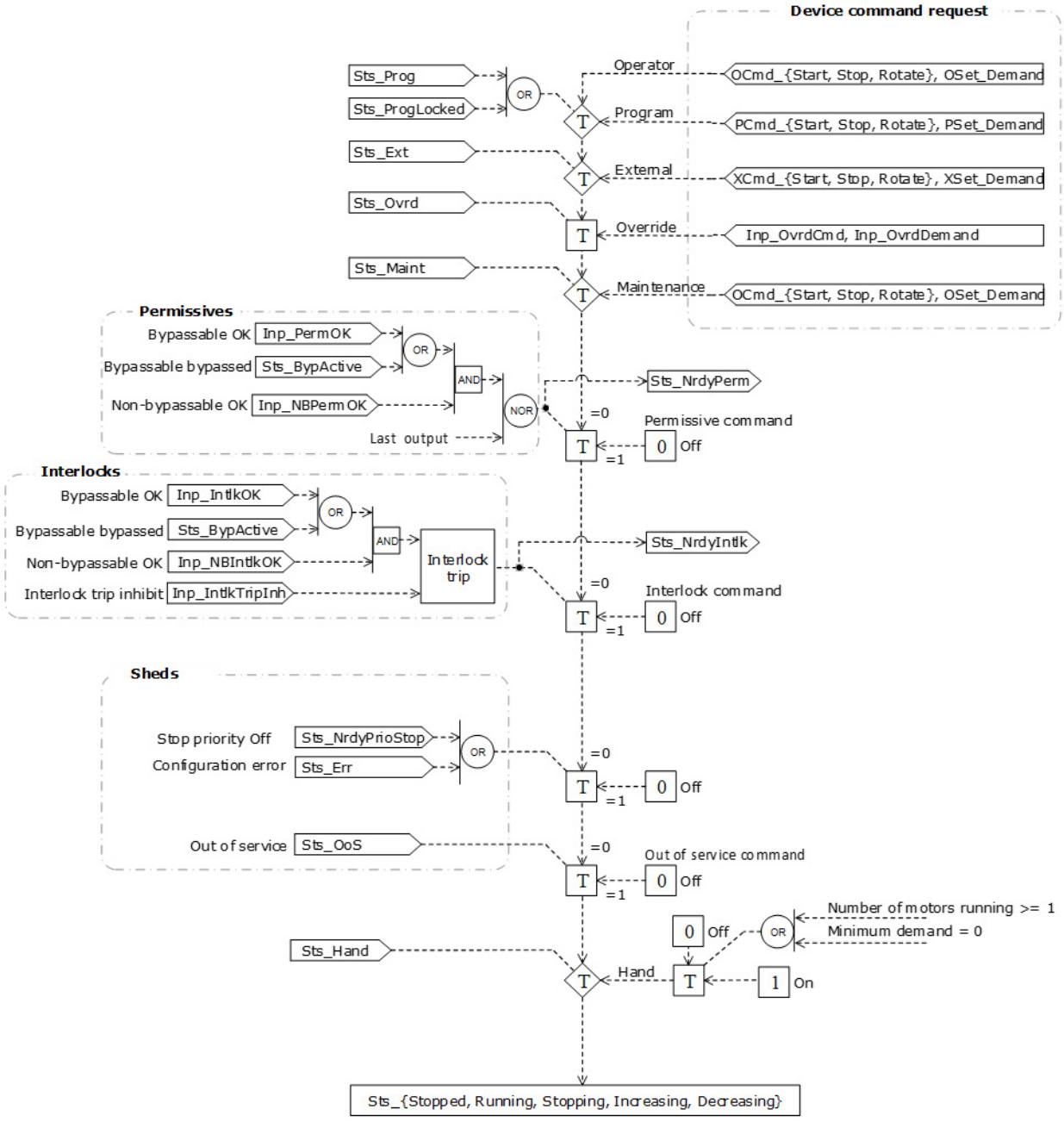
commands are propagated to corresponding commands (ProgAck, ProgSuppress, ProgUnsupress, ProgUnshelve) of the tag-based alarm.

There are Program, Operator, and External commands available that Acknowledge, Reset, Suppress and Unsuppress all alarms of the instruction (Alarm Set) at the same time.



Operation

This diagram illustrates the functionality of the PLLS instruction:



Configuration of Strings for HMI

Configure strings for HMI faceplates (FactoryTalk View) and for the Logix Designer configuration dialog box. The strings are set to extended properties of tag items.

- Description
- Label for graphic symbol
- Display Library for HMI Faceplate call-up
- Instruction name
- Area name
- URL link

- More Information

Implementation

An operator or other logic determines the demand for motors. The PLLS instruction determines which motors to run to meet demand. For the PLLS instruction to start and stop motors in the group, they must be available. A motor is available when it has no faults and is in Program Mode.

The PLLS instruction uses a sorting algorithm to deal with motors that are not available. If a motor is running and not available (perhaps running in Operator Mode), the motor is forced to the top of the sort. If a motor is stopped and not available (perhaps faulted), the motor is forced to the bottom of the sort. The motors that are available to start and stop are controlled to meet the demand. If the demand cannot be met because of unavailable motors, a status/alarm is provided.

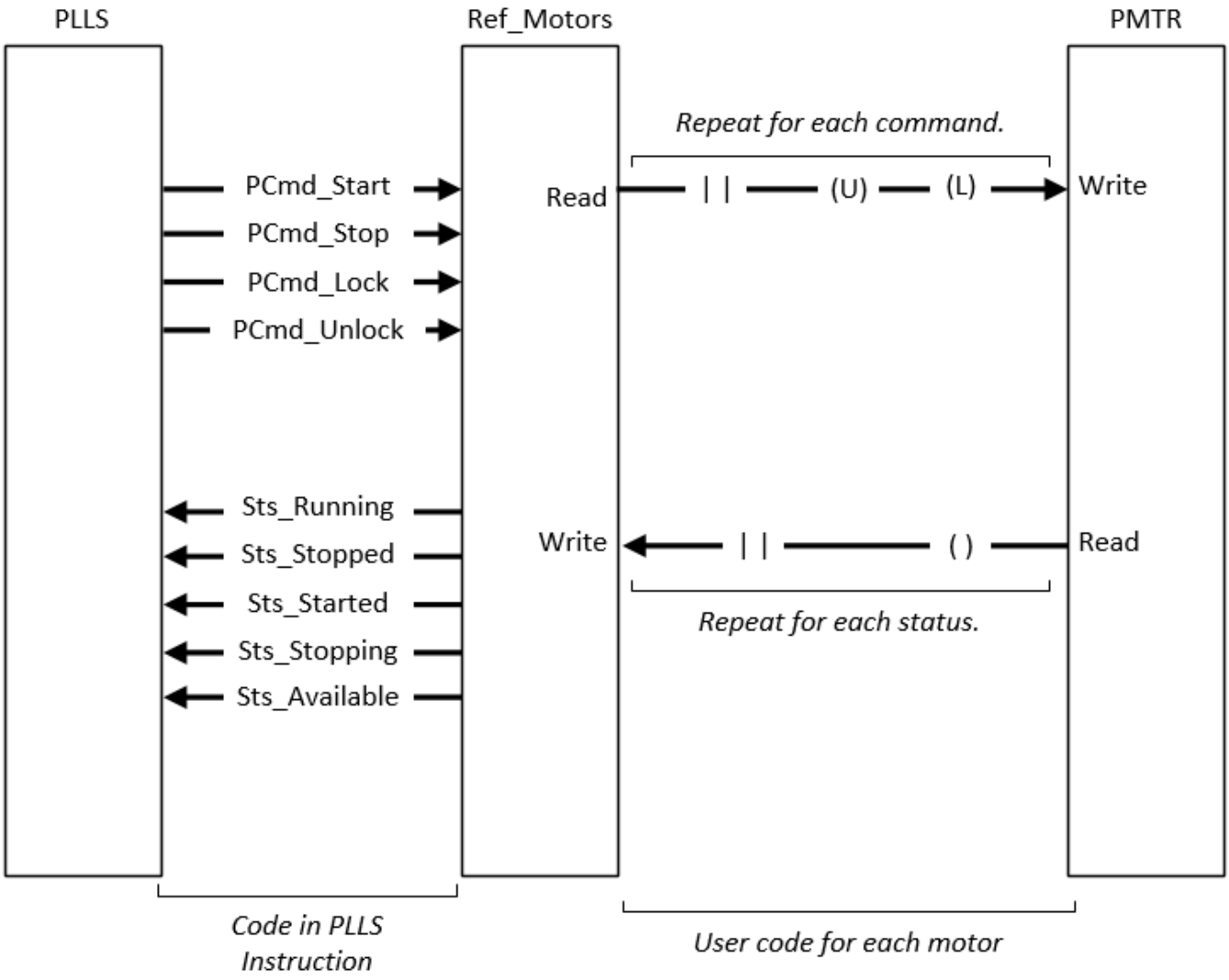
The PLLS instruction uses an array of structures of the type P_LEAD_LAG_STANDBY_Motor to interface to the motors. Each interface element in the array provides the signals that are required between the PLLS instruction and one motor. Configuration data for the motor is also provided in the array. This data includes Priority and Preference values that can be used to affect the sorting of the motors. A Maintenance out of service flag that removes a motor from consideration in the sort is also included. The interface also includes a user sort value that can be used, for example, to push motors up or down the sort based on accumulated runtime or other criteria.

P_LEAD_LAG_STANDBY_MOTOR Array Member Content

This table describes the array members.

Members	Data Type	Description
Inp_OtherSel	DINT	Other motor selection criteria (0...255) (input to PLLS).
Inp_Demote	BOOL	Demote this motor to bottom of list (for example, on high runtime) (input to PLLS).
Cfg_Prio	DINT	Motor priority in list (0...31 -- if unused, set to 0).
OSet_Pref	DINT	Operator setting for motor preference in list (0 to 31), all else being equal.
PCmd_Start	BOOL	Program Command to start motor (output from PLLS).
PCmd_Stop	BOOL	Program Command to stop motor (output from PLLS).
PCmd_Lock	BOOL	Command to acquire and lock motor in Program (output from PLLS).
PCmd_Unlock	BOOL	Command to unlock motor from Program (output from PLLS).
Sts_Available	BOOL	Motor is in Program mode and ready to operate (input to PLLS).
Sts_Stopped	BOOL	Motor is currently confirmed stopped (input to PLLS).
Sts_Starting	BOOL	Motor is currently starting (input to PLLS).
Sts_Running	BOOL	Motor is currently confirmed running (input to PLLS).
Sts_Stopping	BOOL	Motor is currently stopping (input to PLLS).
Val_Pref	DINT	This motor's current preference in list (1 = Lead, 2 = Lag, ...).
Val_Rank	DINT	This motor's current rank in list (1 = Lead, 2 = Lag, ...)

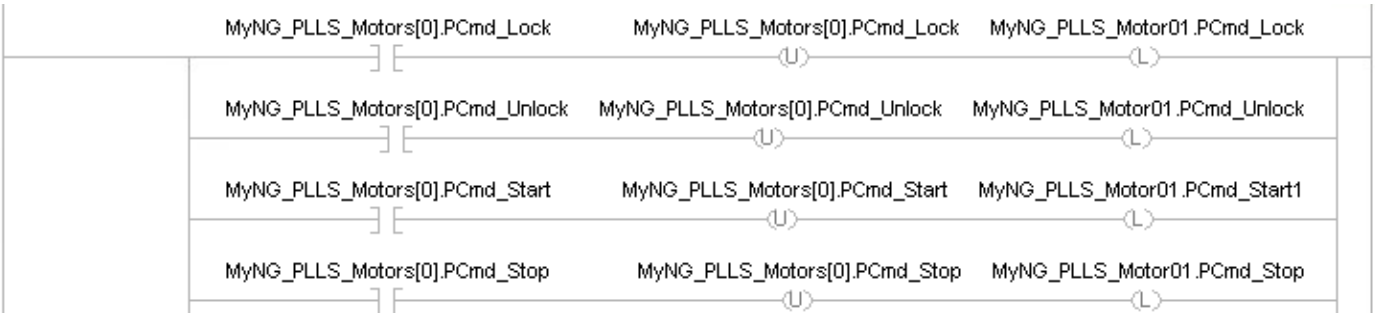
This image shows the relationship between the PLLS instruction, Ref_Motors(interface), and the PMTR instruction.



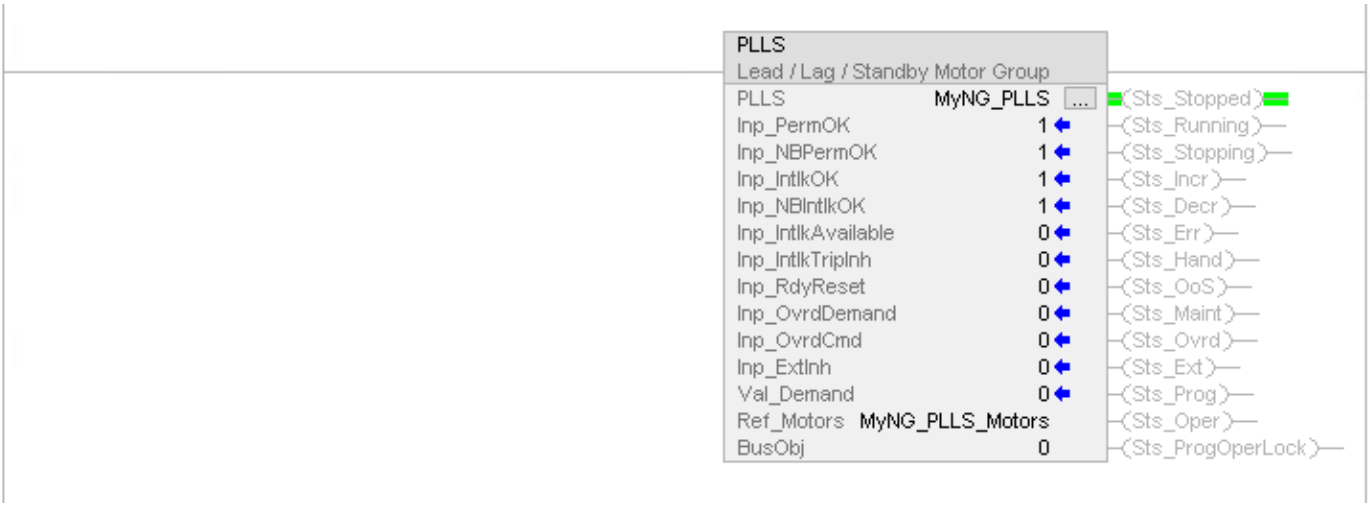
These images show an example of the ladder logic for transferring commands and motor status for one motor.

The process for forwarding each of the commands (PCmd_Lock, PCmd_Unlock, PCmd_Start, and PCmd_Stop) is:

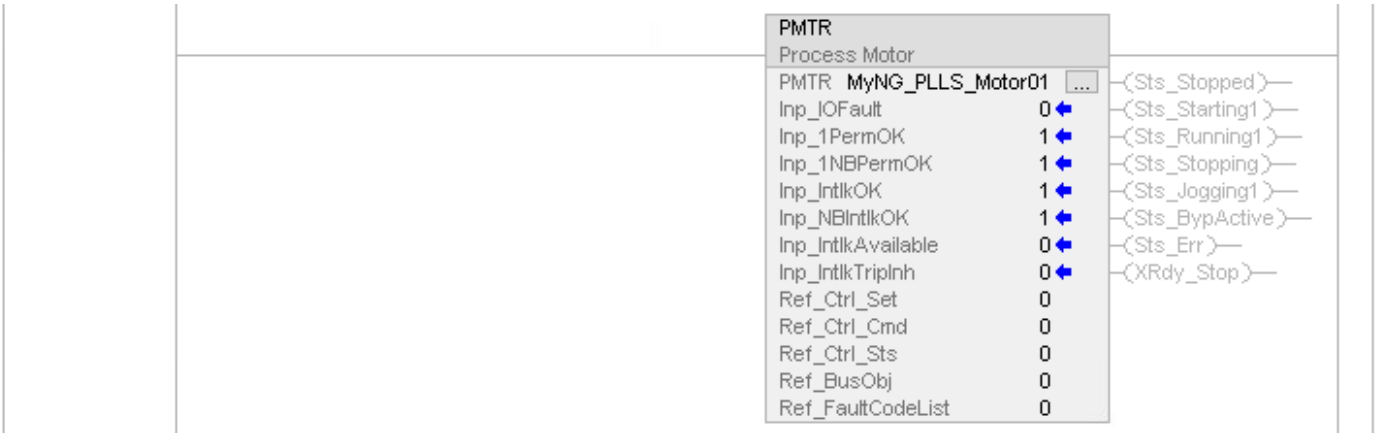
- Test the appropriate bit in the interface to see if it is set.
- If the bit is set, the bit is cleared and the corresponding program command on the motor is set.



- Execute the PLLS to select which motors to run.



- Next, the motor logic is executed. The motor logic uses the program commands to control the physical motor. The motor logic also receives feedback from the motor.



- The status (available, stopped, starting, running, and stopping) is read from the motor and written to the interface.

Monitor the PLLS Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Any commands received before first scan are discarded. The motor is de-energized and treated as if it were commanded to stop.
Instruction first run	Any commands received before first scan are discarded. The motor is de-energized and treated as if it were commanded to stop.
Rung-condition-in is false	Handled the same as if the group is disabled by command. The motor outputs are de-energized, and the group is shown as disabled on the HMI. The mode is shown as No mode. All alarms are cleared.
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	Any commands received before first scan are discarded. The motor is de-energized and treated as if it were commanded to stop.
Instruction first run	Any commands received before first scan are discarded. The motor is de-energized and treated as if it were commanded to stop.
Instruction first scan	See instruction first run in the function block diagram table.
EnableIn is false	Handled the same as if the group is disabled by command. The motor outputs are de-energized, and the group is shown as disabled on the HMI. The mode is shown as No mode. All alarms are cleared.
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

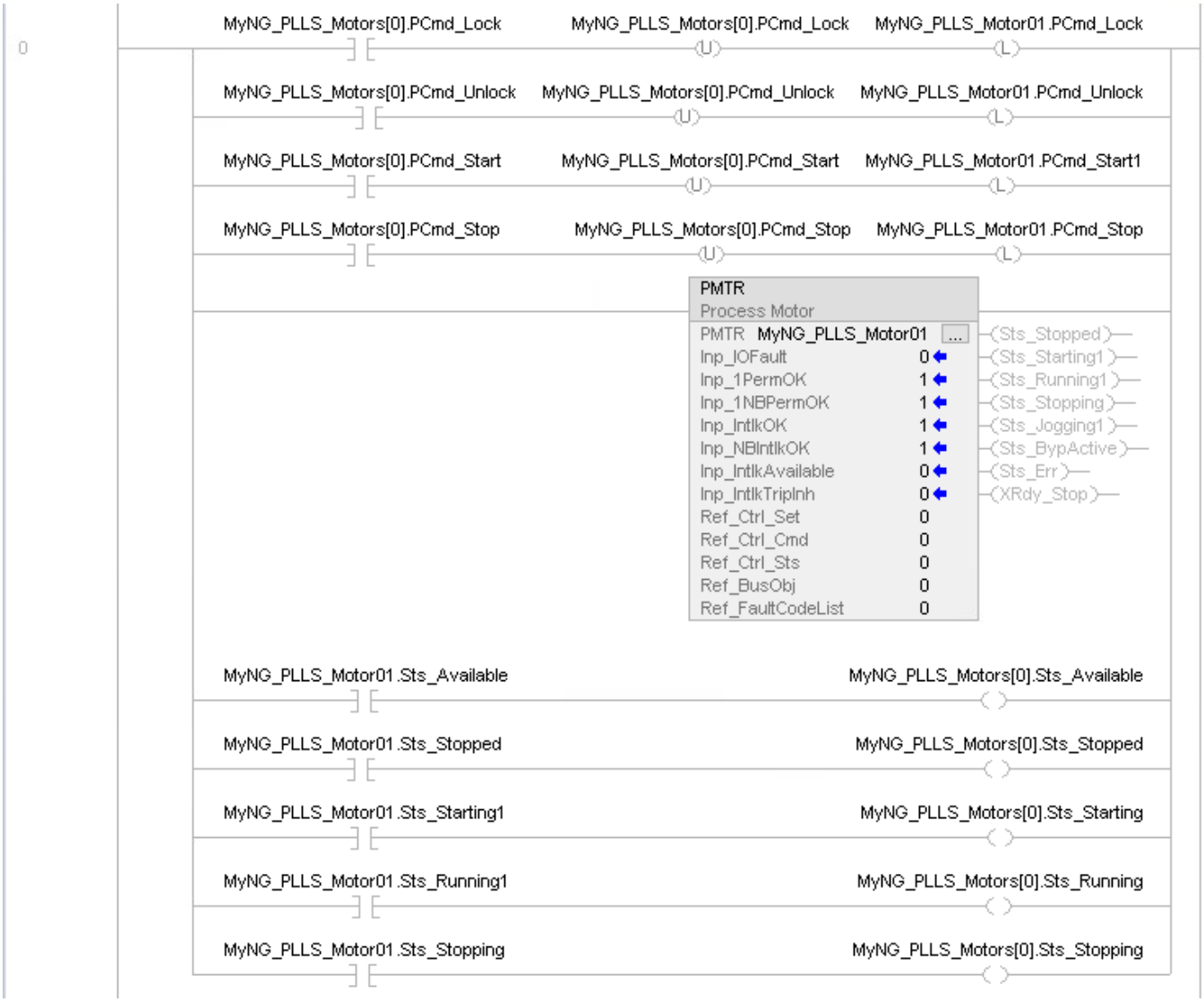
In Structured Text, EnableIn is always true during normal scan. Therefore, if the instruction is in the control path activated by the logic it will execute.

Condition/State	Action Taken
Prescan	Any commands received before first scan are discarded. The motor is de-energized and treated as if it were commanded to stop.
Instruction first run	Any commands received before first scan are discarded. The motor is de-energized and treated as if it were commanded to stop.
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

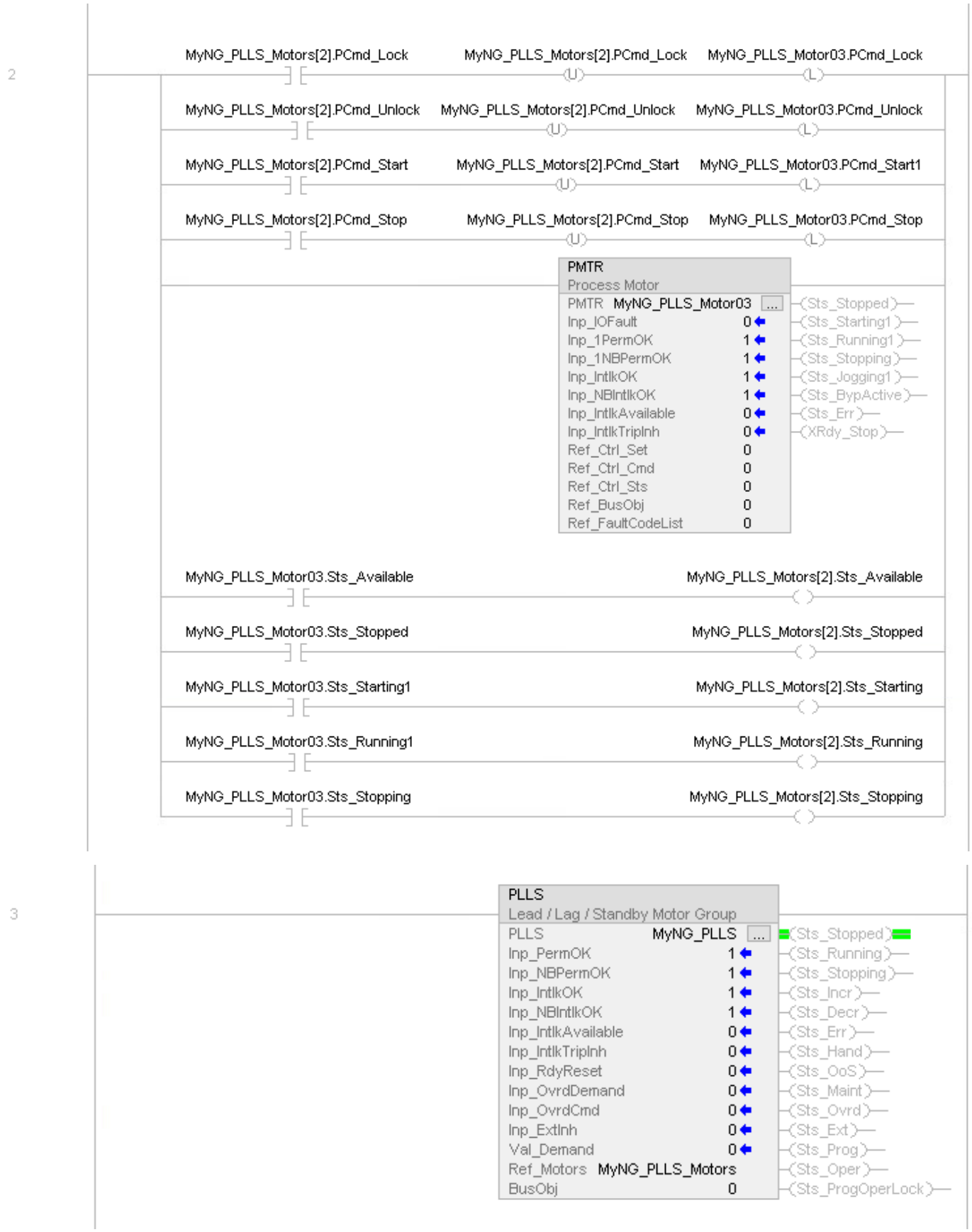
Example

This example shows a PLLS instruction being used to control three process motors.

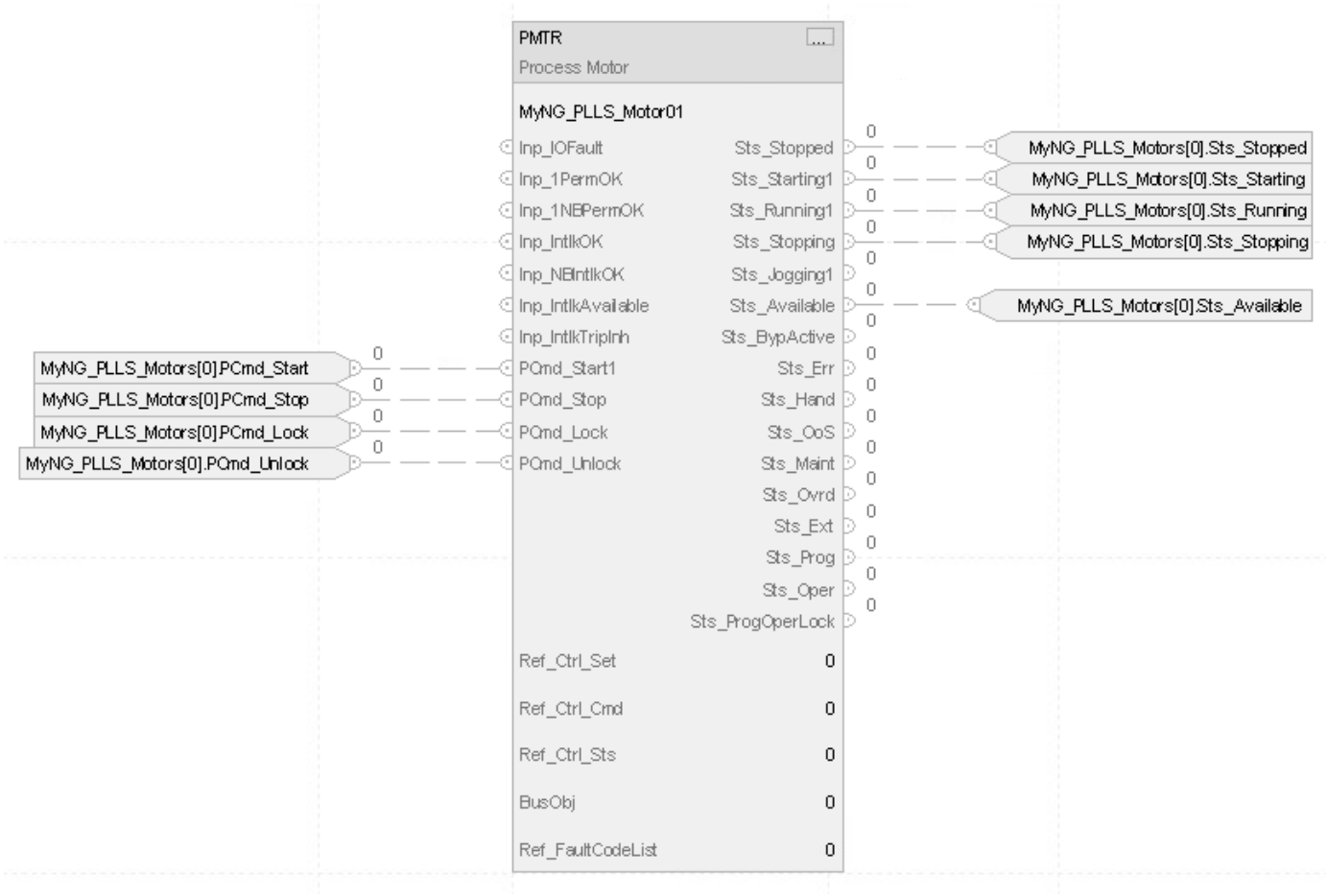
Ladder Diagram

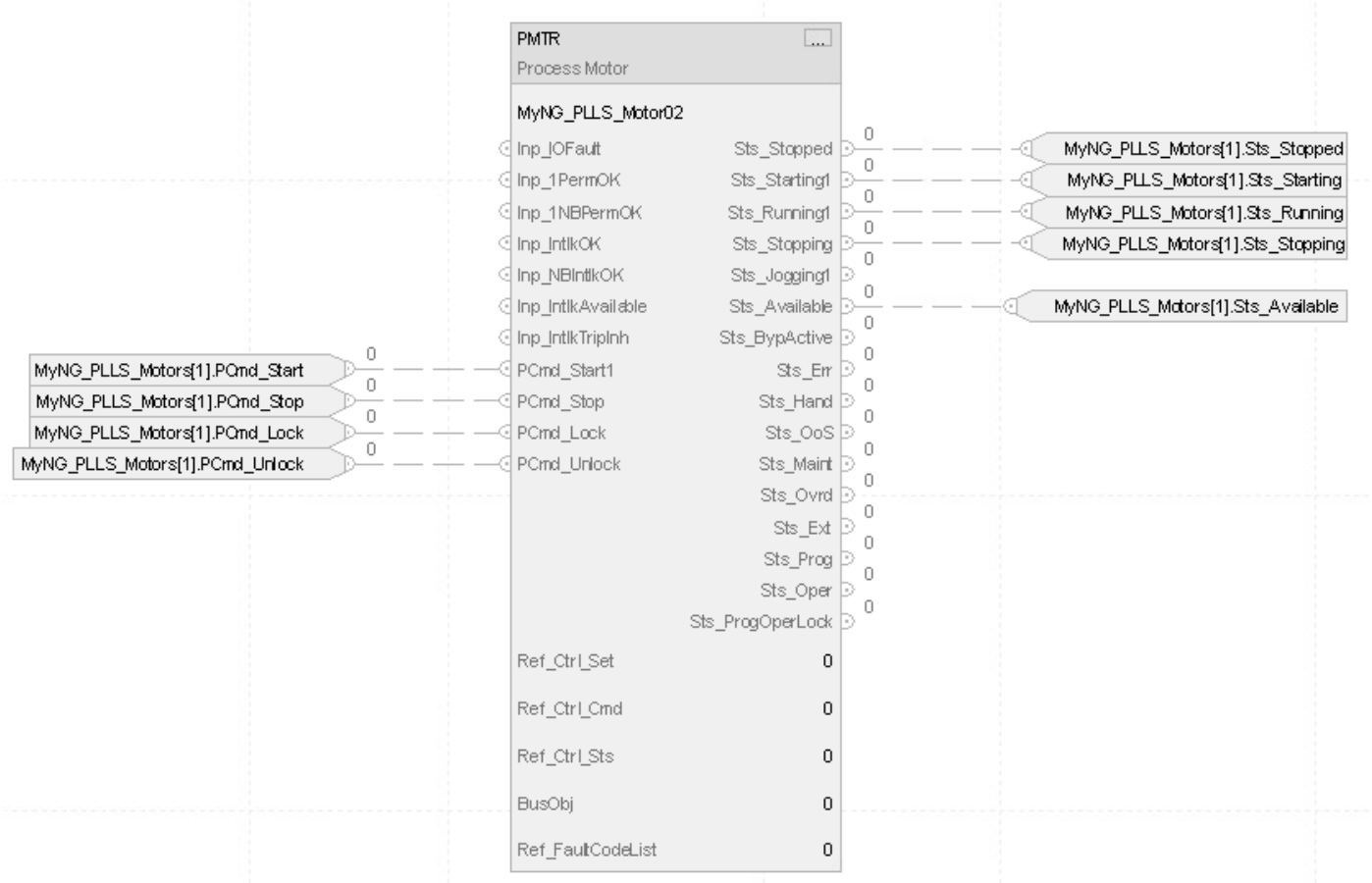


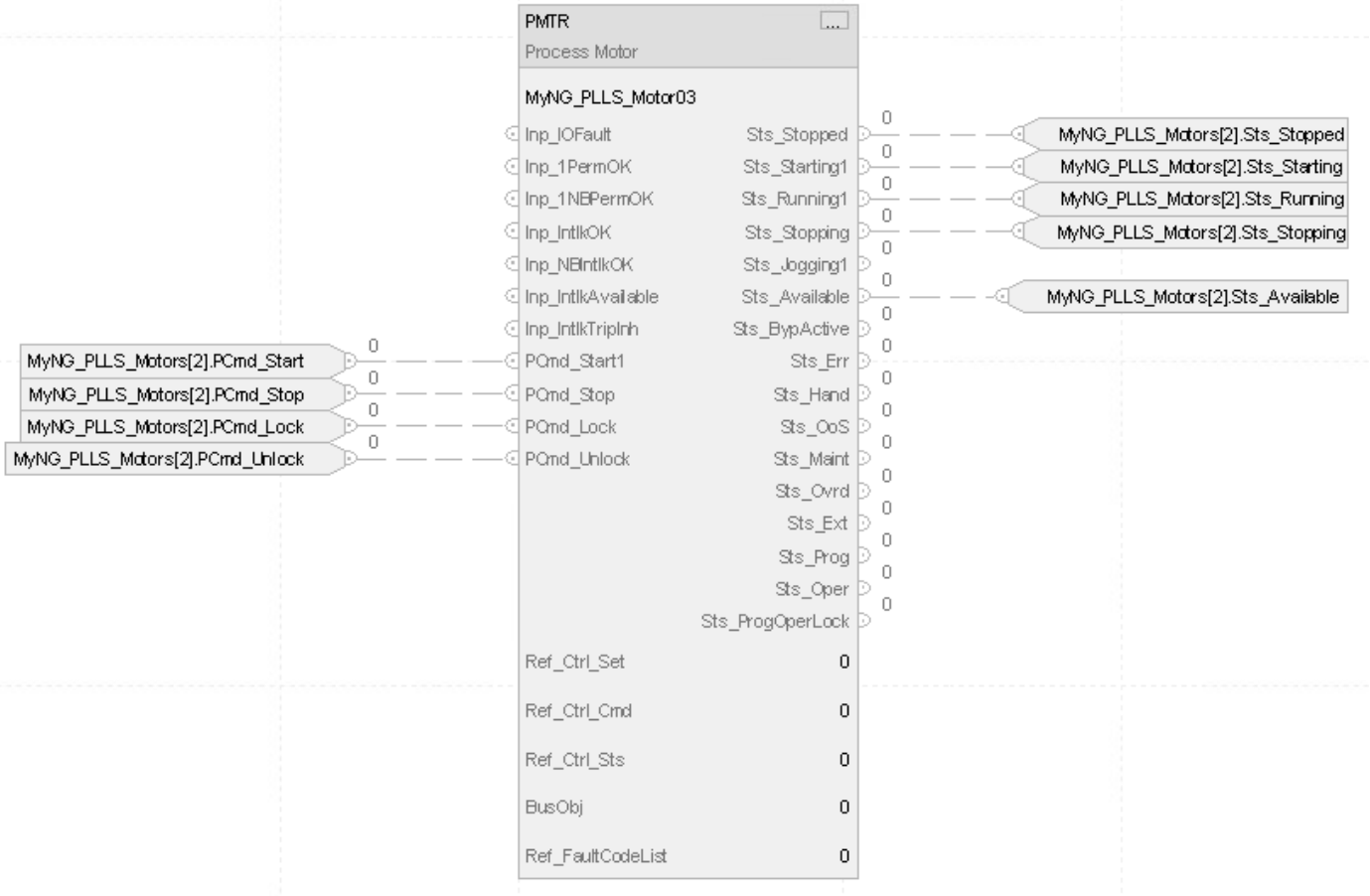
1
MyNG_PLLS_Motors[1].PCmd_Lock
MyNG_PLLS_Motors[1].PCmd_Lock
MyNG_PLLS_Motor02.PCmd_Lock
MyNG_PLLS_Motors[1].PCmd_Unlock
MyNG_PLLS_Motors[1].PCmd_Unlock
MyNG_PLLS_Motor02.PCmd_Unlock
MyNG_PLLS_Motors[1].PCmd_Start
MyNG_PLLS_Motors[1].PCmd_Start
MyNG_PLLS_Motor02.PCmd_Start1
MyNG_PLLS_Motors[1].PCmd_Stop
MyNG_PLLS_Motors[1].PCmd_Stop
MyNG_PLLS_Motor02.PCmd_Stop
PMTR
Process Motor
PMTR MyNG_PLLS_Motor02
Inp_IOFault 0
Inp_1PermOK 1
Inp_1NBPermOK 1
Inp_IntlkOK 1
Inp_NBIntlkOK 1
Inp_IntlkAvailable 0
Inp_IntlkTripInh 0
Ref_Ctrl_Set 0
Ref_Ctrl_Cmd 0
Ref_Ctrl_Sts 0
Ref_BusObj 0
Ref_FaultCodeList 0
Sts_Stopped
Sts_Starting1
Sts_Running1
Sts_Stopping
Sts_Jogging1
Sts_BypActive
Sts_Err
XRdy_Stop
MyNG_PLLS_Motor02.Sts_Available
MyNG_PLLS_Motors[1].Sts_Available
MyNG_PLLS_Motor02.Sts_Stopped
MyNG_PLLS_Motors[1].Sts_Stopped
MyNG_PLLS_Motor02.Sts_Starting1
MyNG_PLLS_Motors[1].Sts_Starting
MyNG_PLLS_Motor02.Sts_Running1
MyNG_PLLS_Motors[1].Sts_Running
MyNG_PLLS_Motor02.Sts_Stopping
MyNG_PLLS_Motors[1].Sts_Stopping

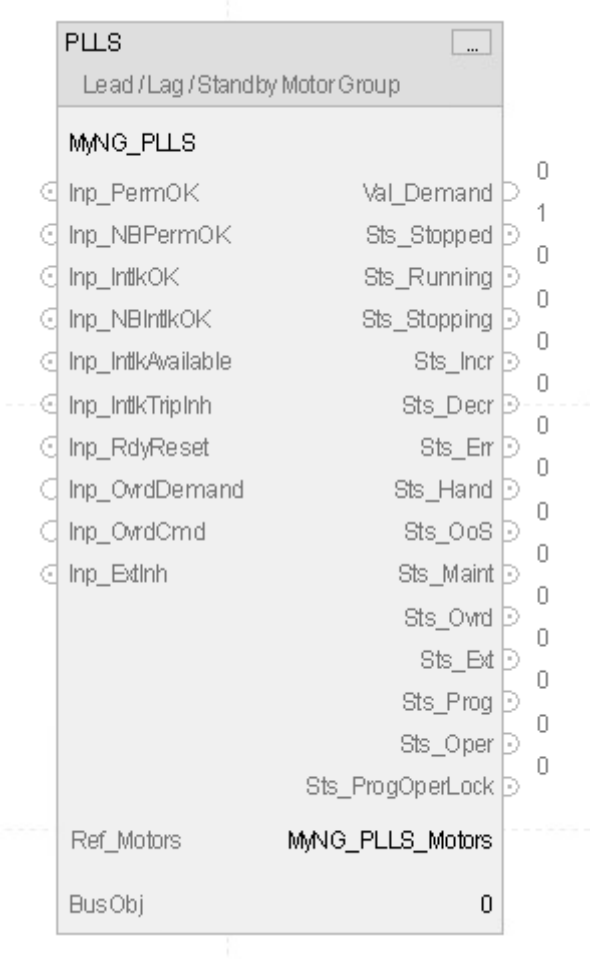

2
MyNG_PLLS_Motors[2].PCmd_Lock
MyNG_PLLS_Motors[2].PCmd_Lock
MyNG_PLLS_Motor03.PCmd_Lock
MyNG_PLLS_Motors[2].PCmd_Unlock
MyNG_PLLS_Motors[2].PCmd_Unlock
MyNG_PLLS_Motor03.PCmd_Unlock
MyNG_PLLS_Motors[2].PCmd_Start
MyNG_PLLS_Motors[2].PCmd_Start
MyNG_PLLS_Motor03.PCmd_Start1
MyNG_PLLS_Motors[2].PCmd_Stop
MyNG_PLLS_Motors[2].PCmd_Stop
MyNG_PLLS_Motor03.PCmd_Stop
PMTR
Process Motor
PMTR MyNG_PLLS_Motor03
Inp_IOFault 0
Inp_1PermOK 1
Inp_1NBPermOK 1
Inp_IntlkOK 1
Inp_NBIntlkOK 1
Inp_IntlkAvailable 0
Inp_IntlkTripInh 0
Ref_Ctrl_Set 0
Ref_Ctrl_Cmd 0
Ref_Ctrl_Sts 0
Ref_BusObj 0
Ref_FaultCodeList 0
Sts_Stopped
Sts_Starting1
Sts_Running1
Sts_Stopping
Sts_Jogging1
Sts_BypActive
Sts_Err
XRdy_Stop
MyNG_PLLS_Motor03.Sts_Available
MyNG_PLLS_Motors[2].Sts_Available
MyNG_PLLS_Motor03.Sts_Stopped
MyNG_PLLS_Motors[2].Sts_Stopped
MyNG_PLLS_Motor03.Sts_Starting1
MyNG_PLLS_Motors[2].Sts_Starting
MyNG_PLLS_Motor03.Sts_Running1
MyNG_PLLS_Motors[2].Sts_Running
MyNG_PLLS_Motor03.Sts_Stopping
MyNG_PLLS_Motors[2].Sts_Stopping
3
PLLS
Lead / Lag / Standby Motor Group
PLLS MyNG_PLLS
Inp_PermOK 1
Inp_NBPermOK 1
Inp_IntlkOK 1
Inp_NBIntlkOK 1
Inp_IntlkAvailable 0
Inp_IntlkTripInh 0
Inp_RdyReset 0
Inp_OvrdDemand 0
Inp_OvrdCmd 0
Inp_ExtInh 0
Val_Demand 0
Ref_Motors MyNG_PLLS_Motors
BusObj 0
Sts_Stopped
Sts_Running
Sts_Stopping
Sts_Incr
Sts_Decr
Sts_Err
Sts_Hand
Sts_OoS
Sts_Maint
Sts_Ovrd
Sts_Ext
Sts_Prog
Sts_Oper
Sts_ProgOperLock

Function Block Diagram




PMTR
Process Motor
MyNG_PLLS_Motor02
Inp_IOFault
Inp_1PermOK
Inp_1NBPermOK
Inp_IntlkOK
Inp_NBIntlkOK
Inp_IntlkAvailable
Inp_IntlkTripInh
PCmd_Start1
PCmd_Stop
PCmd_Lock
PCmd_Unlock
Sts_Stopped
Sts_Starting1
Sts_Running1
Sts_Stopping
Sts_Jogging1
Sts_Available
Sts_BypActive
Sts_Err
Sts_Hand
Sts_OoS
Sts_Maint
Sts_Ovrd
Sts_Ext
Sts_Prog
Sts_Oper
Sts_ProgOperLock
Ref_Ctrl_Set
Ref_Ctrl_Cmd
Ref_Ctrl_Sts
BusObj
Ref_FaultCodeList
MyNG_PLLS_Motors[1].PCmd_Start
MyNG_PLLS_Motors[1].PCmd_Stop
MyNG_PLLS_Motors[1].PCmd_Lock
MyNG_PLLS_Motors[1].PCmd_Unlock
MyNG_PLLS_Motors[1].Sts_Stopped
MyNG_PLLS_Motors[1].Sts_Starting
MyNG_PLLS_Motors[1].Sts_Running
MyNG_PLLS_Motors[1].Sts_Stopping
MyNG_PLLS_Motors[1].Sts_Available


PMTR
Process Motor
MyNG_PLLS_Motor03
Inp_IOFault
Inp_1PermOK
Inp_1NBPermOK
Inp_IntlkOK
Inp_NBIntlkOK
Inp_IntlkAvailable
Inp_IntlkTripInh
PCmd_Start1
PCmd_Stop
PCmd_Lock
PCmd_Unlock
Sts_Stopped
Sts_Starting1
Sts_Running1
Sts_Stopping
Sts_Jogging1
Sts_Available
Sts_BypActive
Sts_Err
Sts_Hand
Sts_OoS
Sts_Maint
Sts_Ovrd
Sts_Ext
Sts_Prog
Sts_Oper
Sts_ProgOperLock
Ref_Ctrl_Set
Ref_Ctrl_Cmd
Ref_Ctrl_Sts
BusObj
Ref_FaultCodeList
MyNG_PLLS_Motors[2].PCmd_Start
MyNG_PLLS_Motors[2].PCmd_Stop
MyNG_PLLS_Motors[2].PCmd_Lock
MyNG_PLLS_Motors[2].PCmd_Unlock
MyNG_PLLS_Motors[2].Sts_Stopped
MyNG_PLLS_Motors[2].Sts_Starting
MyNG_PLLS_Motors[2].Sts_Running
MyNG_PLLS_Motors[2].Sts_Stopping
MyNG_PLLS_Motors[2].Sts_Available



Structured Text

MyNG_PLLS_Motor01.PCMD_Start1:=MyNG_PLLS_Motors[0].PCmd_Start;

MyNG_PLLS_Motor01.PCMD_Stop:=MyNG_PLLS_Motors[0].PCmd_Stop;

MyNG_PLLS_Motor01.PCMD_Lock:=MyNG_PLLS_Motors[0].PCmd_Lock;

MyNG_PLLS_Motor01.PCMD_Unlock:=MyNG_PLLS_Motors[0].PCmd_Unlock;

PMTR(MyNG_PLLS_Motor01,0,0,0,0,0);

MyNG_PLLS_Motors[0].Sts_Available:=MyNG_PLLS_Motor01.Sts_Available;

MyNG_PLLS_Motors[0].Sts_Starting:=MyNG_PLLS_Motor01.Sts_Starting1;

MyNG_PLLS_Motors[0].Sts_Running:=MyNG_PLLS_Motor01.Sts_Running1;

MyNG_PLLS_Motors[0].Sts_Stopped:=MyNG_PLLS_Motor01.Sts_Stopped;

MyNG_PLLS_Motors[0].Sts_Stopping:=MyNG_PLLS_Motor01.Sts_Stopping;

MyNG_PLLS_Motor02.PCMD_Start1:=MyNG_PLLS_Motors[1].PCmd_Start;

MyNG_PLLS_Motor02.PCMD_Stop:=MyNG_PLLS_Motors[1].PCmd_Stop;

```
MyNG_PLLS_Motor02.PCMD_Lock:=MyNG_PLLS_Motors[1].PCmd_Lock;
MyNG_PLLS_Motor02.PCMD_Unlock:=MyNG_PLLS_Motors[1].PCmd_Unlock;
PMTR(MyNG_PLLS_Motor02,0,0,0,0,0);
MyNG_PLLS_Motors[1].Sts_Available:=MyNG_PLLS_Motor02.Sts_Available;
MyNG_PLLS_Motors[1].Sts_Starting:=MyNG_PLLS_Motor02.Sts_Starting1;
MyNG_PLLS_Motors[1].Sts_Running:=MyNG_PLLS_Motor02.Sts_Running1;
MyNG_PLLS_Motors[1].Sts_Stopped:=MyNG_PLLS_Motor02.Sts_Stopped;
MyNG_PLLS_Motors[1].Sts_Stopping:=MyNG_PLLS_Motor02.Sts_Stopping;
MyNG_PLLS_Motor03.PCMD_Start1:=MyNG_PLLS_Motors[2].PCmd_Start;
MyNG_PLLS_Motor03.PCMD_Stop:=MyNG_PLLS_Motors[2].PCmd_Stop;
MyNG_PLLS_Motor03.PCMD_Lock:=MyNG_PLLS_Motors[2].PCmd_Lock;
MyNG_PLLS_Motor03.PCMD_Unlock:=MyNG_PLLS_Motors[2].PCmd_Unlock;
PMTR(MyNG_PLLS_Motor03,0,0,0,0,0);
MyNG_PLLS_Motors[2].Sts_Available:=MyNG_PLLS_Motor03.Sts_Available;
MyNG_PLLS_Motors[2].Sts_Starting:=MyNG_PLLS_Motor03.Sts_Starting1;
MyNG_PLLS_Motors[2].Sts_Running:=MyNG_PLLS_Motor03.Sts_Running1;
MyNG_PLLS_Motors[2].Sts_Stopped:=MyNG_PLLS_Motor03.Sts_Stopped;
MyNG_PLLS_Motors[2].Sts_Stopping:=MyNG_PLLS_Motor03.Sts_Stopping;
PLLS(MyNG_PLLS,MyNG_PLS_Motors,0);
```

See also

Motor Sort Algorithm for Process Lead Lag Standby Motor Group (PLLS) instructions on page 572

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Motor Sort Algorithm for Process Lead Lag Standby Motor Group (PLLS) instructions

To determine the order in which the motors (pumps) are started when using a Process Lead Lag Standby Motor Group (PLLS) instruction, signed integer bit patterns for each motor are sorted by numeric value. During sorting, bit patterns are evaluated in this order:

- Out of service bit
- Status value
- Priority value
- User-input value
- Preference value
- Current position value

Out of Service (Bit 31)

This bit is used to flag the motor out of service (value = 1) and automatically send it to the bottom of the list. If this bit = 0, the motor is free to operate and bits 5...30 determine its start order.

If multiple motors are out of service, bits 5...30 determine their position at the bottom of the list.

Out of service motors are not commanded and are not counted as running even if actually running.

Status Value (Bits 30...28)

The status of the motor determines the value of these bits:

- 100 - The motor is in Hand and is not available to stop
- 010 - The motor is in Auto and is free to start or stop
- 001 - The motor is Off and is not available to start

If all motors have the same value, these bits do not affect the sort; the next set of bits become the determining factor in the sort.

Priority Value (Bits 27...23)

These bits are next in the order of precedence for sorting the array list. The value of these bits corresponds to the number entered in the Motor Priority field in the Motor Configuration dialog box.

The highest priority value has a pattern of 11111 (31), the next highest priority value is 11110 (30), and so forth.

If this priority is not to be used for the sort, set the priority value to zero for every motor.

If all motors have the same value, these bits do not affect the sort; the next set of bits become the determining factor in the sort.

User-input Values (Bits 22...15)

If the Status Values are equal and the Priority values are equal, enter values in these bits to sort the motors in the array list to the desired order.

The highest user-input value has a pattern of 11111111 (255), the next highest user-input value is 11111110 (254), and so forth.

If this value is not to be used for the sort, set the value to zero for every motor.

If all motors have the same value, these bits do not affect the sort; the next set of bits become the determining factor in the sort.

Preference Value (Bits 14...10)

These bits are next in the order of precedence for determining the order of the motors in the array list. The value of these bits corresponds to the number entered in the **Motor Preference** box in the **Motor Configuration** dialog box.

The highest preference value has a pattern of 11111 (31), the next preference value is 11110 (30), and so forth.

If this value is not to be used for the sort, set the value to zero for every motor.

If all motors have the same value, these bits do not affect the sort; the next set of bits become the determining factor in the sort.

Current Position (Bits 9...5)

IMPORTANT The current position bits are the only set of bits that cannot be equal.

These bits are next in the order of precedence for determining the order of the motors in the array list. The value of these bits corresponds to the value of the current position of the motor in the list, and the value is established by the PLLS instruction. There is no user entry for this field.

- Lead motor - 11111 (31)
- First Lag motor - 11110 (30)
- Second Lag motor - 11101 (29) and so on

The Status value, Priority value, User-input value, and Preference value must be equal for all motors for the Current Position to be a determining factor in the sort.

See also

Process Lead Lag Standby Motor Group (PLLS)

Process Motor (PMTR)

This information applies to the ControlLogix 5380P and 5580P controllers.

The Process Motor (PMTR) instruction monitors and controls a fixed single-speed, two-speed, or reversing motor using a full-voltage contactor or intelligent motor controller (soft starter). The motor can be run or jogged, including jogging reverse or jogging fast, as configured by the user. The interface to the hardware motor controller can be through a Device Object Interface or through individual pins. The object is a configurable, built-in combination of the existing PlantPAx P_Motor (single speed), P_Motor2Spd (two speed), P_MotorRev (reversing), and P_MotorHO (hand-operated or monitor-only) Add-on instructions in the Rockwell Automation Library of Process Objects.

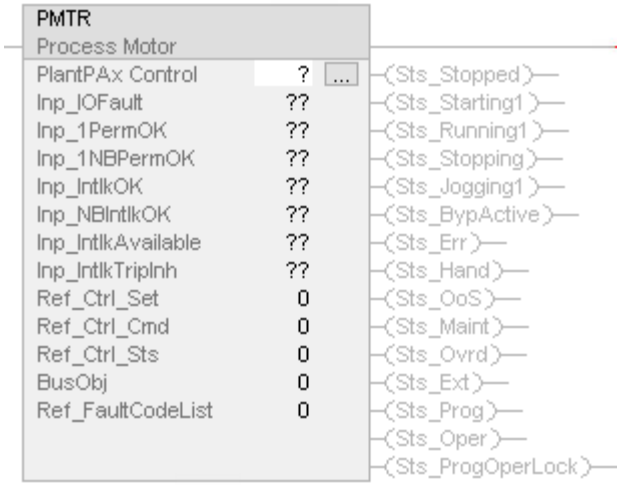
Use the PMTR instruction to:

- Monitor and control a single speed, two speed, or reversing motor using a full voltage contactor (or contactor pair) or a smart motor control (soft starter). This instruction is not used with variable speed drives controlling velocity or position, and it does not use any motion axes.
- Select Operator, Program, External, Override, Maintenance, Out of Service, or Hand as the source of motor commands.
- Use the selected command source to start the motor forward.
- Use the selected command source to start the motor reverse, if configured for reversing, or start the motor at high speed, if configured for two-speed operation.
- Use the selected command source to jog the motor forward. Only Operator, External and Maintenance command sources are permitted to jog the motor.
- Use the selected command source to jog the motor reverse, if configured for reversing, or jog the motor at high speed, if configured for two-speed operation. Only Operator, External and Maintenance command sources are permitted to jog the motor.
- Monitor actual motor status, including:
  - Run feedback (including separate feedback for slow and fast for two speed operation or forward and reverse for reversing operation)
  - Motor controller ready
  - Commanded direction / speed
  - Actual direction / speed
  - Motor controller warning
  - Motor controller faulted (with fault code and description)

- Interface to a motor Device Object using a set of Power Discrete interface tags. If the interface tags are not linked (optional InOut parameters), a set of input and output parameters are used to interface to the starter or motor controller signal-by-signal.
- Search a linked Fault Code Lookup Table to provide textual motor controller fault information, or use text provided via the Power Discrete interface fault record.
- Participate in a control strategy bus (BUS_OBJ) with other devices and process instructions.
- Configure an output to pre-start warning audible (horn) with configurable alert time before starting or jogging.
- Configure virtualization, providing simulated feedback of a working motor while disabling outputs to the physical device.
- Monitor run feedback and status or alarms for failure to start in the configured time and failure to stop in the configured time.
- Monitor Permissive conditions to allow starting or jogging the motor forward / slow.
- Monitor Permissive conditions to allow starting or jogging the motor reverse / fast.
- Monitor Interlock conditions to stop and prevent starting or jogging the motor.
- Monitor I/O communication faults.
- Trigger an alarm if interlock conditions cause the motor to be stopped.
- Automatically clear latched alarms and motor controller faults when an Operator Command (Start, Stop, Jog) is received.
- Automatically clear latched alarms and motor controller faults when an External Command (Start, Stop, Jog) is received.
- Use HMI breadcrumbs for Alarm Inhibited, Bad Configuration, Not Ready, and Maintenance Bypass Active.
- Use Available status for use by automation logic to indicate whether a motor can be controlled by other objects.
- Use Alarms for Fail to Start, Fail to Stop, Interlock Trip, I/O Fault, and Motor Fault.

Available Languages

Ladder Diagram



Function Block Diagram

Structured Text

PMTR (PMTRTag, Ref_Ctrl_Set tag, Ref_Ctrl_Cmd tag, Ref_Ctrl_Sts tag, BusObj tag, Ref_FaultCodeList tag);

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PlantPAx Control	P_MOTOR_DISCRETE	tag	Data structure required for proper operation of instruction.
Ref_Ctrl_Set	RAC_ITF_DVC_PWRDISCRETE_SET	tag	Power Discrete Device Object Settings Interface.
Ref_Ctrl_Cmd	RAC_ITF_DVC_PWRDISCRETE_CMD	tag	Power Discrete Device Object Command Interface.
Ref_Ctrl_Sts	RAC_ITF_DVC_PWRDISCRETE_STS	tag	Power Discrete Device Object Status Interface.
BusObj	BUS_OBJ	tag	Bus component.
Ref_FaultCodeList	RAC_CODE_DESCRIPTION[x]	tag	Fault Code to Fault Description lookup table for intelligent motor controller.

P_MOTOR_DISCRETE Structure

Public members are standard, visible tag members that are programmatically accessible. Private, hidden members are used in HMI faceplates and are not programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
EnableIn	BOOL	Not Visible	Not Required	Input	Enable Input - System Defined Parameter Default is true.
Inp_InitializeReq	BOOL	Not Visible	Not Required	Input	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request when reinitialization is needed. The instruction clears this operand automatically. Default is true.
Inp_OwnerCmd	DINT	Not Visible	Not Required	Input	Owner device command. 0 = None, .10 = Operator Lock, .11 = Operator Unlock, .12 = Program Lock, .13 = Program Unlock, .14 = Acquire Maintenance, .15 = Release Maintenance, .16 = Acquire External, .17 = Release External, .29 = Echo. Default is 0.
Inp_LastFaultCodeData	DINT	Not Visible	Not Required	Input	Most recent intelligent motor controller Fault Code (enumeration). Default is 0.
Inp_ReadyData	BOOL	Not Visible	Not Required	Input	1=Intelligent motor controller is ready to run. Default is true.
Inp_1RunFdbkData	BOOL	Not Visible	Not Required	Input	1=Motor is Running Forward or Slow. Default is false.
Inp_2RunFdbkData	BOOL	Not Visible	Not Required	Input	1=Motor is Running Reverse or Fast. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Inp_AlarmData	BOOL	Not Visible	Not Required	Input	1=Intelligent motor controller has a Warning or Alarm. See controller display or manual. Default is false.
Inp_FaultedData	BOOL	Not Visible	Not Required	Input	1=Intelligent motor controller has Faulted. See controller display or manual. Default is false.
Inp_DvcNotify	SINT	Not Visible	Not Required	Input	Related device object alarm priority and acknowledgement status. 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Inp_IOFault	BOOL	Visible	Not Required	Input	Indicates the IO data are inaccurate. 0 = The IO data are good, 1 = The IO data are bad, causing fault. If the Motor is not virtual, this input sets Sts_IOFault, which raises IOFault Alarm. Default is false.
Inp_1PermOK	BOOL	Visible	Not Required	Input	1 = Permissives OK, motor can start or jog Forward / Slow. Default is true.
Inp_1NBPermOK	BOOL	Visible	Not Required	Input	1 = Non-Bypassable Permissives OK, motor can start or jog Forward / Slow. Default is true.
Inp_2PermOK	BOOL	Not Visible	Not Required	Input	1 = Permissives OK, motor can start or jog Reverse / Fast. Default is true.
Inp_2NBPermOK	BOOL	Not Visible	Not Required	Input	1 = Non-Bypassable Permissives OK, motor can start or jog Reverse / Fast. Default is true.
Inp_IntlkOK	BOOL	Visible	Not Required	Input	1 = Interlocks OK, motor can start or jog and keep running. Default is true.
Inp_NBIntlkOK	BOOL	Visible	Not Required	Input	1 = Non-bypassable interlocks OK, motor can start or jog and keep running. Default is true.
Inp_IntlkAvailable	BOOL	Visible	Not Required	Input	1 = Interlock Availibility OK. Default is false.
Inp_IntlkTripInh	BOOL	Visible	Not Required	Input	1 = Inhibit Interlock Trip Status Default is false.
Inp_RdyReset	BOOL	Not Visible	Not Required	Input	1 = Related object, reset by this object, is ready to be reset. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Inp_Hand	BOOL	Not Visible	Not Required	Input	1 = Acquire Hand (typically hardwired local), 0 = Release Hand. Default is false.
Inp_Ovrd	BOOL	Not Visible	Not Required	Input	1 = Acquire Override (higher priority program logic), 0 = Release Override. Default is false.
Inp_OvrdCmd	SINT	Not Visible	Not Required	Input	Override Command: 0 = None, 1 = Stop, 2 = Start 1 (Forward / Slow), 3 = Start2 (Reverse / Fast). Default is 0.
Inp_ExtInh	BOOL	Not Visible	Not Required	Input	1 = Inhibit External acquisition, 0 = Allow External acquisition. Default is false.
Inp_HornInh	BOOL	Not Visible	Not Required	Input	1 = Inhibit audible alert, 0 = Alllow audible alert. Default is false.
Inp_Reset	BOOL	Not Visible	Not Required	Input	1 = Reset Shed Latches and Cleared Alarms. Default is false.
Cfg_AllowDisable	BOOL	Not Visible	Not Required	Input	1 = Allow Maintenance to disable alarms. Default is true.
Cfg_AllowShelve	BOOL	Not Visible	Not Required	Input	1 = Allow Operator to shelve alarms. Default is true.
Cfg_eObjType	SINT	Not Visible	Not Required	Input	Type of Motor: 0 = single-speed, 1 = reversing, 2 = 2-speed (enumeration). Default is 0.
Cfg_HasStart1	BOOL	Not Visible	Not Required	Input	1 = Motor Start1 (Forward / Slow) Command enabled and visible, 0 = Motor Start1 Command not allowed. Default is false.
Cfg_HasStart2	BOOL	Not Visible	Not Required	Input	1 = Motor Start2 (Reverse / Fast) Command enabled and visible, 0 = Motor Start2 Command not allowed. Default is false.
Cfg_HasJog1	BOOL	Not Visible	Not Required	Input	1 = Motor Jog1 (Forward / Slow) Command enabled and visible, 0 = Motor Jog1 Command not allowed. Default is false.
Cfg_HasJog2	BOOL	Not Visible	Not Required	Input	1 = Motor Jog2 (Reverse / Fast) Command enabled and visible, 0 = Motor Jog2 Command not allowed. Default is false.
Cfg_HasStop	BOOL	Not Visible	Not Required	Input	1 = Motor Stop Command enabled and visible, 0 = Instruction has no control, only monitors Motor. Default is true.
Cfg_AllowLocal	BOOL	Not Visible	Not Required	Input	1 = Allow Local Start/Stop without alarm, 0 = Start/Stop by command only. Default is false.
Cfg_HasRunFdbk	BOOL	Not Visible	Not Required	Input	1 = Motor provides feedback signal when running. Default is false.
Cfg_UseRunFdbk	BOOL	Not Visible	Not Required	Input	1 = Motor run feedback should be used for failure checking. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_HasDvcObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI a Device (e.g., overload relay) object is connected. Default is false.
Cfg_Has1PermObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI an object is connected to Inp_1Perm inputs. Default is false.
Cfg_Has2PermObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI an object is connected to Inp_2Perm inputs. Default is false.
Cfg_HasIntlkObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI an object is connected to Inp_Intlk inputs. Default is false.
Cfg_HasResInhObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI a Restart Inhibit object is connected. Default is false.
Cfg_HasRunTimeObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI a Run Time / Starts object is connected. Default is false.
Cfg_HasMoreObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI an object with more info is available. Default is false.
Cfg_OperStopPrio	BOOL	Not Visible	Not Required	Input	1 = OCmd_Stop accepted any time; 0 = OCmd_Stop accepted only when Oper is selected. Default is false.
Cfg_ExtStopPrio	BOOL	Not Visible	Not Required	Input	1 = XCmd_Stop accepted any time; 0 = XCmd_Stop accepted only when Ext is selected. Default is false.
Cfg_OCmdResets	BOOL	Not Visible	Not Required	Input	1 = Any Motor OCmd resets shed latches and cleared alarms; 0 = OCmdReset is required. Default is false.
Cfg_XCmdResets	BOOL	Not Visible	Not Required	Input	1 = Any Motor XCmd resets shed latches and cleared alarms; 0 = XCmdReset is required. Default is false.
Cfg_OvrdPermIntlk	BOOL	Not Visible	Not Required	Input	1 = Override ignores Bypassable Perm/ Intlk; 0 = Override uses all Perm/Intlk. Default is false.
Cfg_ShedOnFailToStart	BOOL	Not Visible	Not Required	Input	1 = Stop Motor and Alarm on Fail to Start; 0 = Alarm only on Fail to Start. Default is true.
Cfg_ShedOnIOFault	BOOL	Not Visible	Not Required	Input	1 = Stop Motor and Alarm on I/O Fault; 0 = Alarm only on I/O Fault. Default is true.
Cfg_HasOper	BOOL	Not Visible	Not Required	Input	1 = Operator (unlocked) exists, can be selected. Default is true.
Cfg_HasOperLocked	BOOL	Not Visible	Not Required	Input	1 = Operator Locked exists, can be selected. Default is true.
Cfg_HasProg	BOOL	Not Visible	Not Required	Input	1 = Program (unlocked) exists, can be selected. Default is true.
Cfg_HasProgLocked	BOOL	Not Visible	Not Required	Input	1 = Program Locked exists, can be selected. Default is true.
Cfg_HasExt	BOOL	Not Visible	Not Required	Input	1 = External exists, can be selected. Default is false.
Cfg_HasMaint	BOOL	Not Visible	Not Required	Input	1 = Maintenance exists, can be selected. Default is true.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_HasMaintOoS	BOOL	Not Visible	Not Required	Input	1 = Maintenance Out of Service exists, can be selected. Default is true.
Cfg_OvrdOverLock	BOOL	Not Visible	Not Required	Input	1 = Override supersedes Program/Operator Lock, 0 = Don't override Lock. Default is true.
Cfg_ExtOverLock	BOOL	Not Visible	Not Required	Input	1 = External supersedes Program/Operator Lock, 0 = Don't override Lock. Default is false.
Cfg_ProgPwrUp	BOOL	Not Visible	Not Required	Input	1 = Power up to Program, 0 = Power up to Operator. Default is false.
Cfg_ProgNormal	BOOL	Not Visible	Not Required	Input	Normal Source: 1 = Program if no requests; 0 = Operator if no requests. Default is false.
Cfg_PCmdPriority	BOOL	Not Visible	Not Required	Input	Command priority. 1 = Program commands win, 0 = Operator commands win. Default is false.
Cfg_PCmdProgAsLevel	BOOL	Not Visible	Not Required	Input	1 = PCmd_Prog used as Level (1 = Acquire, 0 = Release). Default is false.
Cfg_PCmdLockAsLevel	BOOL	Not Visible	Not Required	Input	1 = PCmd_Lock used as a Level (1 = Lock, 0 = Unlock). Default is false.
Cfg_ExtAcqAsLevel	BOOL	Not Visible	Not Required	Input	1 = XCmd_Acq used as Level (1 = Acquire, 0 = Release). Default is false.
Cfg_PauseTime	REAL	Not Visible	Not Required	Input	Delay in seconds with contactors open when changing speed or direction. Valid = 0.0 to 2147483.0 seconds. Default is 3.0.
Cfg_StartHornTime	REAL	Not Visible	Not Required	Input	Time in seconds to sound audible on commanded start. Valid = 0.0 to 1000.0 seconds, 0.0 = disabled. Default is 0.0.
Cfg_VirtualFdbkTime	REAL	Not Visible	Not Required	Input	Time in seconds to echo run feedback when Virtualized. Valid = 0.0 to 2147483.0 seconds. Default is 10.0.
Cfg_FailToStartTime	REAL	Not Visible	Not Required	Input	Time in seconds after Start to receive Run Feedback before Fault. Valid = 0.0 to 2147483.0 seconds. Default is 15.0.
Cfg_FailToStopTime	REAL	Not Visible	Not Required	Input	Time in seconds after Stop to drop Run Feedback before Fault. Valid = 0.0 to 2147483,0 seconds. Default is 15.0.
Cfg_ResetPulseTime	REAL	Not Visible	Not Required	Input	Time in seconds to pulse Out_Reset to clear Motor fault. Valid = 0.0 to 2147483.0 seconds. Default is 2.0.
Cfg_MaxJogTime	REAL	Not Visible	Not Required	Input	Maximum jog time in seconds. Valid = 0.0 to 2147483.0 seconds, 0.0 = unlimited). Default is 0.0.
Cfg_eKeepStart	SINT	Not Visible	Not Required	Input	Ownership of Start commands (enumeration): 0 = follows CmdSrc, 1 = Operator, 2 = Program, 3 = External. Default is 0.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_eKeepJog	SINT	Not Visible	Not Required	Input	Ownership of Jog commands (enumeration): 0 = follows CmdSrc, 1 = Operator, 3 = External. Default is 0.
Cfg_CnfrmReqd	SINT	Not Visible	Not Required	Input	Operator Command Confirmation Required. Represents the type of command confirmation required. 0 = None, 1 = Command confirmation required, 2 = Performer e-signature required, 3 = Performer and approver e-signature required. Default is 0.
Cfg_HasHistTrend	SINT	Not Visible	Not Required	Input	Has Historical Trend. This enables navigation to the Device Historical Trend Faceplate from the HMI. 0 = No external historical trend, 1 = Datalog historical trend, 2 = Historian historical trend. Default is 0.
PSet_Owner	DINT	Not Visible	Not Required	Input	Program owner request ID (non-zero) or release (zero). Default is 0.
PCmd_Virtual	BOOL	Not Visible	Not Required	Input	Program command to select Virtual (simulated) device operation. The instruction clears this operand automatically. Default is false.
PCmd_Physical	BOOL	Not Visible	Not Required	Input	Program command to select Physical device operation (not simulated). The instruction clears this operand automatically. Default is false.
PCmd_Start1	BOOL	Not Visible	Not Required	Input	Program command to Start Motor Forward / Slow. The instruction clears this operand automatically. Default is false.
PCmd_Start2	BOOL	Not Visible	Not Required	Input	Program command to Start Motor Reverse / Fast. The instruction clears this operand automatically. Default is false.
PCmd_Stop	BOOL	Not Visible	Not Required	Input	Program command to Stop Motor. The instruction clears this operand automatically. Default is false.
PCmd_Prog	BOOL	Not Visible	Not Required	Input	Program command to select Program (Operator to Program). The instruction clears this operand automatically if Cfg_PCmdProgAsLevel = 0. Default is false.
PCmd_Oper	BOOL	Not Visible	Not Required	Input	Program command to select Operator (Program to Operator). The instruction clears this operand automatically. Default is false.
PCmd_Lock	BOOL	Not Visible	Not Required	Input	Program command to lock Program (disallow Operator). The instruction clears this operand automatically if Cfg_PCmdLockAsLevel = 0. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
PCmd_Unlock	BOOL	Not Visible	Not Required	Input	Program command to unlock Program (allow Operator to acquire). The instruction clears this operand automatically. Default is false.
PCmd_Normal	BOOL	Not Visible	Not Required	Input	Program command to select Normal command source (Operator or Program). The instruction clears this operand automatically. Default is false.
PCmd_Reset	BOOL	Not Visible	Not Required	Input	Program command to reset all alarms and latched shed conditions requiring reset. The instruction clears this operand automatically. Default is false.
MCmd_Rel	BOOL	Not Visible	Not Required	Input	Maintenance command to release ownership (Maintenance to Operator/Program/External/Override). The instruction clears this operand automatically. Default is false.
OCmd_Unlock	BOOL	Not Visible	Not Required	Input	Operator command to unlock / release (allow Program to acquire) ownership. The instruction clears this operand automatically. Default is false.
XCmd_Start1	BOOL	Not Visible	Not Required	Input	External command to Start Motor Forward / Slow. The instruction clears this operand automatically. Default is false.
XCmd_Start2	BOOL	Not Visible	Not Required	Input	External command to Start Motor Reverse / Fast. The instruction clears this operand automatically. Default is false.
XCmd_Stop	BOOL	Not Visible	Not Required	Input	External command to Stop Motor. The instruction clears this operand automatically. Default is false.
XCmd_Jog1	BOOL	Not Visible	Not Required	Input	External command to Jog Motor Forward / Slow. The instruction clears this operand automatically if max jog time is reached. Default is false.
XCmd_Jog2	BOOL	Not Visible	Not Required	Input	External command to Jog Motor Reverse / Fast. The instruction clears this operand automatically if max jog time is reached. Default is false.
XCmd_Acq	BOOL	Not Visible	Not Required	Input	External command to acquire ownership (Operator/Program/Override/Maintenance to External). The instruction clears this operand automatically if Cfg_ExtAcqAsLevel = 0. Default is false.
XCmd_Rel	BOOL	Not Visible	Not Required	Input	External command to release ownership if Cfg_ExtAcqAsLevel = 0 (External to Operator/Program/Override/Maintenance). The instruction clears this operand automatically. Default is false.
XCmd_Reset	BOOL	Not Visible	Not Required	Input	External command to clear shed latches and cleared alarms. The instruction clears this operand automatically. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
XCmd_ResetAckAll	BOOL	Not Visible	Not Required	Input	External command to acknowledge and reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
EnableOut	BOOL	Not Visible	Not Required	Output	Enable Output - System Defined Parameter
Out_Run1Data	BOOL	Not Visible	Not Required	Output	1=Start/Run Motor Forward or Slow, 0=Stop Motor (for held starter type).
Out_Run2Data	BOOL	Not Visible	Not Required	Output	1=Start/Run Motor Reverse or Fast, 0=Stop Motor (for held starter type).
Out_Start1Data	BOOL	Not Visible	Not Required	Output	1=Start Motor Forward or Slow, 0=Motor left in current state.
Out_Start2Data	BOOL	Not Visible	Not Required	Output	1=Start Motor Reverse or Fast, 0=Motor left in current state.
Out_StopData	BOOL	Not Visible	Not Required	Output	1=Stop Motor, 0=Motor left in current state.
Out_ClearFaultData	BOOL	Not Visible	Not Required	Output	1=Attempt to clear Fault on intelligent motor controller.
Out_HornData	BOOL	Not Visible	Not Required	Output	1 = Sound audible prior to commanded motor start.
Out_Reset	BOOL	Not Visible	Not Required	Output	1 = Reset command has been received and accepted.
Out_OwnerSts	DINT	Not Visible	Not Required	Output	Status of command source, owner command handshake and ready status. 0 = None, .10 = Operator Lock, .11 = Operator Unlock, .12 = Program Lock, .13 = Program Unlock, .14 = Acquire Maintenance, .15 = Release Maintenance, .16 = Acquire External, .17 = Release External, .18 = Has Maintenance, .19 = External Override Lock, .20 = Has External, .21 = Has Operator, .22 = Has Operator Locked, .23 = Has Program, .24 = Has Program Locked, .29 = Echo, .30 = Not Ready.
Sts_Initialized	BOOL	Not Visible	Not Required	Output	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.
Sts_Stopped	BOOL	Visible	Not Required	Output	1 = Motor requested to stop and is confirmed stopped.
Sts_Starting1	BOOL	Visible	Not Required	Output	1 = Motor requested to run forward and awaiting run feedback.
Sts_Starting2	BOOL	Not Visible	Not Required	Output	1 = Motor requested to run reverse and awaiting run feedback.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_Running1	BOOL	Visible	Not Required	Output	1 = Motor requested to run and is confirmed running forward.
Sts_Running2	BOOL	Not Visible	Not Required	Output	1 = Motor requested to run and is confirmed running reverse.
Sts_Stopping	BOOL	Visible	Not Required	Output	1 = Motor running / jogging requested to stop and awaiting stopped feedback.
Sts_Jogging1	BOOL	Visible	Not Required	Output	1 = Motor requested to Jog Forward.
Sts_Jogging2	BOOL	Not Visible	Not Required	Output	1 = Motor requested to Jog Reverse.
Sts_Horn	BOOL	Not Visible	Not Required	Output	1 = Motor Audible Alert (Horn) is Active.
Sts_NotReady	BOOL	Not Visible	Not Required	Output	1 = Motor is Not Ready (cannot be started) Check alarms, stops, faults.
Sts_Alarm	BOOL	Not Visible	Not Required	Output	1 = Intelligent motor controller has an Alarm (see controller display or manual).
Sts_Virtual	BOOL	Not Visible	Not Required	Output	1 = The instruction treats the Motor as virtual. The instruction acts as normal but the output is kept de-energized; 0 = The instruction operates the Motor normally. Sts_Virtual is a copy of Sts_Virtual.
SrcQ_IO	SINT	Not Visible	Not Required	Output	Source and quality of primary input or output (enumerated): 0=Good, live, confirmed good 1=Good, live, assumed good 2=Good, no feedback, assumed good 8=Test, virtualized 9=Test, loopback 10=Test, manually entered 16=Uncertain, live, off-spec 17=Uncertain, substituted at device or bus 18=Uncertain, substituted at instruction 19=Uncertain, using last known good 20=Uncertain, using replacement value 32=Bad, signal failure 33=Bad, channel fault 34=Bad, module or communication fault 35=Bad, invalid configuration

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
SrcQ	SINT	Not Visible	Not Required	Output	Source and quality of primary value or status (enumerated): 0=Good, live, confirmed good 1=Good, live, assumed good 2=Good, no feedback, assumed good 8=Test, virtualized 9=Test, loopback 10=Test, manually entered 16=Uncertain, live, off-spec 17=Uncertain, substituted at device or bus 18=Uncertain, substituted at instruction 19=Uncertain, using last known good 20=Uncertain, using replacement value 32=Bad, signal failure 33=Bad, channel fault 34=Bad, module or communication fault 35=Bad, invalid configuration
Sts_eCmd	SINT	Not Visible	Not Required	Output	Motor Command: 0 = None, 1 = Stop, 2 = Start 1, 3 = Start 2, 4 = Jog 1, 5 = Jog 2.
Sts_eFdbk	SINT	Not Visible	Not Required	Output	Motor Feedback: 0 = Stopped, 1 = Running Forward, 2 = Running Reverse, 3 = Running Slow, 4 = Running Fast.
Sts_eSts	SINT	Not Visible	Not Required	Output	Motor Status: 0 = Unknown, 1 = Stopped, 2 = Running 1, 3 = Running 2, 4 = Starting 1, 5 = Starting 2, 6 = Jogging 1, 7 = Jogging 2, 8 = Stopping, 14 = Horn, 15 = Out of Service.
Sts_eFault	SINT	Not Visible	Not Required	Output	Motor Fault Status: 0 = None, 15 = Interlock Trip, 16 = Fail to Start, 17 = Fail to Stop, 18 = Motor Fault, 32 = I/O Fault, 34 = Config Error.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_eNotify	SINT	Not Visible	Not Required	Output	All alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyAll	SINT	Not Visible	Not Required	Output	All alarm status enumerated values including related objects: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyIOFault	SINT	Not Visible	Not Required	Output	IOFault alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyFailToStart	SINT	Not Visible	Not Required	Output	Fail to Start alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_eNotifyFailToStop	SINT	Not Visible	Not Required	Output	Fail to Stop alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyIntlkTrip	SINT	Not Visible	Not Required	Output	IntlkTrip alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyMotorFault	SINT	Not Visible	Not Required	Output	Motor Fault alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged
Sts_UnackAlmCount	DINT	Not Visible	Not Required	Output	Count of unacknowledged alarms.
Sts_eFaultCode	DINT	Not Visible	Not Required	Output	First Trip Code after reset. See Motor manual or Power Discrete Object for description.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_eSrc	INT	Not Visible	Not Required	Output	The current command source enumerated values: 0 = Logic not in use, 4 = Hand, 8 = Maintenance, 16 = Override, 32 = Program, 33 = Program locked, 34 = Program by default (Normal), 64 = Operator, 65 = Operator locked, 66 = Operator by default (Normal), 128 = Maintenance Out of Service, 129 = Programmed Out of Service (rung false), 256 = External.
Sts_bSrc	INT	Not Visible	Not Required	Output	Active selection bitmap (for HMI totem pole with command source request selection): Sts_bSrc.0: Hand, Sts_bSrc.1: Programmed Out of Service (rung false), Sts_bSrc.2: Maintenance Out of Service, Sts_bSrc.3: Maintenance, Sts_bSrc.4: Override, Sts_bSrc.5: External, Sts_bSrc.6: Program locked, Sts_bSrc.7: Program, Sts_bSrc.8: Operator locked, Sts_bSrc.9: Operator.
Sts_Available	BOOL	Not Visible	Not Required	Output	1 = Motor has been acquired by Program and is now available for start/stop control.
Sts_IntlkAvailable	BOOL	Not Visible	Not Required	Output	1 = Motor can be acquired by Program and is available for start/stop control when interlocks are OK.
Sts_Bypass	BOOL	Not Visible	Not Required	Output	1 = Bypassable interlocks are bypassed.
Sts_BypActive	BOOL	Visible	Not Required	Output	1 = Interlock bypassing active (bypassed or maintenance).
Sts_MaintByp	BOOL	Not Visible	Not Required	Output	1 = Device has a maintenance bypass function active.
Sts_NotRdy	BOOL	Not Visible	Not Required	Output	1 = Device is not ready, see detail bits (Sts_Nrdyxxx) for reason.
Sts_NrdyCfgErr	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: Configuration error.
Sts_NrdyDvcNotReady	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: Intelligent motor controller Not Ready.
Sts_NrdyFail	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: Device failure (Shed requires Reset).
Sts_NrdyIntlk	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: Interlock not OK.
Sts_NrdyIOFault	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: IO Fault (Shed requires Reset).
Sts_NrdyOoS	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: Device disabled by Maintenance.
Sts_Nrdy1Perm	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: Permissive 1 not OK.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_Nrdy2Perm	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: Permissive 2 not OK.
Sts_NrdyPrioStop	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: Operator or External priority Stop command requires reset.
Sts_NrdyTrip	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: Device Tripped, intelligent motor controller fault requires Reset.
Sts_Err	BOOL	Visible	Not Required	Output	1 = Error in configuration: See detail bits (Sts_Errxxx) for reason.
Sts_ErrAlm	BOOL	Not Visible	Not Required	Output	1 = Error in Logix tag-based alarm settings.
Sts_ErrPauseTime	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Invalid Pause time: use 0 to 2147483.
Sts_ErrVirtualFdbkTime	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Virtual feedback echo time: use 0 to 2147483.
Sts_ErrFailToStartTime	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Fail to Start timer preset: use 0 to 2147483.
Sts_ErrFailToStopTime	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Fail to Stop timer preset: use 0 to 2147483.
Sts_ErrResetPulseTime	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Reset Pulse timer preset: use 0 to 2147483.
Sts_ErrMaxJogTime	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Maximum Jog Time timer preset: use 0 to 2147483.
Sts_Hand	BOOL	Visible	Not Required	Output	1 = Hand is selected (supersedes OoS, Maintenance, Override, External, Program, Operator).
Sts_OoS	BOOL	Visible	Not Required	Output	1 = Out of Service is selected (supersedes Maintenance, Override, External, Program, Operator).
Sts_Maint	BOOL	Visible	Not Required	Output	1 = Maintenance is selected (supersedes Override, External, Program, Operator).
Sts_Ovrd	BOOL	Visible	Not Required	Output	1 = Override is selected (supersedes External, Program, Operator).
Sts_Ext	BOOL	Visible	Not Required	Output	1 = External is selected (supersedes Program and Operator).
Sts_Prog	BOOL	Visible	Not Required	Output	1 = Program is selected.
Sts_ProgLocked	BOOL	Not Visible	Not Required	Output	1 = Program is selected and Locked.
Sts_Oper	BOOL	Visible	Not Required	Output	1 = Operator is selected.
Sts_OperLocked	BOOL	Not Visible	Not Required	Output	1 = Operator is selected and Locked.
Sts_ProgOperSel	BOOL	Not Visible	Not Required	Output	Program/Operator selection (latch) state: 1 = Program, 0 = Operator.
Sts_ProgOperLock	BOOL	Visible	Not Required	Output	Program/Operator lock (latch) state, 1 = Locked, 0 = Unlocked.
Sts_Normal	BOOL	Not Visible	Not Required	Output	1 = Selection equals the Normal (Program or Operator).
Sts_ExtReqInh	BOOL	Not Visible	Not Required	Output	1 = External request inhibited, cannot get to External from current state.
Sts_ProgReqInh	BOOL	Not Visible	Not Required	Output	1 = Program request inhibited, cannot get to Program from current state.
Sts_MAcqRcvd	BOOL	Not Visible	Not Required	Output	1 = Program request inhibited, cannot get to Program from current state.
Sts_CmdConflict	BOOL	Not Visible	Not Required	Output	1 = Conflicting commands received this scan.
Sts_Alm	BOOL	Not Visible	Not Required	Output	1 = An alarm is active.
Sts_AlmInh	BOOL	Not Visible	Not Required	Output	1 = An alarm is shelved or disabled.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_IOFault	BOOL	Not Visible	Not Required	Output	I/O Fault status: 1 = Bad, 0 = OK. There is a predefined default discrete Logix tag-based alarm for the status. Set standard configuration members of the discrete Logix tag-based alarm. Alarm elements can be accessed as follows: PMTRTag.@Alarms.Alm_IOFault.AlarmElement.
Sts_FailToStart	BOOL	Not Visible	Not Required	Output	1 = Motor failed to Start. There is a predefined default discrete Logix tag-based alarm for the status. Set standard configuration members of the discrete Logix tag-based alarm. Alarm elements can be accessed as follows: PMTRTag.@Alarms.Alm_FailToStart.AlarmElement .
Sts_FailToStop	BOOL	Not Visible	Not Required	Output	1 = Motor failed to Stop. There is a predefined default discrete Logix tag-based alarm for the status. Set standard configuration members of the discrete Logix tag-based alarm. Alarm elements can be accessed as follows: PMTRTag.@Alarms.Alm_FailToStop.AlarmElement.
Sts_IntlkTrip	BOOL	Not Visible	Not Required	Output	1 = Motor stopped by an interlock Not OK. There is a predefined default discrete Logix tag-based alarm for the status. Set standard configuration members of the discrete Logix tag-based alarm. Alarm elements can be accessed as follows: PMTRTag.@Alarms.Alm_IntlkTrip.AlarmElement.
Sts_MotorFault	BOOL	Not Visible	Not Required	Output	1 = Intelligent Motor Control Fault. See control device display or user manual.
Sts_RdyAck	BOOL	Not Visible	Not Required	Output	1 = An alarm is ready to be acknowledged.
Sts_RdyReset	BOOL	Not Visible	Not Required	Output	1 = A latched alarm or shed condition is ready to be reset.
XRdy_Acq	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Acq, enable HMI button.
XRdy_Rel	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Rel, enable HMI button.
XRdy_Start1	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Start1, enable button.
XRdy_Start2	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Start2, enable button.
XRdy_Jog1	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Jog1, enable button.
XRdy_Jog2	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Jog2, enable button.
XRdy_Stop	BOOL	Visible	Not Required	Output	1 = Ready for XCmd_Stop, enable button.
XRdy_Reset	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Reset, enable button.
XRdy_ResetAckAll	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_ResetAckAll, enable button.
Val_Owner	DINT	Not Visible	Not Required	Output	Current object owner ID (0 = not owned).

Private Input Members	Data Type	Description
HMI_BusObjIndex	DINT	This object's index in the bus array, for use by HMI display. Default is 0.
MCmd_Acq	BOOL	Maintenance command to acquire ownership (Operator/Program/External/Override to Maintenance). The instruction clears this operand automatically. Default is false.
MCmd_Bypass	BOOL	Maintenance command to bypass all bypassable interlocks and permissives. The instruction clears this operand automatically. Default is false.

Private Input Members	Data Type	Description
MCmd_Check	BOOL	Maintenance command to check (not bypass) all interlocks and permissives. The instruction clears this operand automatically. Default is false.
MCmd_IS	BOOL	Maintenance command to select In Service. The instruction clears this operand automatically. Default is false.
MCmd_OoS	BOOL	Maintenance command to select Out of Service. The instruction clears this operand automatically. Default is false.
MCmd_Physical	BOOL	Maintenance command to select Physical device operation (not simulated). The instruction clears this operand automatically. Default is false.
MCmd_Virtual	BOOL	Maintenance command to select Virtual (simulated) device operation. The instruction clears this operand automatically. Default is false.
OCmd_Jog1	BOOL	Operator command to Jog Motor Forward / Slow. The instruction clears this operand automatically if max jog time is reached. Default is false.
OCmd_Jog2	BOOL	Operator command to Jog Motor Reverse / Fast. The instruction clears this operand automatically if max jog time is reached. Default is false.
OCmd_Lock	BOOL	Operator command to lock Operator (disallow Program). The instruction clears this operand automatically. Default is false.
OCmd_Normal	BOOL	Operator command to select Normal (Operator or Program). The instruction clears this operand automatically. Default is false.
OCmd_Oper	BOOL	Operator command to select Operator (Program to Operator). The instruction clears this operand automatically. Default is false.
OCmd_Prog	BOOL	Operator command to select Program (Operator to Program). The instruction clears this operand automatically. Default is false.
OCmd_Reset	BOOL	Operator command to reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.
OCmd_ResetAckAll	BOOL	Operator command to acknowledge and reset all alarms and latched shed conditions. The use of OCmd_ResetAckAll is restricted to HMI. The instruction clears this operand automatically. Default is false.
OCmd_Start1	BOOL	Operator command to Start Motor Forward / Slow. The instruction clears this operand automatically. Default is false.
OCmd_Start2	BOOL	Operator command to Start Motor Reverse / Fast. The instruction clears this operand automatically. Default is false.
OCmd_Stop	BOOL	Operator command to Stop Motor. The instruction clears this operand automatically. Default is false.

Private Output Members	Data Type	Description
MRdy_Acq	BOOL	1 = Ready for MCmd_Acq, enable HMI button.
MRdy_Bypass	BOOL	1 = Ready for MCmd_Bypass, enable HMI button.

Private Output Members	Data Type	Description
MRdy_Check	BOOL	1 = Ready for MCmd_Check, enable HMI button.
MRdy_IS	BOOL	1 = Ready for MCmd_IS, enable HMI button.
MRdy_OoS	BOOL	1 = Ready for MCmd_OoS, enable HMI button.
MRdy_Physical	BOOL	1 = Ready for MCmd_Physical, enable HMI button.
MRdy_Rel	BOOL	1 = Ready for MCmd_Rel, enable HMI button.
MRdy_Virtual	BOOL	1 = Ready for MCmd_Virtual, enable HMI button.
ORdy_Jog1	BOOL	1 = Ready for OCmd_Jog1, enable HMI button.
ORdy_Jog2	BOOL	1 = Ready for OCmd_Jog2, enable HMI button.
ORdy_Lock	BOOL	1 = Ready for OCmd_Lock, enable HMI button.
ORdy_Normal	BOOL	1 = Ready for OCmd_Normal, enable HMI button.
ORdy_Oper	BOOL	1 = Ready for OCmd_Oper, enable HMI button.
ORdy_Prog	BOOL	1 = Ready for OCmd_Prog, enable HMI button.
ORdy_Reset	BOOL	1 = A latched alarm or shed condition is ready to be reset.
ORdy_ResetAckAll	BOOL	1 = A latched alarm or shed condition is ready to be reset or acknowledged.
ORdy_Start1	BOOL	1 = Ready for OCmd_Start1, enable HMI button.
ORdy_Start2	BOOL	1 = Ready for OCmd_Start2, enable HMI button.
ORdy_Stop	BOOL	1 = Ready for OCmd_Stop, enable HMI button.
ORdy_Unlock	BOOL	1 = Ready for OCmd_Unlock, enable HMI button.
Sts_FaultDesc	STRING	Description of intelligent motor controller fault, lookup from last fault code.

Public InOut Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Ref_Ctrl_Set	RAC_ITF_DVC_PWRDISCRETE_SET	Visible	Required	InOut	Discrete Power Automation Device Object Settings Interface.
Ref_Ctrl_Cmd	RAC_ITF_DVC_PWRDISCRETE_CMD	Visible	Required	InOut	Discrete Power Automation Device Object Command Interface.
Ref_Ctrl_Sts	RAC_ITF_DVC_PWRDISCRETE_STS	Visible	Required	InOut	Discrete Power Automation Device Object Status Interface.
BusObj	BUS_OBJ	Visible	Required	InOut	Bus component
Ref_FaultCodeList	RAC_CODE_DESCRIPTION[1]	Visible	Required	InOut	Fault Code to Fault Description lookup table for intelligent motor controller.

RAC_ITF_DVC_PWRDISCRETE_SET Structure

The RAC_ITF_DVC_PWRDISCRETE_SET structure is the first of three structures exchanged with the associated Power Discrete Device Object to interface with the intelligent motor control device. This structure handles settings, such as the Command Inhibit, sent to the motor controller.

This parameter is used to link the instruction to an external tag that contains necessary data for the instruction to operate. The external tag must be of the data type shown, or may be NULL. If NULL, other pins on the instruction are used to link the necessary data.

Members	Data Type	Description
InhibitCmd	BOOL	1=Inhibit user Commands from external sources; 0=AllowControl
InhibitSet	BOOL	1=Inhibit user Settings from external sources; 0=Allow

RAC_ITF_DVC_PWRDISCRETE_CMD Structure

The RAC_ITF_DVC_PWRDISCRETE_CMD structure is the second of three structures exchanged with the associated Power Discrete Device Object to interface with the intelligent motor control device. This structure handles commands, such as the start, stop, direction, and speed, sent to the motor controller. It is an InOut parameter configured as optional (May Be Null).

This parameter is used to link the instruction to an external tag that contains necessary data for the instruction to operate. The external tag must be of the data type shown, or may be NULL. If NULL, other pins on the instruction are used to link the necessary data.

Members	Data Type	Description
bCmd	INT	Commands (Bit Overlay)
Physical	BOOL	Operate as a physical device
Virtual	BOOL	Operate as a virtual device
ResetWarn	BOOL	Reset Warning Status
ResetFault	BOOL	Reset Fault Status
Activate	BOOL	Activate Output Power Structure
Deactivate	BOOL	DeActivate Output Power Structure
CmdDir	BOOL	Command Direction; 0=Forward, 1=Reverse
Jog	BOOL	Jog Command
Fast	BOOL	Fast speed of a two-speed device
Slow	BOOL	Slow Speed of a two-speed device

RAC_ITF_DVC_PWRDISCRETE_STS Structure

The RAC_ITF_DVC_PWRDISCRETE_STS structure is the third of three structures exchanged with the associated Power Discrete Device Object to interface with the intelligent motor control device. This structure handles status, such as the run feedback, connection status, commanded and actual direction, received from the motor controller. It is an InOut parameter configured as optional (May Be Null).

This parameter is used to link the instruction to an external tag that contains necessary data for the instruction to operate. The external tag must be of the data type shown, or may be NULL. If NULL, other pins on the instruction are used to link the necessary data.

Members	Data Type	Description
eState	DINT	Enumerated state value: 0=Unused 1=Initializing 2=Disconnected 3=Disconnecting, 4=Connecting, 5=Idle, 6=Configuring, 7=Available
FirstWarning	RAC_UDT_EVENT	First Warning
FirstFault	RAC_UDT_EVENT	First Fault
eCmdFail	DINT	Enumerated command failure code
bSts	INT	Status (Bit Overlay)
Physical	BOOL	1=Operating as a physical device
Virtual	BOOL	1=Operating as a virtual device
Connected	BOOL	Connected Status
Available	BOOL	Available Status
Warning	BOOL	Warning
Faulted	BOOL	Faulted
Ready	BOOL	1=Device is active (Power Structure Active)
Active	BOOL	1=Device is active (Power Structure Active)
CmdDir	BOOL	Command direction; 1=Reverse, 0=Forward
ActDir	BOOL	Actual direction; 1=Reverse, 0=Forward
CmdSpd	BOOL	1 = Motor is Commanded to run fast 0 = slow
Fast	BOOL	Fast speed selected (two-speed device)
Slow	BOOL	Slow speed selected (two-speed device)

BUS_OBJ Structure

The BUS_OBJ structure is used to link the motor to other devices and instructions in a complex control strategy, typically into a hierarchy. A Bus Object rolls up status and alarm information from lower level devices to higher level control and fans out commands from higher level control to lower level devices, and items link to the bus by referencing a single member of the BUS_OBJ array associated with the bus.

This parameter is used to link the instruction to an external tag that contains necessary data for the instruction to operate. The external tag must be of the data type shown, or may be NULL. If NULL, the Bus functions of this instruction are not available.

Members	Data Type	Description
Inp_Cmd	DINT	Input to assert commands
Out_Cmd	DINT	Resultant commands
Inp_CmdLLH	DINT	Input for level normally high commands
Out_CmdLLH	DINT	Resultant line level high commands
Inp_Sts	DINT	Input to assert status
Out_Sts	DINT	Resultant status

Members	Data Type	Description
Inp_CmdAck	DINT	Input to assert a command acknowledgement
Out_CmdAck	DINT	Resultant command acknowledgements
Inp_SeverityMax	DINT	Input: maximum alarm severity
Out_SeverityMax	DINT	Resultant of maximum alarm severity
Cfg_CmdMask	DINT	Propagation mask for commands
Cfg_CmdLLHMask	DINT	Propagation mask for line level high commands
Cfg_StsMask	DINT	Propagation mask for status
Ref_Index	DINT	Bus array index

RAC_CODE_DESCRIPTION[x] Structure

The RAC_CODE_DESCRIPTION[x] structure is an array of intelligent motor controller fault code number and fault code description pairs, used as a lookup table. The instruction searches the table for the fault code received from the motor controller and displays the corresponding fault description text.

This parameter is used to link the instruction to an external tag that contains necessary data for the instruction to operate. The external tag must be of the data type shown, or may be NULL. If NULL, the fault code lookup function is not performed. Fault descriptions will only be shown if provided through the Device Object Status interface.

Members	Data Type	Description
Code	DINT	Code for which to look up Description
Desc	STRING	Description for given Code

RAC_EVENTStructure

RAC_EVENTstructures are used by the FirstFault and FirstWarning members in the RAC_ITF_DVC_PWRDISCRETE_STS structure. These items hold the event data received from the intelligent motor controller for the first fault and first warning records in the motor controller event history.

Members	Data Type	Description
Type	DINT	1 = Status 2 = Warning 3 = Fault 4 ...n = User
ID	DINT	User definable event ID
Category	DINT	User definable category (Electrical,Mechanical,Materials,Utility,etc.)
Action	DINT	User definable event action code
Value	DINT	User definable event value or fault code
Message	STRING	Event message text
EventTime_L	LINT	Timestamp
EventTime_D	DINT[7]	Timestamp (Y,M,D,h,m,s,us)

Alarms

Discrete Logix tag-based alarms are defined for these members.

Member	Alarm Name	Description
Sts_FailToStart	Alm_FailToStart	Motor failed to start within the allotted time when commanded to start
Sts_FailToStop	Alm_FailToStop	Motor failed to stop within the allotted time when commanded to stop
Sts_IntlkTrip	Alm_IntlkTrip	Motor stopped by an Interlock Not OK
Sts_IOFault	Alm_IOFault	Motor communication with controller failed
Sts_MotorFault	Alm_MotorFault	The motor controller is reporting it has a fault condition

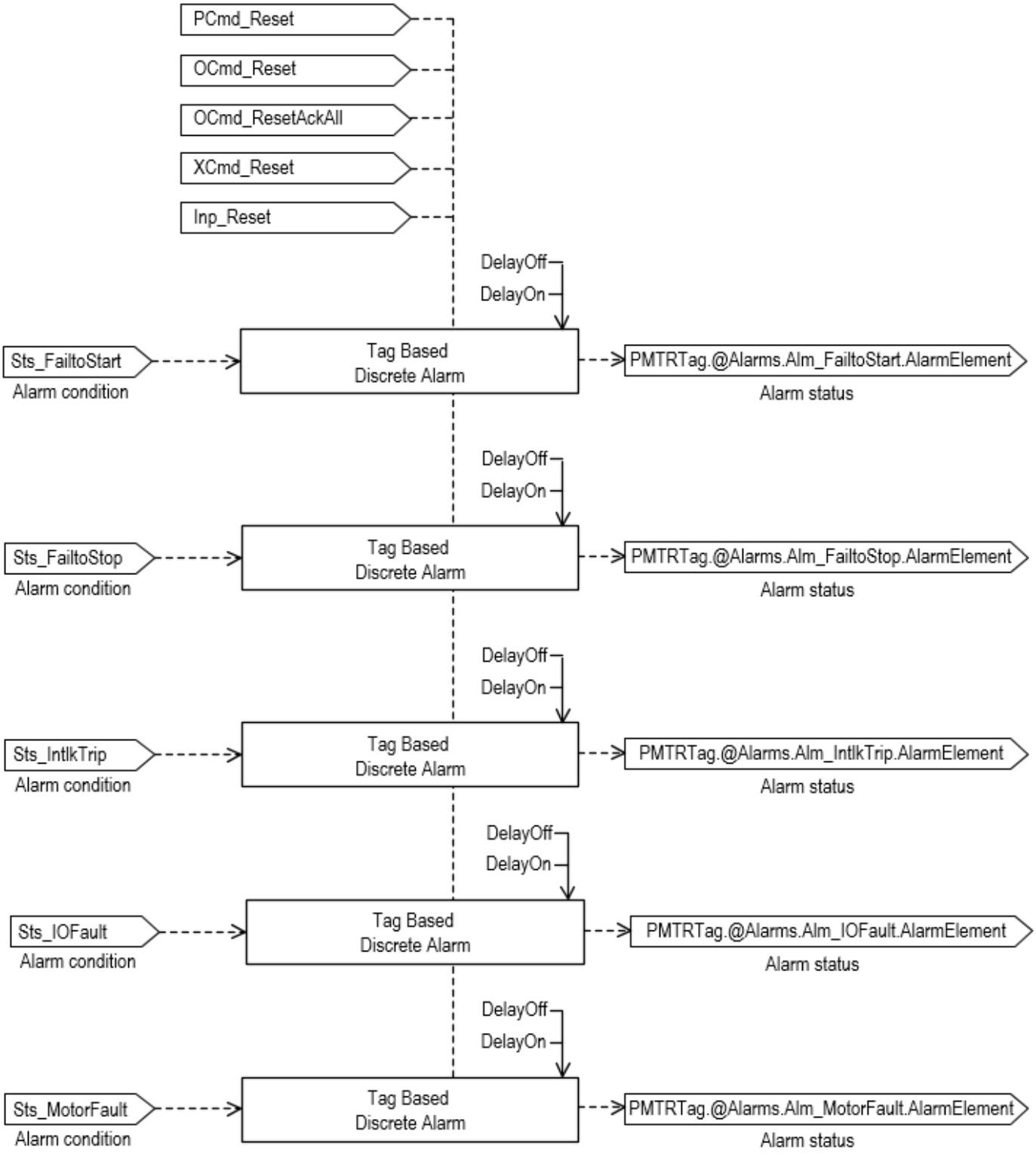
Mark the alarm as used or unused and set standard configuration members of the discrete tag-based alarm. Use this format to access alarm elements:

Tag.@Alarms.AlarmName.AlarmElement

The PMTR instruction uses these alarms:

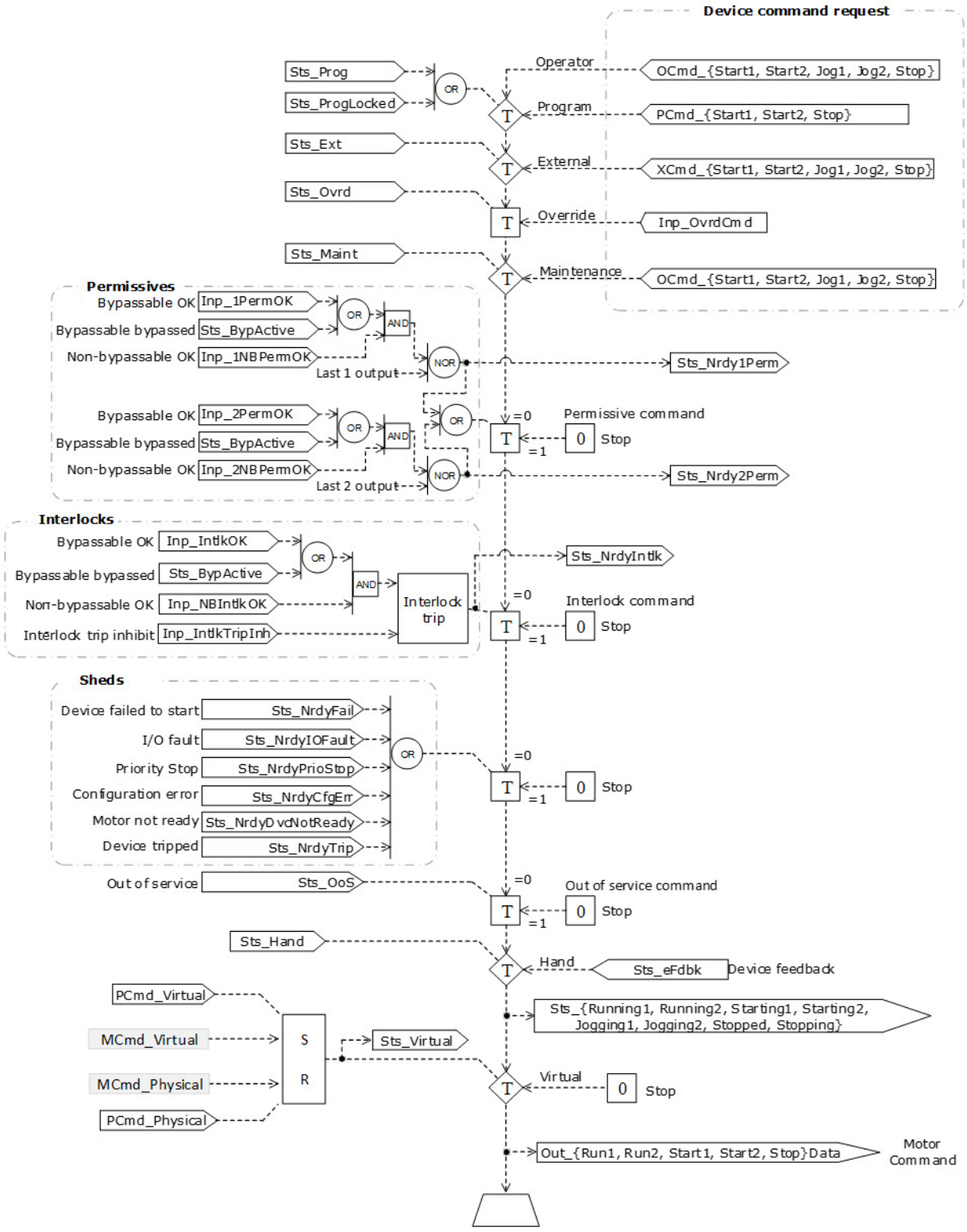
- Raises the Fail to Start alarm when the motor is commanded to start but run feedback is not received within the configured failure time.
- Raises the Fail to Stop alarm when the motor is commanded to stop but run feedback does not drop within the configured failure time.
- Raises the Interlock Trip alarm when the motor is running and an interlock not-OK condition causes the motor to stop. If interlocks are not bypassed, a bypassable interlock or a non-bypassable interlock not-OK condition initiates an interlock trip. If interlocks are bypassed, only a non-bypassable interlock not-OK condition initiates an interlock trip.
- Raises the I/O Fault alarm when I/O communication with the motor controller or other I/O device is lost. For the Power Discrete Device interface, this is detected when the Ref_Ctrl_Sts.Connected bit goes false (to 0). For the discrete signal interface, used when Ref_Ctrl_Sts is NULL, this is detected when Inp_IOFault goes true (to 1).
- Raises the Motor Fault alarm when the motor controller reports a faulted condition. For the Power Discrete Device interface, this is detected when the Ref_Ctrl_Sts.Faulted bit goes true (to 1). For the discrete signal interface, which is used when Ref_Ctrl_Sts is NULL, this is detected when Inp_Faulted goes true (to 1).

Program, Operator, and External commands enable the Reset of latched alarms, and Reset & Acknowledge of all alarms of the instruction (Alarm Set) at the same time. This diagram shows how the commands interact with the PMTR instruction.



Operation

This diagram illustrates functionality of the PMTR instruction:



Operator command request confirmation

The PMTR instruction enables these operator command requests:

- OCmd_Jog1
- OCmd_Jog2

- OCmd_Start1
- OCmd_Start2
- OCmd_Stop

Enforced security might require the request to be confirmed or canceled before the selected command executes. The instruction checks the security rules inspecting Cfg_CnfrmReqd. If Cfg_CnfrmReqd=0 no confirmation is required and the request executes immediately. If Cfg_CnfrmReqd=1 the instruction waits for confirmation OCmd_CmdCnfrm=1 and/or cancellation. For Cfg_CnfrmReqd=2 or 3, eSignature is needed before the confirmation and cancellation is enabled.

For more information on command sources, see *Process Motor (PMTR) Command Source*.

Virtualization

Use virtualization for instruction testing and operator training. Command to virtual operation using program command PCmd_Virtual or maintenance command MCmd_Virtual. After finishing virtual operation, use program command PCmd_Physical or maintenance command MCmd_Physical to return to normal physical device operation.

When Virtualization is active, the outputs of the PMTR instruction hold at 0, virtual feedback of a working device is provided, and I/O faults are ignored. The setting of Cfg_VirtualFdbkTime operand determines the time delay between a command to start or stop the device and the echo of the running or stopped status. Manipulate the instruction to operate as if a working motor is present.

Initialization

The instruction is normally initialized in the instruction first run. Request re-initialization by setting Inp_InitializeReq = 1. For proper initialization, when adding the instruction while performing an online edit of the code, make sure that Inp_InitializeReq = 1, the default value.

Configuration of Strings for HMI

Configure strings for HMI faceplates (FactoryTalk View) and for the Logix Designer configuration dialog box. The strings are set to extended properties of tag items.

- Description
- Label for graphic symbol
- Display Library for HMI Faceplate call-up

- Instruction name
- Area name
- URL link
- More Information
- Running 1 Text
- Running 2 Text
- Name of Fail to Start Alarm
- Name of Fail to Stop Alarm
- Name of Interlock Trip Alarm
- Name of I/O Fault Alarm
- Name of Motor Fault Alarm

Implementation

The PMTR instruction monitors and controls a discrete motor. The start, stop and jog commands to the motor can come from a variety of sources, determined by an embedded instance of PCMDSRC. Available command sources are:

- Operator, through the HMI
- Program, through logic connected to the block
- External, through logic connected to the block
- Override, through logic connected to the block
- Maintenance, through the HMI
- Out of Service
- Hand (assumes the block has no control of the motor, so aligns with the actual motor status in order to achieve bumpless transfer from Hand back to one of the other command sources)

The PMTR instruction has two aspects, which can be kept by a particular command source whenever the command source selection is Operator, Program or External. Either or both of the aspects can be kept at any given time, or can follow the selection of the PCMDSRC. The aspects are:

- Start1 and Start2 commands
- Jog1 and Jog2 commands

The Jog commands cannot be kept by the Program command source.

The PMTR instruction supports virtualization. When selected to Virtual, the instruction provides status to the operator and other blocks as if a working motor were connected while keeping the outputs to the physical motor de-energized (zero). When selected to Physical, the instruction monitors and controls the physical motor device. Use Virtualization to provide off-process functional testing of higher-level control strategies or simulation for operator training.

The PMTR instruction supports interlocks, conditions that must be OK for the motor to run and which stop the motor if not OK, and permissives, conditions

that must be OK for the motor to start but which are ignored once the motor is running. Bypassable permissives and interlocks can be bypassed for maintenance, while non-bypassble interlocks and permissives are always evaluated.

The PMTR instruction supports a bus for forwarding commands (fanout) and gathering status (rollup) in a hierarchy of objects. Refer to the Bus Object for more information on the commands and status (including alarm status) sent on the bus.

The PMTR instruction optionally supports the ability to look up the text to display for the most recent intelligent motor controller fault code, given a provided fault code lookup table. This table is an array of Code and Description pairs and is searched whenever the last fault code from the motor controller changes.

The PMTR instruction interface to the physical motor can be through a Power Discrete Device Object interface or by connecting individual motor controller signals to input and output pins of the instruction. Details on the Power Discrete Device Object interface are given below. Three interface tags are used, provided as InOut Parameters. These tags provide motor Settings, motor Commands, such as start forward, jog reverse and stop, and retrieve motor Status, such as connected, active (running), commanded direction and speed, actual direction and speed, warning, faulted, and extended motor controller warning and fault information.

PMTR Motor Settings: Ref_Ctrl_Set InOut Parameter (RAC_ITF_DVC_PWRDISCRETE_SET) Structure

Private Input Members	Data Type	Description
InhibitCmd	BOOL	1 = Inhibit user Commands from external sources; 0 = Allow control.
InhibitSet	BOOL	1 = Inhibit user Settings from external source, 0 = Allow.

PMTR Motor Commands: Ref_Ctrl_Cmd InOut Parameter (RAC_ITF_DVC_PWRDISCRETE_CMD) Structure

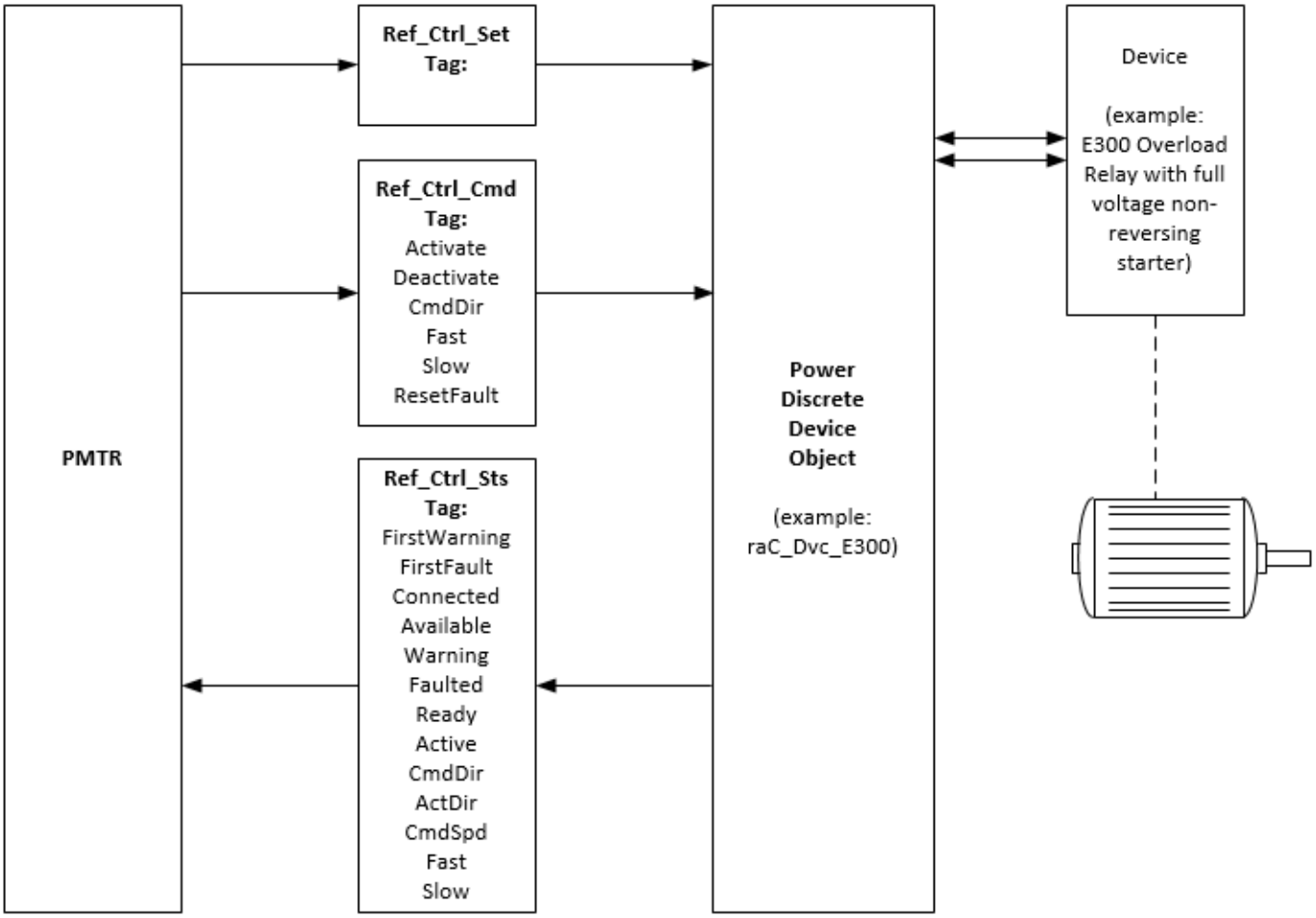
Private Input Members	Data Type	Description
bCmd	INT	Commands (bit overlay), consisting of:
Physical	BOOL	Operate as a Physical device
Virtual	BOOL	Operate as a Virtual device
ResetWarn	BOOL	Reset Warning status
ResetFault	BOOL	Reset Fault status
Activate	BOOL	Activate Output Power Structure (if speed reference is not zero, the motor will run)
Deactivate	BOOL	Deactivate Output Power Structure (motor will stop)
CmdDir	BOOL	Commanded Direction, 0 = Forward
Jog	BOOL	Jog Command
Fast	BOOL	Fast speed of a two-speed device

Private Input Members	Data Type	Description
Slow	BOOL	Slow Speed of a two-speed device

PMTR Motor Status: Ref_Ctrl_Sts InOut Parameter (RAC_ITF_DVC_PWRDISCRETE_STS) Structure

Private Input Members	Data Type	Description
eState	DINT	Enumerated state value: 0 = unused, 1 = Initializing, 2 = Disconnected, 3 = Disconnecting, 4 = Connecting, 5 = Idle, 6 = Configuring, 7 = Available
FirstWarning	RAC_ITF_EVENT	First Warning, consisting of:
Type	DINT	1 = Status, 2 = Warning, 3 = Fault, 4 ... n = User
ID	DINT	User definable event ID
Category	DINT	User definable category (electrical, mechanical, materials, utility, etc.)
Action	DINT	User definable event action code
Value	DINT	User definable event value or fault code
Message	STRING	Event message text
EventTime_L	LINT	Timestamp
EventTime_D	DINT[7]	Timestamp (Yr, Mo, Da, Hr, Min, Sec, Microsec)
FirstFault	RAC_ITF_EVENT	First Fault, consisting of:
Type	DINT	1 = Status, 2 = Warning, 3 = Fault, 4 ... n = User
ID	DINT	User definable event ID
Category	DINT	User definable category (electrical, mechanical, materials, utility, etc.)
Action	DINT	User definable event action code
Value	DINT	User definable event value or fault code
Message	STRING	Event message text
EventTime_L	LINT	Timestamp
EventTime_D	DINT[7]	Timestamp (Yr, Mo, Da, Hr, Min, Sec, Microsec)
eCmdFail	DINT	Enumerated command failure code
bSts	INT	Status, consisting of:
Physical	BOOL	1 = Operating as a physical device
Virtual	BOOL	1 = Operating as a virtual device
Connected	BOOL	Connected status
Available	BOOL	Available status
Warning	BOOL	Device Warning
Faulted	BOOL	Device Faulted
Ready	BOOL	1 = Device is ready (can be activated)
Active	BOOL	1 = Device is active (power structure active, running)
CmdDir	BOOL	Commanded direction: 1 = reverse, 0 = forward
ActDir	BOOL	Actual direction (of rotation): 1 = reverse, 0 = forward
CmdSpd	BOOL	1 = Motor is Commanded to run fast 0 = slow
Fast	BOOL	Fast speed selected (two-speed device)
Slow	BOOL	Slow speed selected (two-speed device)

This illustration shows the relationship between a PMTR instance and its associated Power Discrete Device Object.



Monitor the PMTR Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See *Index Through Arrays* for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Any commands received before first scan are discarded. The motor is de-energized and treated as if it were commanded to stop.
Instruction first run	Any commands received before first scan are discarded. The motor state is evaluated and the instruction aligns with the current state of the motor, as if the Hand command source were selected.
Rung-condition-in is false	Handled the same as if the motor is taken Out of Service by command. The motor outputs are de-energized, and the motor Command Source is shown as Program Out of Service on the HMI. All alarms are cleared. The rung-condition-out continues as false.
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	Any commands received before first scan are discarded. The motor is de-energized and treated as if it were commanded to stop.
Instruction first run	Any commands received before first scan are discarded. The motor state is evaluated and the instruction aligns with the current state of the motor, as if the Hand command source were selected.
Instruction first scan	See instruction first run in the function block diagram table.
EnableIn is false	Handled the same as if the motor is taken Out of Service by command. The motor outputs are de-energized, and the motor Command Source is shown as Program Out of Service on the HMI. All alarms are cleared. EnableOut is set to false.
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

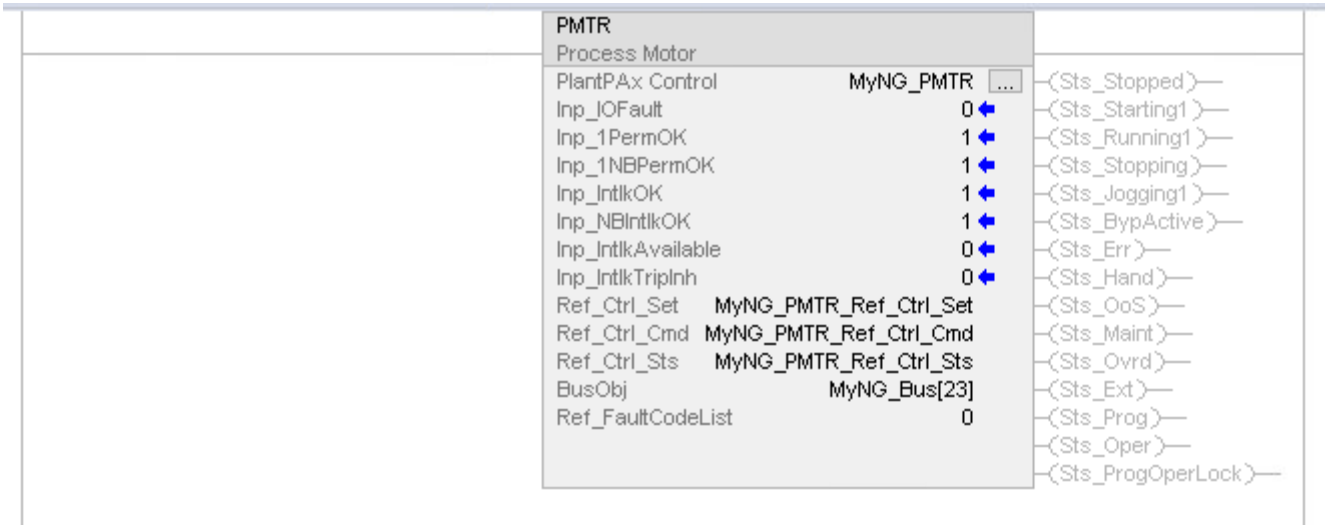
Structured Text

In Structured Text, EnableIn is always true during normal scan. Therefore, if the instruction is in the control path activated by the logic it will execute.

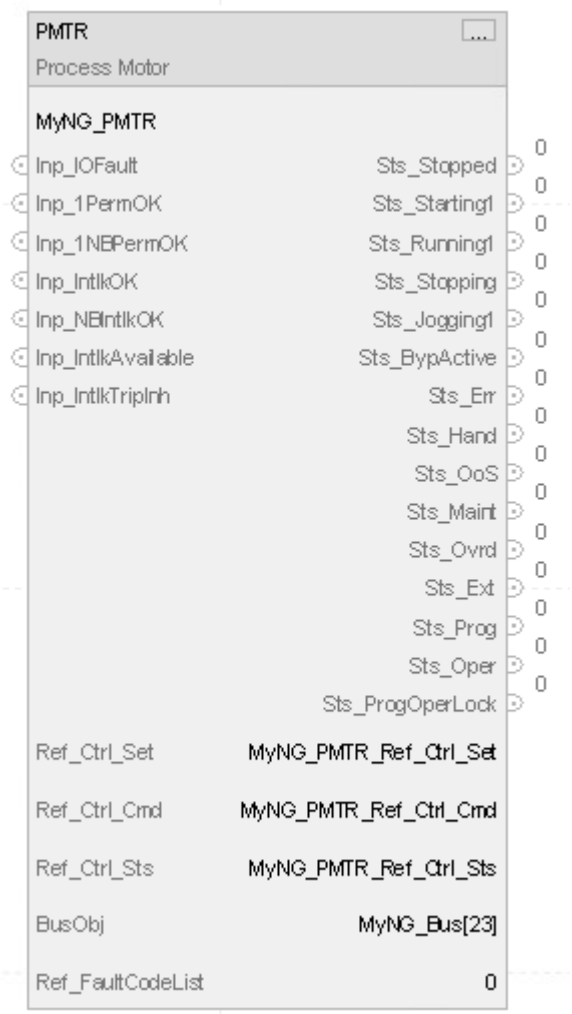
Condition/State	Action Taken
Prescan	See Prescan in the Function Block Diagram table.
Instruction first run	See Instruction first run in the Function Block Diagram table.
EnableIn is true	See EnableIn is true in the Function Block Diagram table.
Postscan	See Postscan in the Function Block Diagram table.

Example

Ladder Diagram



Function Block Diagram



Structured Text

PMTR(MyNG_PMTR, MyNG_PMTR_Ref_Ctrl_Set, MyNG_PMTR_Ref_Ctrl_Cmd, MyNG_PMTR_Ref_Ctrl_Sts, MyNG_Bus[23], 0)

See also

Process Motor (PMTR) Command Source on page 609

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Data Conversions on page 1086

Process Motor (PMTR) Command Source

The Process Motor (PMTR) instruction uses these command sources. The command sources are prioritized in order from highest to lowest in this table.

Command Source	Description
Hand	Hardwired logic or other logic outside the instruction controls the device. The instruction tracks the state of the device for bumpless transfer back to one of the other command sources. This is the highest priority command source.
Out-of-Service	The instruction is disabled and accepts no device commands.
Maintenance	Maintenance controls the device and supersedes Operator, Program, External and Override control. Operator commands and settings from the HMI are accepted.
Override	Priority logic controls the device and supersedes Operator, Program and External control. Override Input (Inp_Ovrd) is accepted.
External	External logic (for exapmle, field pilot control or upstream SCADA) controls the device. External commands (XCmd_) are accepted.
Program locked	Program logic controls the device. Program commands (PCmd_) are accepted. Operator cannot take control from the Program. Override cannot take control from theProgram unless Cfg_OvrdOverLock = 1.
Program	Program logic controls the device. Program commands (PCmd_) are accepted.
Operator locked	The Operator controls the device. Operator commands (OCmd_) from the HMI are accepted. Program cannot take control from Operator. Override cannot take control from Operator unless Cfg_OvrdOverLock = 1.
Operator	The Operator controls the device. Operator commands (OCmd_) from the HMI are accepted. This is the lowest priority command source.

The instruction enables or disables these operations:

- Maintenance Out-of-Service exists
- Maintenance exists
- External exists
- Program (unlocked) exists
- Program locked exists
- Operator (unlocked) exists
- Operator locked exists
- PCmd_Lock used as a Level (1 = Lock, 0 = Unlock)
- XCmd_Acq used as a Level (1 = Acquire, 0 = release)

The instruction checks for invalid configurations of control and forces the nearest valid configuration.

The core control model arbitrates the source of the commands and parameters that the receiving function accepts. The core control model determines if the source is:

- A programmatic entity which resides entirely within the processing environment, or
- An external interface entity which issues commands and parameters external and asynchronously to the processing environment.

Locking a control source prevents the other control source from acquiring privilege.

Core Command Source Model

The core control model consists of these control sources:

- Oper
- OperLocked
- Prog
- ProgLocked

The control model defaults to this configuration. Other control sources may be present in the model but act as overriding control sources, acting independent of the base Operator/Program state machine.

Enable control sources as Configuration

The user can enable and disable individual control sources. The default configuration uses the entire base model; upon power-up of the processing environment the control source will be the designated default. Some combinations of enabled control sources are disallowed as they are either unnecessary or could create unintended changes.

Prog Power Up

Configuration allows the user to specify whether Operator or Program is the power-up default.

Prog Priority

Configuration allows the user to specify whether Operator or Program commands take priority when simultaneously asserted.

Automatic reset of commands

Unless specifically configured as Level, above, all commands are treated as one-shot-latched (Edge). Commands are automatically cleared when the instruction executes and processes them.

Change Destination States

Under certain configurations the destination command source for some commands may change. This is in keeping with the intent of the command. For example, if the Program state is disabled, the destination of the OCmd_Prog command becomes the Program Locked state instead of the Program state. This maintains the intent of the OCmd_Prog command: the

operator entity wishes to place the function in control of the program. If the command was eliminated then there would be no way to accomplish this. This is only done in configurations where it would cause no conflict or race condition, but serves to preserve as much user functionality as is practical.

Higher Priority Command Sources

These Higher priority command sources operate independently within the model:

- External
- Override
- Maintenance
- Out-of-Service
- In-Service
- Hand

See also

Process Motor (PMTR)

Process Permissives (PPERM)

This information applies to the ControlLogix 5380P and 5580P controllers.

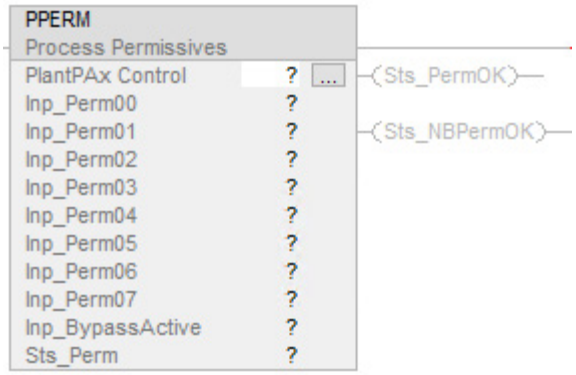
The Process Permissives (PPERM) instruction collects, or sums up, the permissive conditions that allow a piece of equipment to energize. In most cases, permissive conditions must be true to energize equipment. Once the equipment is energized, permissives are ignored.

The PPERM instruction provides:

- Permissive input OK Check. Evaluates the inputs. If all inputs are in the configured OK state, the instruction sets the All Permissives OK status to true.
- Permissive bypass. Evaluates the non-permissive inputs to bypass. If all inputs are in their configured OK state, the instruction sets the All Non-Bypassable Permissives OK status to true.
- Summary status. Summarizes its 32 permissive input conditions into two primary status bits:
  - Sts_PermOK. Indicates all permissive conditions are clear, or ready to energize.
  - Sts_NBPermOK. Indicates all permissive conditions that cannot be bypassed are clear, or ready to energize after bypassing permissive conditions.

Available Languages

Ladder Diagram



Function Block Diagram

Structured Text

PPERM (PPERM tag);

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PlantPAx Control	P_PERMISSIVE	tag	Data structure required for proper operation of instruction.

P_PERMISSIVE Structure

Public members are standard, visible tag members that are programmatically accessible. Private, hidden members are used in HMI faceplates and are not programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	Description
EnableIn	BOOL	Enable Input - System Defined Parameter Default is true.
Inp_InitializeReq	BOOL	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request when reinitializing. The instruction clears this operand automatically. Default is true.
Inp_Perm00	BOOL	Permissive condition 00, OK to energize if in configured OK state. Default is true.
Inp_Perm01	BOOL	Permissive condition 01, OK to energize if in configured OK state. Default is true.
Inp_Perm02	BOOL	Permissive condition 02, OK to energize if in configured OK state. Default is true.
Inp_Perm03	BOOL	Permissive condition 03, OK to energize if in configured OK state. Default is true.
Inp_Perm04	BOOL	Permissive condition 04, OK to energize if in configured OK state. Default is true.
Inp_Perm05	BOOL	Permissive condition 05, OK to energize if in configured OK state. Default is true.
Inp_Perm06	BOOL	Permissive condition 06, OK to energize if in configured OK state. Default is true.
Inp_Perm07	BOOL	Permissive condition 07, OK to energize if in configured OK state. Default is true.
Inp_Perm08	BOOL	Permissive condition 08, OK to energize if in configured OK state. Default is true.
Inp_Perm09	BOOL	Permissive condition 09, OK to energize if in configured OK state. Default is true.
Inp_Perm10	BOOL	Permissive condition 10, OK to energize if in configured OK state. Default is true.
Inp_Perm11	BOOL	Permissive condition 11, OK to energize if in configured OK state. Default is true.
Inp_Perm12	BOOL	Permissive condition 12, OK to energize if in configured OK state. Default is true.
Inp_Perm13	BOOL	Permissive condition 13, OK to energize if in configured OK state. Default is true.
Inp_Perm14	BOOL	Permissive condition 14, OK to energize if in configured OK state. Default is true.
Inp_Perm15	BOOL	Permissive condition 15, OK to energize if in configured OK state. Default is true.

Public Input Members	Data Type	Description
Inp_Perm16	BOOL	Permissive condition 16, OK to energize if in configured OK state. Default is true.
Inp_Perm17	BOOL	Permissive condition 17, OK to energize if in configured OK state. Default is true.
Inp_Perm18	BOOL	Permissive condition 18, OK to energize if in configured OK state. Default is true.
Inp_Perm19	BOOL	Permissive condition 19, OK to energize if in configured OK state. Default is true.
Inp_Perm20	BOOL	Permissive condition 20, OK to energize if in configured OK state. Default is true.
Inp_Perm21	BOOL	Permissive condition 21, OK to energize if in configured OK state. Default is true.
Inp_Perm22	BOOL	Permissive condition 22, OK to energize if in configured OK state. Default is true.
Inp_Perm23	BOOL	Permissive condition 23, OK to energize if in configured OK state. Default is true.
Inp_Perm24	BOOL	Permissive condition 24, OK to energize if in configured OK state. Default is true.
Inp_Perm25	BOOL	Permissive condition 25, OK to energize if in configured OK state. Default is true.
Inp_Perm26	BOOL	Permissive condition 26, OK to energize if in configured OK state. Default is true.
Inp_Perm27	BOOL	Permissive condition 27, OK to energize if in configured OK state. Default is true.
Inp_Perm28	BOOL	Permissive condition 28, OK to energize if in configured OK state. Default is true.
Inp_Perm29	BOOL	Permissive condition 29, OK to energize if in configured OK state. Default is true.
Inp_Perm30	BOOL	Permissive condition 30, OK to energize if in configured OK state. Default is true.
Inp_Perm31	BOOL	Permissive condition 31, OK to energize if in configured OK state. Default is true.
Inp_BypassActive	BOOL	1 = Permissive bypassing is currently active. Default is false.
Cfg_OKState	DINT	Bits indicate which state (0 or 1) of each input is OK to energize. Default is 2#1111_1111_1111_1111_1111_1111_1111_1111.
Cfg_Bypassable	DINT	Set bits indicate which conditions can be bypassed. Default is 2#0000_0000_0000_0000_0000_0000_0000_0000.
Cfg_HasMoreObj	BOOL	1 = Tells HMI an object with more information is available. Default is false.
Cfg_HasNav	DINT	Set bits indicate which navigation buttons are enabled. Default is 2#0000_0000_0000_0000_0000_0000_0000_0000.

Public Output Members	Data Type	Description
EnableOut	BOOL	Enable Output - System Defined Parameter
Sts_Initialized	BOOL	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.
Sts_PermOK	BOOL	Overall permissive status (1 = OK to energize).
Sts_NBPermOK	BOOL	Non-Bypassable permissive status (1 = all non-bypassable permissives OK to energize).
Sts_BypActive	BOOL	1 = Permissive bypassing is active (ignore bypassable permissives).

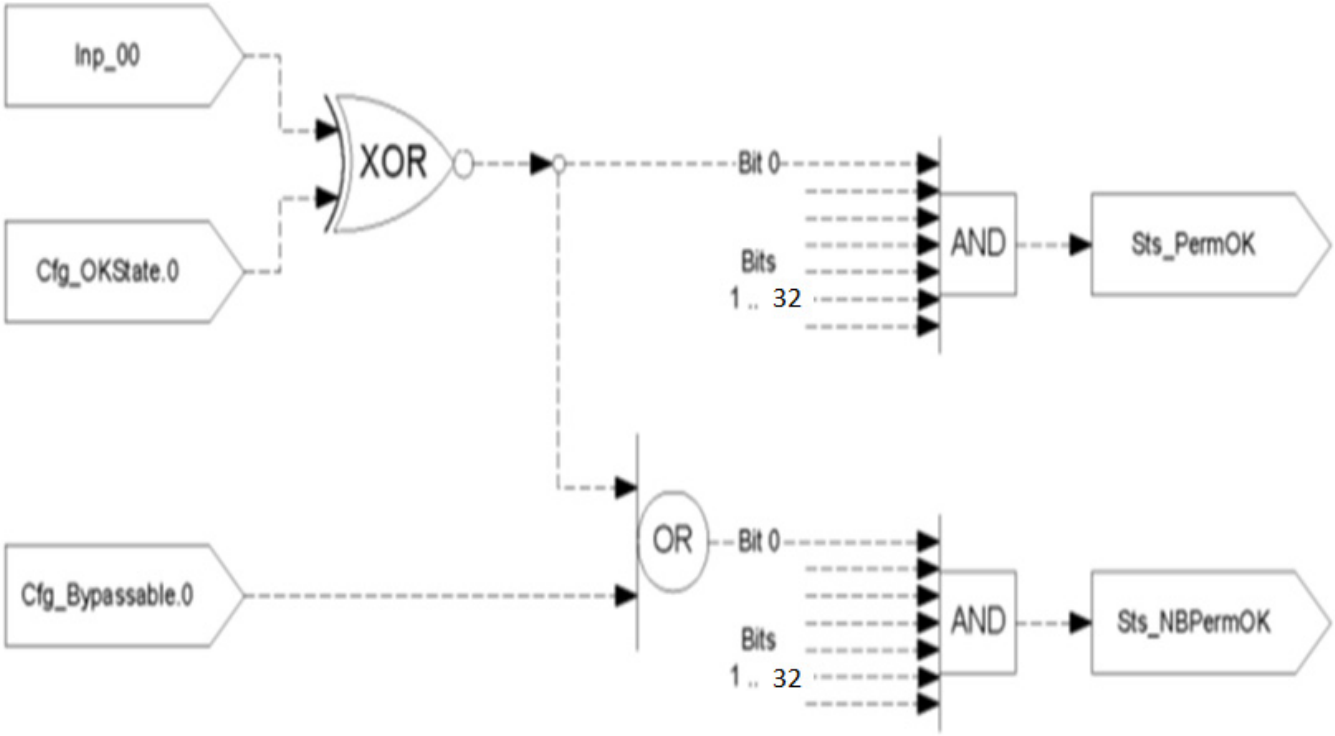
Public Output Members	Data Type	Description
Sts_Perm	DINT	Individual permissive status (1 = OK, 0 = don't energize).

Private Input Members	Data Type	Description
MSet_Bypass	DINT	Individual condition maintenance bypass toggles. Default is 2#0000_0000_0000_0000_0000_0000_0000_0000.

Private Output Members	Data Type	Description
N/A	N/A	N/A

Operation

This diagram illustrates the functionality of the PPERM instruction:



Configuration of Strings for HMI

Configure strings for HMI faceplates (FT View) and for the Logix Designer configuration dialog box. The strings are set to extended properties of tag items. Configure the strings in Logix Designer only.

- Description
- Label for graphic symbol
- Display Library for HMI Faceplate call-up
- Instruction name

- Area name
- URL link
- Input Conditional Text
- Navigation Path
- More Information

Monitor the PPERM Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Rung-condition-out is cleared to false.
Instruction first run	The instruction executes normally.
Rung-condition-in is false	Rung-condition-out is cleared to false. If this instruction is off-scan, then set the summary Permissive OK status bits to false. Only set individual permissive bypasses for conditions that are configured for bypassing. All the MSets for inputs that are NOT bypassable will be cleared.
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	EnableOut is cleared to false.
Instruction first run	The instruction executes normally.
Instruction first scan	See Instruction first run in the Function Block Diagram table.

Condition/State	Action Taken
EnableIn is false	EnableOut is cleared to false. If this instruction is off-scan, then set the summary Permissive OK status bits to false. Only set individual permissive bypasses for conditions that are configured for bypassing. All the MSets for inputs that are NOT bypassable will be cleared.
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

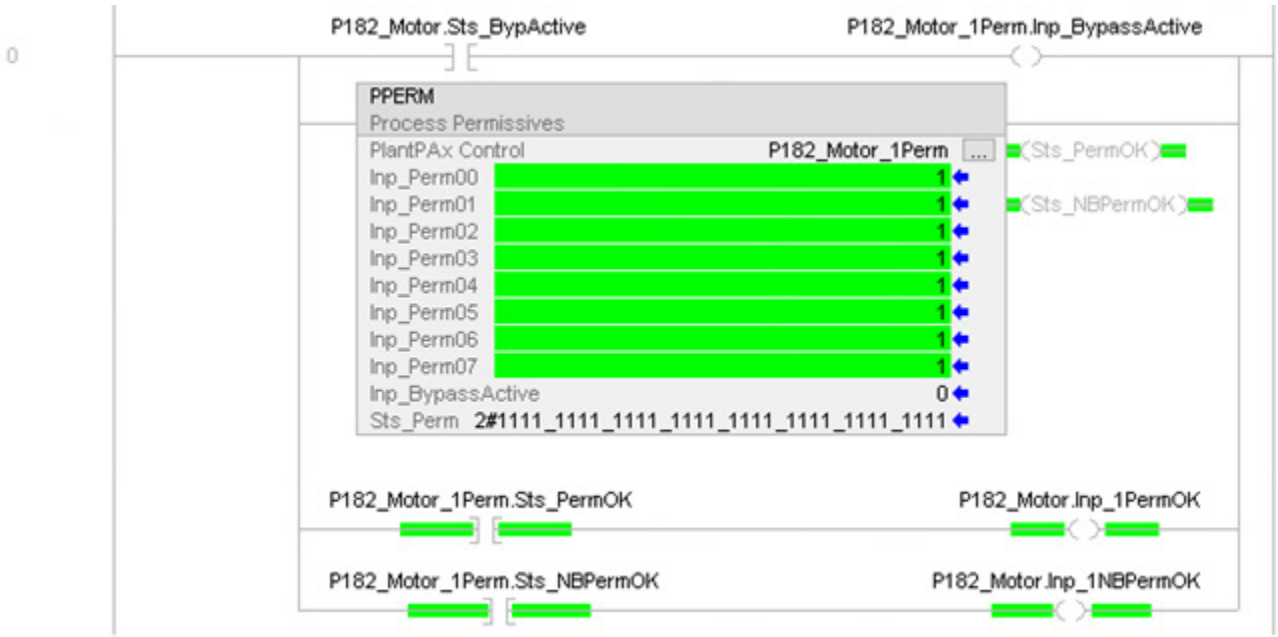
Structured Text

In Structured Text, EnableIn is always true during normal scan. The instruction executes when it is in the control path activated by the logic.

Condition/State	Action Taken
Prescan	See Prescan in the Function Block Diagram table.
Instruction first run	See Instruction first run in the Function Block Diagram table.
EnableIn is true	See EnableIn is true in the Function Block Diagram table.
Postscan	See Postscan in the Function Block Diagram table.

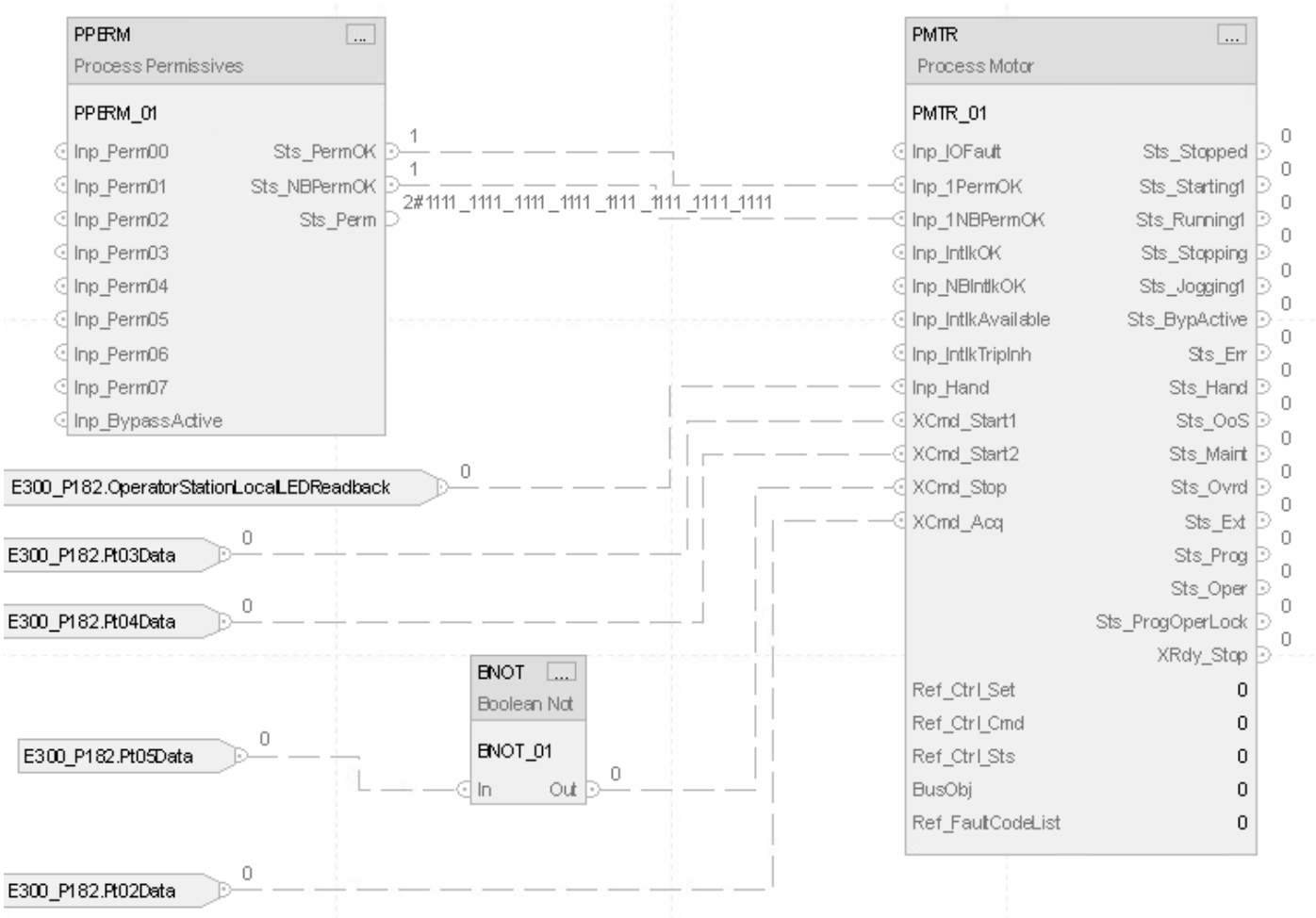
Example

Ladder Diagram



1
E300_P182.OperatorStationLocalLEDReadback
P182_Motor.Inp_Hand
E300_P182.Pt03Data
P182_Motor.XCmd_Start1
E300_P182.Pt04Data
P182_Motor.XCmd_Start2
E300_P182.Pt05Data
P182_Motor.XCmd_Stop
E300_P182.Pt02Data
P182_Motor.XCmd_Acq
PMTR
Process Motor
PlantPAx Control I82_Motor
Inp_IOFault 0
Inp_1PermOK 1
Inp_1NBPermOK 1
Inp_IntlkOK 1
Inp_NBIntlkOK 1
Inp_IntlkAvailable 0
Inp_IntlkTripInh 0
Ref_Ctrl_Set 0
Ref_Ctrl_Cmd 0
Ref_Ctrl_Sts 0
BusObj 0
Ref_FaultCodeList 0
Sts_Stopped
Sts_Starting1
Sts_Running1
Sts_Stopping
Sts_Jogging1
Sts_BypActive
Sts_Err
Sts_Hand
Sts_OoS
Sts_Maint
Sts_Ovrd
Sts_Ext
Sts_Prog
Sts_Oper
Sts_ProgOperLock
XRdy_Stop

Function Block Diagram



Structured Text

```
P182_Motor_1Perm.Inp_BypActive := P182_Motor.Sts_BypActive;
PPERM(P182_Motor_1Perm);
P182_Motor.Inp_1PermOK := P182_Motor_1Perm.Sts_PermOK;
P182_Motor.Inp_1NBPermOK := P182_Motor_1Perm.Sts_NBPermOK;
P182_Motor.Inp_Hand := E300_P182:I.OperatorStationLocalLEDReadback;
P182_Motor.XCmd_Start1 := E300_P182:I.Pt03Data;
P182_Motor.XCmd_Start2 := E300_P182:I.Pt04Data;
P182_Motor.XCmd_Stop := NOT(E300_P182:I.Pt05Data);
P182_Motor.XCmd_Acq := E300_P182:I.Pt02Data;
PMTR_ci(P182_Motor,P182_CtrlSet,P182_CtrlCmd,P182_CtrlSts);
```

See also

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Process Proportional + Integral + Derivative (PPID)

This information applies to the ControlLogix 5380P and 5580P controllers.

Use the Process Proportional + Integral + Derivative (PPID) instruction to manipulate the Control Variable (CV) in regulatory control loops in response to Process Variable (PV) readings and Setpoint (SP, the target PV) settings.

The CV is typically used as a cascade setpoint for a secondary, or inner, control loop or is sent to an Analog Output channel on an IO card.

The PPID instruction integrates functions of the existing PID, PIDE, and P_PIDE AOI into a single built-in instruction and adds additional features.

The PPID instruction:

- Calculates CV with velocity PID algorithm. Velocity algorithm is also known as incremental algorithm. Velocity algorithm computes the CV value by summing ΔPTerm, ΔITerm, ΔDTerm, ΔFF, and CV from the previous execution of the instruction. When Inp_UseCVPrev is set, CV previous is set equal to Inp_CVPrev. When Inp_UseFFPrev is set, previous FF is set equal to Inp_FFPrev. This lets you preset CV to a specified value before output CV value is computed, CV = CV previous + ΔPTerm + ΔITerm + ΔDTerm + ΔFF. Velocity form of the PID algorithm supports bumpless transfer in parameter change.
- Provides an option to suppress bumpless transfer in gain change. When this option is selected the PID calculation behaves as a position PID algorithm where control action changes with loop error and not error change.
- Uses a two degrees of freedom PID formula for calculating CV with proportional PTerm action derived from weighted SP and PV difference (b*SP-PV), integral ITerm action derived from control error (SP-PV), and derivative DTerm action derived from change in weighted SP and PV difference (c*SP-PV). The change in weight setting in run-time is bumpless.
- Uses an error-squared (option) algorithm with CV more aggressive when error rises. The use of error square is restricted to PTerm.
- Provides Direct/Reverse action (option). The action is Direct when CV rises with PV increase (Cfg_CtrlAction=1). The action is Reverse if CV decreases with PV increase (Cfg_CtrlAction=0).

- Allows deviation deadband. CV is not sensitive to loop error variation when within the band around zero error. Configured band levels allow for additional hysteresis. The deadband level for PV approaching SP (Cfg_DevDBEnter) may be set lower than deadband level for PV going away from SP (Cfg_DevDB). The active deadband status is set for PV within deadband. Provides an option to stop CV moves or just stop the integration while leaving proportional and derivative action live when in deadband (Cfg_UseIntegDevDB).
- Displays engineering units on the Logix Designer application interface. SP and PV values are displayed and entered in PV engineering units. Val_CVSet for CV target and Val_CVOut for CV output are displayed and entered in CV engineering units. The PID algorithm uses all variables scaled to percent of span internally and these values are also available as outputs Val_PVPercent, Val_SPPercent, Val_EPercent, and Val_CVPercent.
- Is configurable for Independent and Dependent gains. Interpretation of values stored in PGain, IGain and DGain tags depends on the instruction configuration.
  - Independent gains configuration: PGain = Kp … proportional gain, IGain = Ki … integral gain [1/minute], DGain = Kd … derivative gain [minutes].
  - Dependent gains configuration: PGain = K … controller gain, IGain = Ti … reset time [minutes], DGain = Td … rate time [minutes].

  Use independent gains when you want the three gains for the proportional, integral, and derivative terms to operate independently. Use dependent gains when you want an overall controller gain that affects all three terms (P, I, and D).
- Provides optional derivative smoothing (derivative limit at high frequencies). A pure derivative gives a very large amplification of measurement noise. Both dependent and independent PID algorithms can be configured for limiting the derivative term gain at high frequencies.
- Guarantees anti-windup. The PID algorithm is equipped with integral (reset) windup prevention (reset feedback). Internal windup when CV saturates is treated automatically. External windup of the inner loop indicated by Inp_WindupHi or Inp_WindupLo is also treated if the inputs are in use.
- Allows external CV tracking (option). When Cfg_UseCVTrack is set or the inner loop is not available for the PPID instruction (Inp_InnerAvailable is false), the CV tracks Inp_CVTrack value. The option is useful when the inner (secondary) loop in cascade is not able to follow CV value calculated by this outer (primary) loop PID, and Inp_WindupHi or Inp_WindupLo is either not used or not true. Tracking is allowed with configured dynamics specified in Cfg_CVTrackGain. CVTrackGain is treated as tracking gain Kt

(1/minute) for independent or tracking time constant Tt (minutes) for dependent gains.

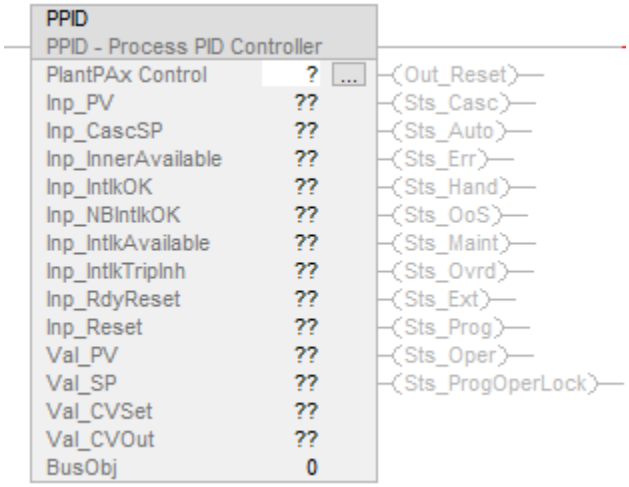
- Provides configured SP clamping and ramping: clamping (in PVEU) and ramping (in PVEU/second) of the setpoint at specified limits (separate increase and decrease rate of change limits).
- Provides configured CV clamping and ramping: clamping (in CVEU) and ramping (in CVEU/second) of CV at specified limits (separate increase and decrease rate of change limits).
- Enables CV Hand feedback.
- Supports operation from these command sources: Hand, Maintenance, Override, and full loop control (Cascade, Auto with Setpoint, Manual with CV) from External, Program, and Operator.
- Supports three automatic (Auto, Cascade, Cascade/Ratio) loop modes and one manual (Manual) loop mode.
- In Auto loop mode, setpoint (SP) in engineering units (PVEU) is read from the Program, Operator, Override or External entry. You can also enter a setpoint target in PV engineering units (PVEU) and ramp time (seconds) and use a command to trigger a built-in setpoint ramp.
- In Manual loop mode the control variable (CV) in engineering units is read from the Program, Operator, Override, or External entry.
- Monitors interlock conditions which cause output CV and SP to shed. CV shed can be configured to hold the last good CV value or to use the configured safe value. SP is shed to current PV.
- Monitors I/O communication faults.
- Supports Power-up and Initialize operations. In Power-up, the SP, CV, and Loop modes are set to configured values.
- Monitors alarm conditions for Interlock Trip, Loop Failure, High-High Deviation, High Deviation, Low Deviation, and Low-Low Deviation from the setpoint.
  - High-High Deviation status is raised when the difference between the setpoint and the process variable calculated as PV-SP exceeds configured thresholds.
  - High Deviation status is raised when the difference between the setpoint and the process variable calculated as PV-SP exceeds configured thresholds.
  - Low Deviation status is raised when the difference between the setpoint and the process variable calculated as PV-SP exceeds configured thresholds.
  - Low-Low Deviation status is raised when the difference between the setpoint and the process variable calculated as PV-SP exceeds configured thresholds.
  - Loop Failure status is raised when the PPID indicates a severe configuration error, such as invalid clamping limits, scaling limits, and deadbands. This status is also raised when PV quality is not good enough for the PID loop to work with, and when the I/O Fault

input/status is true. This status is also raised when the hand feedback input is reported bad in hand.

- Interlock Trip status is raised when an interlock not-OK condition causes the output CV to be changed to the configured Interlock CV value or held at its last value. If interlocks are not bypassed, a bypassable interlock or a non-bypassable interlock not-OK condition initiates an interlock trip. If interlocks are bypassed, only a non-bypassable interlock not-OK condition initiates an interlock trip.

Available Languages

Ladder Diagram



Function Block Diagram

Structured Text

PPID(PPID tag, BusObj);

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PlantPAx Control	P_PID	tag	Data structure required for proper operation of instruction.
BusObj	BUS_OBJ	tag	Bus component. May be null.

P_PID Structure

Public members are standard, visible tag members that are programmatically accessible. Private, or hidden, members are used in HMI faceplates and are not programmatically accessible. Private members are listed in separate tables after public members.

Public input member	Data Type	Default Value	Description
EnableIn	BOOL	True	Enable input. Ladder Diagram: Corresponds to the rung condition. Default is true.
Inp_InitializeReq	BOOL	True	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request to re-initialize. When Inp_InitializeReq = 1, power up configuration values are used. The instruction clears this operand automatically. Default is true.
Inp_OwnerCmd	DINT	0	Owner device command: 0 = None, Inp_OwnerCmd.10 = Operator Lock, Inp_OwnerCmd.11 = Operator Unlock, Inp_OwnerCmd.12 = Program Lock, Inp_OwnerCmd.13 = Program Unlock, Inp_OwnerCmd.14 = Acquire Maintenance, Inp_OwnerCmd.15 = Release Maintenance, Inp_OwnerCmd.16 = Acquire External, Inp_OwnerCmd.17 = Release External, Inp_OwnerCmd.29 = Echo. Default is 0.
Inp_PV	REAL	0.0	Process Variable (PVEU). Valid any float. Default is 0.0.
Inp_CascSP	REAL	0.0	SP in Cascade loop mode, independent PV in Ratio loop mode (PVEU). Valid any float. Default is 0.0.
Inp_FF	REAL	0.0	FeedForward term (CVEU). Valid any float between -(Cfg_CVEUMax-Cfg_CVEUMin) and (Cfg_CVEUMax-Cfg_CVEUMin). Default is 0.0.
Inp_FFPrev	REAL	0.0	Previous scan FeedForward (CVEU) when Inp_UseFFPrev = 1. Valid any float between -(Cfg_CVEUMax-Cfg_CVEUMin) and (Cfg_CVEUMax-Cfg_CVEUMin). Default is 0.0.
Inp_CVTrack	REAL	0.0	CV to track if Cfg_UseCVTrack = 1 or if Inp_InnerAvailable = 0 (CVEU). Valid any float. Default is 0.0.
Inp_CVInitialVal	PVREAL	0.0	Value to initialize the CV to per request Inp_UseCVInitialVal=1 (CVEU). Valid any float. Default is 0.0.
Inp_CVPrev	REAL	0.0	Previous scan CV, Val_CVOut (CVEU). Valid any float between Cfg_CVEUMin and Cfg_CVEUMax. Default is 0.0.

Public input member	Data Type	Default Value	Description
Inp_UseFFPrev	BOOL	False	1 = Use Inp_FFPrev as previous FF value, 0 = Use last scan Inp_FF value as previous FF value. Default is false.
Inp_UseCVInitialVal	BOOL	False	1 = Initialize CV to Inp_CVInitialVal. Default is false.
Inp_UseCVPrev	BOOL	False	1 = Use Inp_CVPrev as previous CV value, 0 = Use last scan value. Default is false.
Inp_WindupHi	BOOL	False	Windup high signal. When true, the CV cannot integrate in a positive direction. The signal is typically obtained from the Windup hi output from a inner loop. Default is false.
Inp_WindupLo	BOOL	False	Windup low signal. When true, the CV cannot integrate in a negative direction. The signal is typically obtained from the Windup low output from a inner loop. Default is false.
Inp_InnerAvailable	BOOL	True	1 = Inner loop (slave object) is available. 0 = Inner loop is not available, PPID tracks Inp_CVTrack, typically inner loop SP or actuator position. Default is true.
Inp_IntlkOK	BOOL	True	1 = Bypassable interlocks OK, CV can be set. Default is true.
Inp_NBIntlkOK	BOOL	True	1 = Non-Bypassable interlocks OK, CV can be set. Default is true.
Inp_IntlkAvailable	BOOL	False	1 = Interlock availability OK. Default is false.
Inp_IntlkTripInh	BOOL	False	1 = Inhibit interlock trip status. Default is false.
Inp_RdyReset	BOOL	False	1 = Related object, reset by this object, is ready to be reset. Ready for ORdy_Reset, enables HMI button. Default is false.
Inp_CVIOFault	BOOL	False	1 = CV I/O communications status bad, 0 = OK. Default is false.
Inp_PVBad	BOOL	False	1 = PV quality or PV I/O communications status bad, 0 = OK. Default is false.
Inp_PVUncertain	BOOL	False	1 = PV value not reliable, 0 = OK. Default is false.

Public input member	Data Type	Default Value	Description
Inp_PVSrcQ	SINT	0	Inp_PV source status and quality: 0 = Good, live, confirmed good, 1 = Good, live, assumed good, 2 = Good, no feedback, assumed good, 8 = Test, simulated, 9 = Test, loopback, 10 = Test, manually entered, 16 = Uncertain, live, off-spec, 17 = Uncertain, substituted at device, 18 = Uncertain, substituted at instruction, 19 = Uncertain, using last known good, 20 = Uncertain, using replacement value, 32 = Bad, signal failure, 33 = Bad, channel fault, 34 = Bad, module/communications fault, 35 = Bad, invalid configuration. Default is 0.
Inp_PVNotify	SINT	0	Related PV object alarm priority and acknowledgement status: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged. Default is 0.
Inp_CVNotify	SINT	0	Related CV object alarm priority and acknowledgement status: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged. Default is 0.
Inp_CascSPNotify	SINT	0	Related Cascade SP object alarm priority and acknowledgement status: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged. Default is 0.

Public input member	Data Type	Default Value	Description
Inp_HiHiDevGate	BOOL	True	The gate input used for HiHi deviation status detection: 1 = The corresponding analog input threshold monitoring is enabled, 0 = Detection is disabled and the corresponding status output is forced off. Default is true.
Inp_HiDevGate	BOOL	True	The gate input used for Hi deviation status detection: 1 = The corresponding analog input threshold monitoring is enabled, 0 = Detection is disabled and the corresponding status output is forced off. Default is true.
Inp_LoDevGate	BOOL	True	The gate input used for Lo deviation status detection: 1 = The corresponding analog input threshold monitoring is enabled, 0 = Detection is disabled and the corresponding status output is forced off. Default is true.
Inp_LoLoDevGate	BOOL	True	The gate input used for LoLo deviation status detection: 1 = The corresponding analog input threshold monitoring is enabled, 0 = Detection is disabled and the corresponding status output is forced off. Default is true.
Inp_Hand	BOOL	False	1 = Acquire Hand, 0 = Release Hand. Default is false.
Inp_HandFdbk	REAL	0.0	CV feedback used when owner is Hand (CVEU). Valid any float. Default is 0.0. Default is false.
Inp_HandFdbkBad	BOOL	False	1 = CV hand feedback quality or CV hand feedback I/O communications status bad, 0 = OK. Default is false.
Inp_Ovrd	BOOL	False	1 = Acquire Override (higher priority program logic), 0 = Release Override. Default is false.
Inp_OvrdCmd	SINT	0	Loop mode command in Override: 0 = None, 1 = Manual, 2 = Auto, 3 = Cascade, 4 = Normal, 5 = Start SP ramp, 6 = Stop SP ramp. Default is 0.
Inp_OvrdSP	REAL	0.0	Loop Auto SP in Override (PVEU). Valid any float. Default is 0.0.
Inp_OvrdSPTarget	REAL	0.0	Override setting for SP target in ramp wizard (PVEU). Valid any float. Default is 0.0.
Inp_OvrdSPRampTime	REAL	0.0	Override setting for time to reach SP target in ramp wizard (second). Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Inp_OvrdRatio	REAL	1.0	Loop Ratio in Override (unitless). Valid = 0.0 to maximum positive float. Default is 1.0.

Public input member	Data Type	Default Value	Description
Inp_OvrdCV	REAL	0.0	Loop Manual CV in Override (CVEU). Valid any float. Default is 0.0.
Inp_ExtInh	BOOL	False	1 = Inhibit External acquisition, 0 = Allow External acquisition. Default is false.
Inp_Reset	BOOL	False	1 = Reset shed latches and latched alarms whose conditions have returned to normal. Default is false.
Cfg_AllowDisable	BOOL	True	1 = Allow Maintenance to disable alarms. Default is true.
Cfg_AllowShelve	BOOL	True	1 = Allow Operator to shelve alarms.
Cfg_HasRatio	BOOL	False	1 = Cascade loop mode is Ratio, 0 = Cascade loop mode is Cascade. Default is false.
Cfg_HasCasc	BOOL	False	1 = Enable the loop to be placed into Cascade/Ratio mode. Default is false.
Cfg_HasAuto	BOOL	True	1 = Enable the loop to be placed into Auto mode. Default is true.
Cfg_HasMan	BOOL	True	1 = Enable the loop to be placed into Manual mode. Default is true.
Cfg_HasSPRamp	BOOL	False	1 = Enable the SP ramp wizard function. Default is false.
Cfg_ExecTime	REAL	0.0	Execution period for PID algorithm (second). Configuring the instruction for execution period = 0.0 (default) or period shorter than instruction scan time has no effect and the PID algorithm executes every scan. For the real execution period check Val_ExecTime. Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_PGain	REAL	0.0	Proportional gain Kp for independent gains or loop gain Kc for dependent gains (unitless). Valid = 0.0 to maximum positive float. Default is 0.0.
Cfg_IGain	REAL	0.0	Integral gain Ki (1/minute) for independent or reset time Ti (minutes/repeat) for dependent gains. Valid = 0.0 to maximum positive float. Default is 0.0.
Cfg_DGain	REAL	0.0	Derivative gain Kd (minute) for independent or rate time Td (minute) for dependent gains. Valid = 0.0 to maximum positive float. Default is 0.0.
Cfg_CVTrackGain	REAL	0.0	Tracking gain Kt (1/minute) for independent or tracking time constant Tt (minutes) for dependent gains for CV to track Inp_CVTrack if Cfg_UseCVTrack = 1. Valid = 0.0 to maximum positive float. Default is 0.0.
Cfg_UseCVTrack	BOOL	False	1 = Use Inp_CVTrack reset feedback in tracking, e.g. if PPID output is significantly faster than actuator or inner loop or in override select control.
Cfg_PSPWeight	REAL	1.0	Weight on SP in proportional term in 2DOF PID, beta gain. Valid = 0.0 to 1.5. Default is 1.0.

Public input member	Data Type	Default Value	Description
Cfg_DSPWeight	REAL	0.0	Weight on SP in derivative term in 2DOF PID, gamma gain. Valid = 0.0 to 1.5. Default is 0.0.
Cfg_PVTrack	BOOL	False	1 = SP tracks PV in Manual, 0 = No PV tracking. Default is false.
Cfg_GainBumpless	BOOL	True	1 = CV response to PGain and DGain change is bumpless, 0 = CV response to PGain and DGain change is not bumpless (like in position algorithm). Default is true.
Cfg_PositionBump	BOOL	False	1 = Position form of PD algorithm without bumpless transfer from Manual to Auto or Cascade. Enabled only when Cfg_IGain = 0. Change of proportional gain is not bumpless. 0 = Velocity form of PID algorithm with bumpless transfer from Manual to Auto or Cascade. Default is false.
Cfg_UseESquared	BOOL	False	1 = Use error squared for proportional action. Default is false.
Cfg_CtrlAction	BOOL	False	1 = Control action on E = PV-SP, direct action 0 = Control action on E = SP-PV, reverse action. Default is false.
Cfg_Dependent	BOOL	False	1 = Dependent gains equation, 0 = Independent gains equation. Default is false.
Cfg_UseDSmoothing	BOOL	False	1 = Use derivative smoothing. Default is false.
Cfg_DevDB	REAL	0.0	PV deviation deadband for PV going away from SP (PVEU). Valid = 0.0 to maximum positive float.
Cfg_DevDBEnter	REAL	0.0	PV deviation deadband for PV approaching SP (PVEU). Valid = 0.0 to Cfg_DevDB.
Cfg_UseIntegDevDB	BOOL	False	1 = Only integral term is suspended when PV deviation deadband status is active, proportional and derivative terms remain operational, 0 = All PID terms are suspended and CV does not move when PV deviation deadband status is active. Default is false.
Cfg_SkipCVManLim	BOOL	True	1 = Skip CV clamping in Manual loop mode and for CV from shed, 0 = Always apply CV clamping.
Cfg_SkipCVManRoC	BOOL	True	1 = Skip CV rate-of-change limiting in Manual loop mode and for CV from shed, 0 = Always apply CV rate-of-change limiting.
Cfg_InitializeToMan	BOOL	False	1 = Go to Manual loop mode when initialization is requested (Inp_UseCVInitialVal=1). Default is false.
Cfg_SetTrack	BOOL	True	1 = Program/Operator/External settings tracking, 0 = No settings tracking. Default is true.
Cfg_SetTrackOvrdHand	BOOL	False	1 = Program/Operator/External settings track Override/Hand inputs (CV, SP, Ratio). Default is false.
Cfg_HasIntlkObj	BOOL	False	1 = Tells HMI an interlock object (PINTLK) is connected to Inp_IntlkOK, InpNBIntlkOK, Inp_IntlkAvailable, Inp_IntlkTripInh and Inp_RdyReset. Default is false.
Cfg_HasMoreObj	BOOL	False	1 = Tells HMI an object with more info is available. Default is false.

Public input member	Data Type	Default Value	Description
Cfg_HasHistTrend	SINT	0	Has historical trend. This enables navigation to the device historical trend faceplate from the HMI. 0 = No external historical trend, 1 = Datalog historical trend, 2 = Historian historical trend. Default is 0.
Cfg_HasCascSPNav	BOOL	False	1 = Tells HMI to enable navigation to a connected cascade SP object. Default is false.
Cfg_HasPVNav	BOOL	False	1 = Tells HMI to enable navigation to a connected PV object. Default is false.
Cfg_HasCVNav	BOOL	False	1 = Tells HMI to enable navigation to a connected CV object. Default is false.
Cfg_HasOper	BOOL	True	1 = Operator (unlocked) exists, can be selected. Default is true.
Cfg_HasOperLocked	BOOL	True	1 = Operator Locked exists, can be selected. Default is true.
Cfg_HasProg	BOOL	True	1 = Program (unlocked) exists, can be selected. Default is true.
Cfg_HasProgLocked	BOOL	True	1 = Program Locked exists, can be selected. Default is true.
Cfg_HasExt	BOOL	False	1 = External exists, can be selected. Default is false.
Cfg_HasMaint	BOOL	True	1 = Maintenance exists, can be selected. Default is true.
Cfg_HasMaintOoS	BOOL	True	1 = Maintenance Out of Service exists, can be selected. Default is true.
Cfg_OvrdOverLock	BOOL	True	1 = Override supersedes Program/Operator Lock, 0 = Do not override Lock. Default is true.
Cfg_ExtOverLock	BOOL	False	1 = External supersedes Program/Operator Lock, 0 = Do not override Lock. Default is false.
Cfg_eKeepLM	SINT	0	Loop mode ownership: 0 = Follows command source, 1 = Operator, 2 = Program, 3 = External. Default is 0.
Cfg_eKeepCV	SINT	0	CV ownership: 0 = Follows command source, 1 = Operator, 2 = Program, 3 = External. Default is 0.
Cfg_eKeepSP	SINT	0	SP ownership: 0 = Follows command source, 1 = Operator, 2 = Program, 3 = External. Default is 0.

Public input member	Data Type	Default Value	Description
Cfg_eKeepRatio	SINT	0	Ratio ownership: 0 = Follows command source, 1 = Operator, 2 = Program, 3 = External. Default is 0.
Cfg_ProgPwrUp	BOOL	False	1 = Power up to Program, 0 = Power up to Operator. Default is false.
Cfg_ProgNormal	BOOL	False	Normal source: 1 = Program if no requests, 0 = Operator if no requests. Default is false.
Cfg_PCmdPriority	BOOL	False	Command priority: 1 = Program commands win, 0 = Operator commands win. Default is false.
Cfg_PCmdProgAsLevel	BOOL	False	1 = PCmd_Prog used as Level (1 = Prog, 0 = Oper). Default is false.
Cfg_PCmdLockAsLevel	BOOL	False	1 = PCmd_Lock used as Level (1 = Lock, 0 = Unlock). Default is false.
Cfg_ExtAcqAsLevel	BOOL	False	1 = XCmd_Acq used as Level (XCmd_Acq = 1 acquire external command, XCmd_Acq = 0 release external command). Default is false.
Cfg_OvrdIntlk	BOOL	False	1 = Override bypasses (ignores) bypassable interlocks. 0 = Override abides by all interlock conditions. Default is false.
Cfg_SPFailLatch	BOOL	False	1 = Latch in SP fail shed action until Reset, 0 = Return when SP is good. Default is false.
Cfg_PVFailLatch	BOOL	False	1 = Latch in PV fail shed action until Reset, 0 = Return when PV is good. Default is false.
Cfg_CVFailLatch	BOOL	False	1 = Latch in CV fail shed action until Reset, 0 = Return when CV is good. Default is false.
Cfg_LockLM	BOOL	False	Lock loop mode: 1 = Locked in loop mode configured as Normal (see Cfg_NormLM), 0 = Not locked. Default is false.
Cfg_NormLM	SINT	1	Loop mode defined as Normal: 0 = Disable Normal selection, 1 = Manual, 2 = Auto, 3 = Cascade. Default is 1.

Public input member	Data Type	Default Value	Description
Cfg_PwrUpLM	SINT	4	Loop mode defined for Powerup: 0 = No change - loop mode, CV and SP are initialized using last (powerdown) values, 1 = Manual, 2 = Auto, 3 = Cascade, 4 = Normal. Default is 4.
Cfg_PVFailTrigger	SINT	1	PV fail status response on PV quality: 0 = PV fail response on Inp_PVBad OR Inp_PVSrcQ >= 32 (PV bad), 1 = PV fail response on Inp_PVBad OR Inp_PVSrcQ >= 17 AND Inp_PVSrcQ <> 18, 2 = PV fail response on Inp_PVBad OR Inp_PVUncertain OR Inp_PVSrcQ >= 16 AND Inp_PVSrcQ <> 18. Default is 1.
Cfg_IntlkTripSPAction	SINT	0	Interlock Trip SP action: 0 = None, 1 = Hold last good, 2 = Use Cfg_SPIntlk, 3 = Set SP to current PV. Default is 0.
Cfg_SPFailSPAction	SINT	1	SP Fail SP action: 1 = Hold last good, 2 = Use Cfg_SPIntlk, 3 = Set SP to current PV. Default is 1.
Cfg_PVFailSPAction	SINT	0	PV Fail SP action: 0 = None, 1 = Hold last good, 2 = Use Cfg_SPIntlk. Default is 0.
Cfg_CVFailSPAction	SINT	0	CV Fail SP action: 0 = None, 1 = Hold last good, 2 = Use Cfg_SPIntlk, 3 = Set SP to current PV. Default is 0.
Cfg_IntlkTripCVAction	SINT	2	Interlock trip CV action: 0 = None, 1 = Hold last good, 2 = Use Cfg_CVIntlk. Default is 2.
Cfg_SPFailCVAction	SINT	1	SP Fail CV action: 0 = None, 1 = Hold last good, 2 = Use Cfg_CVIntlk. Default is 1.
Cfg_PVFailCVAction	SINT	1	PV Fail CV action: 1 = Hold last CV, 2 = Use Cfg_CVIntlk. Default is 1.

Public input member	Data Type	Default Value	Description
Cfg_CVFailCVAction	SINT	1	CV Fail CV action: 1 = Hold last good, 2 = Use Cfg_CVIntlk. Default is 1.
Cfg_IntlkTripLMAction	SINT	0	Interlock trip loop mode action: 0 = None, 1 = Manual only, 2 = Auto or Manual only. Default is 0.
Cfg_SPFailLMAction	SINT	0	SP Fail loop mode action: 0 = None, 1 = Manual only, 2 = Auto or Manual only. Default is 0.
Cfg_PVFailLMAction	SINT	0	PV Fail loop mode action: 0 = none, 1 = Manual only, 2 =Auto or Manual only. Default is 0.
Cfg_CVFailLMAction	SINT	0	CV Fail loop mode action: 0 = None, 1 = Manual only, 2 = Auto or Manual only. Default is 0.
Cfg_PVDecPlcs	SINT	2	Number of decimal places for display of PV / SP (up to six). Default is 2.
Cfg_CVDecPlcs	SINT	2	Number of decimal places for display of CV (up to six). Default is 2.
Cfg_RatioDecPlcs	SINT	2	Number of decimal places for display of Ratio (up to six). Default is 2.
Cfg_RatioLoLim	REAL	0.0	Minimum allowed Ratio value (unitless). Valid any float less than or equal to Cfg_RatioHiLim. Default is 0.0.
Cfg_RatioHiLim	REAL	1.0	Maximum allowed Ratio value (unitless). Valid any float greater than or equal to Cfg_RatioLoLim. Default is 1.0.
Cfg_SPLoLim	REAL	0.0	Minimum allowed SP value (PVEU). Valid any float less than or equal to Cfg_SPHiLim and greater than or equal to Cfg_PVEUMin. Default is 0.0.
Cfg_SPHiLim	REAL	100.0	Maximum allowed SP value (PVEU). Valid any float greater than or equal to Cfg_SPLoLim and less than or equal to Cfg_PVEUMax. Default is 100.0.
Cfg_SPRoCIncrLim	REAL	0.0	Maximum allowed SP rate of change increasing value (PVEU/second). The SP rate of change is unlimited when increasing if Cfg_SPRoCIncrLim = 0.0. Valid = 0.0 to maximum positive float. Default is 0.0.
Cfg_SPRoCDecrLim	REAL	0.0	Maximum allowed SP rate of change decreasing value (PVEU/second). The SP rate of change is unlimited when decreasing if Cfg_SPRoCDecrLim = 0.0. Valid = 0.0 to maximum positive float. Default is 0.0.

Public input member	Data Type	Default Value	Description
Cfg_SkipSPRoCLim	BOOL	False	1 = Skip setpoint RoC limiting in interlock, maintenance and override. Default is false.
Cfg_SPRampMaxDev	REAL	100.0	If absolute value of deviation exceeds this value, pause SP ramp, 0.0 = Never pause (PVEU). Valid any nonnegative float. Default is 100.0.
Cfg_PVEUMin	REAL	0.0	PV minimum value for scaling from engineering units to %, PV at 0% (PVEU). Valid any float less than Cfg_PVEUMax. Default is 0.0.
Cfg_PVEUMax	REAL	100.0	PV maximum value for scaling from engineering units to %, PV at 100% (PVEU). Valid any float greater than Cfg_PVEUMin. Default is 100.0.
Cfg_CVEUMin	REAL	0.0	CV minimum value for scaling from % to engineering units (CVEU). Valid any float less than Cfg_CVEUMax. Default is 0.0.
Cfg_CVEUMax	REAL	100.0	CV maximum value for scaling from % to engineering units (CVEU). Valid any float greater than Cfg_CVEUMin.
Cfg_CVLoLim	REAL	0.0	Minimum allowed CV value (CVEU). Valid any float less than or equal to Cfg_CVHiLim and greater than or equal to Cfg_CVEUMin. Default is 0.0.
Cfg_CVHiLim	REAL	100.0	Maximum allowed CV value (CVEU). Valid any float greater than or equal to Cfg_CVLoLim and less than or equal to Cfg_CVEUMax. Default is 100.0.
Cfg_CVRoCIncrLim	REAL	0.0	Maximum allowed CV rate of change increasing value (CVEU/second). The CV rate of change is unlimited when increasing if Cfg_CVRoCIncrLim = 0.0 Valid = 0.0 to maximum positive float. Default is 0.0.
Cfg_CVRoCDecrLim	REAL	0.0	Maximum allowed CV rate of change decreasing value (CVEU/second). The CV rate of change is unlimited when decreasing if Cfg_CVRoCDecrLim = 0.0. Valid = 0.0 to maximum positive float. Default is 0.0.
Cfg_MaxInactiveCV	REAL	0.0	When Val_CVOut is greater than this value, set Sts_Active for HMI (CVEU). Valid any float. Default is 0.0.
Cfg_SPIntlk	REAL	0.0	SP value to use with interlock / bad value SP action (PVEU). Valid any float between Cfg_PVEUMin and Cfg_PVEUMax.
Cfg_CVIntlk	REAL	0.0	CV value to use with interlock / bad value CV action (CVEU). Valid any float between Cfg_CVEUMin and Cfg_CVEUMax.
Cfg_SPPwrUp	REAL	0.0	Loop SP on Powerup (PVEU) used when Cfg_PwrUpLM is not 0. The value is clamped to the SP range (Cfg_SPLoLim, Cfg_SPHiLim). Valid any float between Cfg_PVEUMin and Cfg_PVEUMax.
Cfg_CVPwrUp	REAL	0.0	Loop CV on Powerup (CVEU) used when Cfg_PwrUpLM is not 0. Value may be clamped to the configured limits (Cfg_CVLoLim, CfgHiLim) in cascade or auto, and in manual if so configured. Valid any float between Cfg_CVEUMin and Cfg_CVEUMax.
Cfg_CVPwrUpSel	SINT	0	Selection of Powerup (first run) CV in Auto or Cascade. 0 = Ignore Inp_InnerAvailable and always use Cfg_CVPwrUp or last (Powerdown) CV (if Cfg_PwrpUpLM = 0), 1 = Process Inp_InnerAvailable. Default is false.

Public input member	Data Type	Default Value	Description
Cfg_HiHiDevLim	REAL	1.5e+38	High-High PV deviation status threshold (PVEU). Valid = 0.0 to maximum positive float. Default is 1.5e+38.
Cfg_HiHiDevDB	REAL	0.0	High-High PV deviation status deadband (PVEU). Valid = 0.0 to Cfg_HiHiDevLim. Default is 0.0.
Cfg_HiHiDevGateDly	REAL	0.0	High-High PV deviation status gate delay (second). Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_HiDevLim	REAL	1.5e+38	High PV deviation status threshold (PVEU). Valid = 0.0 to maximum positive float. Default is 1.5e+38.
Cfg_HiDevDB	REAL	0.0	High PV deviation status deadband (PVEU). Valid = 0.0 to Cfg_HiDevLim. Default is 0.0.
Cfg_HiDevGateDly	REAL	0.0	High PV deviation status gate delay (second). Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_LoDevLim	REAL	-1.5e+38	Low PV deviation status threshold (PVEU). Valid = -maximum positive float to 0.0. Default is -1.5e+38.
Cfg_LoDevDB	REAL	0.0	Low PV deviation status deadband (PVEU). Valid = 0.0 to abs(Cfg_LoDevLim). Default is 0.0.
Cfg_LoDevGateDly	REAL	0.0	Low PV deviation status gate delay (second). Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_LoLoDevLim	REAL	-1.5e+38	Low-Low PV deviation status threshold (PVEU). Valid = -maximum positive float to 0.0. Default is -1.5e+38.
Cfg_LoLoDevDB	REAL	0.0	Low-Low PV deviation status deadband (PVEU). Valid = 0.0 to abs(Cfg_LoLoDevLim). Default is 0.0.
Cfg_LoLoDevGateDly	REAL	0.0	Low-Low PV deviation status gate delay (second). Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
Cfg_CnfrmReqd	SINT	0	Operator command confirmation required. Represents the type of command confirmation required. 0 = None, 1 = Command confirmation required, 2 = Performer e-signature required, 3 = Performer and approver e-signature required. Default is 0.
PSet_Ratio	REAL	1.0	Program setting for Ratio, loop mode Cascade/Ratio and Ratio enabled (unitless). Valid = 0.0 to maximum positive float. Default is 1.0.
PSet_SP	REAL	0.0	Program setting for SP, loop mode Auto (PVEU). Valid any float. Default is 0.0.

Public input member	Data Type	Default Value	Description
PSet_SPTarget	REAL	0.0	Program setting for SP target in ramp wizard (PVEU). Valid any float. Default is 0.0.
PSet_SPRampTime	REAL	0.0	Program setting for time to reach SP target in ramp wizard (second). Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
PSet_CV	REAL	0.0	Program setting for CV when loop mode is Manual (CVEU). Valid any float. Default is 0.0.
PSet_Owner	DINT	0	Program owner request ID (non-zero) or release (zero). Default is 0.
XSet_Ratio	REAL	1.0	External setting for Ratio, loop mode Cascade/Ratio and Ratio enabled (unitless). Valid = 0.0 to maximum positive float. Default is 1.0.
XSet_SP	REAL	0.0	External setting for SP, loop mode Auto (PVEU). Valid any float. Default is 0.0.
XSet_SPTarget	REAL	0.0	External setting for SP target in ramp wizard (PVEU). Valid any float. Default is 0.0.
XSet_SPRampTime	REAL	0.0	External setting for time to reach SP target in ramp wizard (second). Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
XSet_CV	REAL	0.0	External setting for CV, loop mode Manual (CVEU). Valid any float. Default is 0.0.
PCmd_Casc	BOOL	False	Program command to select Cascade/Ratio loop mode. The instruction clears this operand automatically. Default is false.
PCmd_Auto	BOOL	False	Program command to select Auto loop mode. The instruction clears this operand automatically. Default is false.
PCmd_Man	BOOL	False	Program command to select Manual loop mode. The instruction clears this operand automatically. Default is false.
PCmd_NormLM	BOOL	False	Program command to select loop mode defined as Normal, see Cfg_NormLM. The instruction clears this operand automatically. Default is false.
PCmd_SPRampStart	BOOL	False	Program command to initiate SP ramping. The instruction clears this operand automatically. Default is false.
PCmd_SPRampStop	BOOL	False	Program command to stop SP ramping. The instruction clears this operand automatically. Default is false.
PCmd_Oper	BOOL	False	Program command to select Operator (Program to Operator). The instruction clears this operand automatically. Default is false.
PCmd_Prog	BOOL	False	Program command to select Program (Operator to Program). The instruction clears this operand automatically. Default is false.

Public input member	Data Type	Default Value	Description
PCmd_Lock	BOOL	False	Program command to lock Program (disallow Operator). The instruction clears this operand automatically if Cfg_PCmdLockAsLevel = 0. Default is false.
PCmd_Unlock	BOOL	False	Program command to unlock Program (allow Operator to acquire). The instruction clears this operand automatically. Default is false.
PCmd_Normal	BOOL	False	Program command to select Normal command source (Operator or Program). The instruction clears this operand automatically. Default is false.
PCmd_Reset	BOOL	False	Program command to reset all alarms and latched shed conditions requiring reset. The instruction clears this operand automatically. Default is false.
XCmd_Casc	BOOL	False	External command to select Cascade/Ratio loop mode. The instruction clears this operand automatically. Default is false.
XCmd_Auto	BOOL	False	External command to select Auto loop mode. The instruction clears this operand automatically. Default is false.
XCmd_Man	BOOL	False	External command to select Manual loop mode. The instruction clears this operand automatically. Default is false.
XCmd_NormLM	BOOL	False	External command to select loop mode defined as Normal, see Cfg_NormLM. The instruction clears this operand automatically. Default is false.
XCmd_SPRampStart	BOOL	False	External command to initiate SP ramping. The instruction clears this operand automatically. Default is false.
XCmd_SPRampStop	BOOL	False	External command to stop SP ramping. The instruction clears this operand automatically. Default is false.
XCmd_Acq	BOOL	False	External command to acquire ownership (Operator/Program/Override/Maintenance to External). The instruction clears this operand automatically if Cfg_ExtAcqAsLevel = 0. Default is false.
XCmd_Rel	BOOL	False	External command to release ownership if Cfg_ExtAcqAsLevel = 0 (External to Operator/Program/Override/Maintenance). The instruction clears this operand automatically. Default is false.
XCmd_Reset	BOOL	False	External command to reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.
XCmd_ResetAckAll	BOOL	False	External command to acknowledge and reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.

Public output member	Data Type	Description
EnableOut	BOOL	Enable output. This output state always reflects EnableIn input state.
Val_PV	REAL	Value of loop PV (PVEU).
Val_Ratio	REAL	Value of loop Ratio (unitless).
Val_SPSet	REAL	Value of selected SP after clamping and before ramping (PVEU).

Public output member	Data Type	Description
Val_SP	REAL	Value of SP being used after clamping and ramping (PVEU).
Val_SPTarget	REAL	Accepted setting for SP ramp target, endpoint for ramp wizard (PVEU).
Val_SPRampTime	REAL	Accepted setting for SP ramp time, time to reach target for ramp wizard (second).
Val_SPRampRoC	REAL	Calculated value of SP rate of change for ramping, from ramp wizard (PVEU/second).
Val_SPRoCIncr	REAL	Current value of SP rate of change limit increasing (PVEU/second). 0.0 = rate of change not limited.
Val_SPRoCDecr	REAL	Current value of SP rate of change limit decreasing (PVEU/second). 0.0 = rate of change not limited.
Val_E	REAL	Loop error, SP-PV for reverse action Cfg_CtrlAction = 0, PV-SP for direct action Cfg_CtrlAction = 1 (PVEU).
Val_CVSet	REAL	Loop CV after clamping and before ramping (CVEU).
Val_CVOut	REAL	Loop CV after clamping and ramping (CVEU).
Val_PVPercent	REAL	Loop PV (percent of span).
Val_SPPercent	REAL	Loop SP (percent of span).
Val_EPercent	REAL	Loop error, SP-PV for reverse action Cfg_CtrlAction = 0, PV-SP for direct action Cfg_CtrlAction = 1 (percent of span).
Val_CVOutPercent	REAL	Loop CV after ramping and clamping (percent of span).
Val_ExecTime	REAL	Actual PID algorithm execution period (second).
Out_Reset	BOOL	1 = Reset command has been received and accepted.
Out_OwnerSts	DINT	Status of command source, owner command handshake and ready status: 0 = None, .10 = Operator Lock, .11 = Operator Unlock, .12 = Program Lock, .13 = Program Unlock, .14 = Acquire Maintenance, .15 = Release Maintenance, .16 = Acquire External, .17 = Release External, .18 = Has Maintenance, .19 = External Override Lock, .20 = Has External, .21 = Has Operator, .22 = Has Operator Locked, .23 = Has Program, .24 = Has Program Locked, .29 = Echo, .30 = Not Ready.
Sts_Initialized	BOOL	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.

Public output member	Data Type	Description
SrcQ_IO	SINT	Source and quality of primary I/O PV quality (enumerated): 0 = Good, live, confirmed good, 1 = Good, live, assumed good, 2 = Good, no feedback, assumed good, 8 = Test, virtualized, 9 = Test, loopback, 10 = Test, manually entered, 16 = Uncertain, live, off-spec, 17 = Uncertain, substituted at device or bus, 18 = Uncertain, substituted at instruction, 19 = Uncertain, using last known good, 20 = Uncertain, using replacement value, 32 = Bad, signal failure, 33 = Bad, channel fault, 34 = Bad, module or communication fault, 35 = Bad, invalid configuration.
SrcQ	SINT	Source and quality of primary PV and CV value or status (enumerated): 0 = Good, live, confirmed good 1 = Good, live, assumed good 2 = Good, no feedback, assumed good 8 = Test, virtualized 9 = Test, loopback 10 = Test, manually entered 16 = Uncertain, live, off-spec 17 = Uncertain, substituted at device or bus 18 = Uncertain, substituted at instruction 19 = Uncertain, using last known good 20 = Uncertain, using replacement value 32 = Bad, signal failure 33 = Bad, channel fault 34 = Bad, module or communication fault 35 = Bad, invalid configuration
Sts_eSts	SINT	Loop mode: 0 = Unknown, 1 = Manual, 2 = Auto, 3 = Cascade (no Ratio), 4 = Ratio.
Sts_eFault	SINT	Loop fault: 0 = none, 1 = PV uncertain, 2 = Low PV deviation, 3 = High PV deviation, 4 = Low-Low PV deviation, 5 = High-High PV deviation, 6 = PV substituted, 7 = Interlock trip, 8 = SP fail, 9 = PV fail, 10 = CV fail, 11 = Configuration error.

Public output member	Data Type	Description
Sts_eState	SINT	Internal logic state for animating state diagram on faceplate. 0 = Manual, 1 = PV deviation in deadband, 2 = PV deviation not in deadband, 3 = SP ramping, 4 = CV ramping, 5 = Ratio clamped, 6 = SP clamped, 7 = CV clamped, 8 = Windup Lo, 9 = Windup Hi, 10 = SP held, 11 = SP set to IntlkSP, 12 = CV held, 13 = CV set to IntlkCV, 14 = Hand, 15 = Initializing.
Sts_eNotify	SINT	Highest severity alarm status. All alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyAll	SINT	Highest severity alarm status. All alarm status enumerated values including related objects for CV, PV, SP, CascSP: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyHiHiDev	SINT	High-High PV deviation alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.

Public output member	Data Type	Description
Sts_eNotifyHiDev	SINT	High PV deviation alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyLoDev	SINT	Low PV deviation alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyLoLoDev	SINT	Low-Low PV deviation alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyFail	SINT	Fail alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyIntlkTrip	SINT	Interlock trip alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.

Public output member	Data Type	Description
Sts_UnackAlmCount	DINT	Count of unacknowledged alarms.
Sts_Casc	BOOL	1 = Loop is in Cascade/Ratio mode.
Sts_Auto	BOOL	1 = Loop is in Auto mode.
Sts_Man	BOOL	1 = Loop is in Manual mode.
Sts_NormLM	BOOL	1 = Loop is in Normal loop mode configured in Cfg_NormLM.
Sts_Initializing	BOOL	1 = CV is initializing because of request Inp_UseCVInitialVal or when Inp_InnerAvailable=0.
Sts_WindupHi	BOOL	1 = This loop winding up High, usually connects to Inp_WindupHi of outer loop.
Sts_WindupLo	BOOL	1 = This loop winding up Low, usually connects to Inp_WindupLo of outer loop.
Sts_RatioClamped	BOOL	1 = Selected Ratio (PSet/OSet_Ratio or Inp_OvrdRatio) has been clamped.
Sts_IntlkSP	BOOL	1 = SP value is being set by shed to Interlock SP.
Sts_SPHeld	BOOL	1 = SP value is being set by shed to hold last good SP.
Sts_SPShedPV	BOOL	1 = SP value is being set by shed to current PV.
Sts_SPShed	BOOL	1 = SP value from Shed, 0 = SP from Program, Operator, Override (Auto), Casc SP Input (Cascade) or PV Track (Manual).
Sts_SPTrackPV	BOOL	1 = SP value is being set by PV tracking in Manual loop mode.
Sts_SPHiClamped	BOOL	1 = Selected SP is being clamped at high limit.
Sts_SPLoClamped	BOOL	1 = Selected SP is being clamped at low limit.
Sts_SPClamped	BOOL	1 = Selected SP is being clamped, for faceplate animation.
Sts_SPRampingUp	BOOL	1 = SP is ramping up toward Val_SPSet.
Sts_SPRampingDown	BOOL	1 = SP is ramping down toward Val_SPSet.
Sts_SPRamping	BOOL	1 = SP is ramping toward Val_SPSet, 0 = Ramp complete.
Sts_SPRampWizardInProgress	BOOL	1 = SP is ramping toward SP target set by the owner in ramp wizard, 0 = Ramp complete.
Sts_SkipSPRoCLim	BOOL	1 = SP rate limiting is being skipped, for faceplate animation.
Sts_DevDBAct	BOOL	1 = PV deviation deadband active.
Sts_PVUncertain	BOOL	1 = PV input value quality is uncertain.
Sts_PVBad	BOOL	1 = PV input value, communications, quality or EU limit is bad.
Sts_SPBad	BOOL	1 = Cascade SP input value quality is bad.
Sts_FFBad	BOOL	1 = Feedforward term Inp_FF value is invalid.
Sts_FFPrevBad	BOOL	1 = Feedforward term Inp_FFPrev value is invalid.
Sts_CVInfNaN	BOOL	1 = CV value equal to +/- infinity or NaN detected.
Sts_CVBad	BOOL	1 = CV value quality is bad or invalid or CV comms fault.
Sts_CVPrevBad	BOOL	1 = Inp_CVPrev value is invalid.
Sts_HandFdbkBad	BOOL	1 = Hand FB (Tieback) value quality is bad or invalid or communication fault.
Sts_IntlkCV	BOOL	1 = CV value is being set by shed to Interlock CV.
Sts_CVHeld	BOOL	1 = CV value is being set by shed to hold last good CV.
Sts_CVShed	BOOL	1 = CV from shed, 0 = CV from Program, Operator, Override (Manual), or PID (Auto, Cascade).
Sts_CVHiClamped	BOOL	1 = CV is being clamped at high limit.
Sts_CVLoClamped	BOOL	1 = CV is being clamped at low limit.
Sts_CVClamped	BOOL	1 = Selected CV is being clamped, used in faceplate animation.
Sts_CVRampingUp	BOOL	1 = CV is ramping up toward Val_CVSet.
Sts_CVRampingDown	BOOL	1 = CV is ramping down toward Val_CVSet.
Sts_CVRamping	BOOL	1 = CV is ramping toward Val_CVSet, 0 = Ramp complete.
Sts_Active	BOOL	1 = CV is greater than Cfg_MaxInactiveCV, show graphic symbol as active.
Sts_Available	BOOL	1 = PID loop can be acquired by Program and is available for control.
Sts_CascAvailable	BOOL	1 = PID inner loop is available for cascade control with an outer loop. 0 = PID inner loop is not available, initialize outer loop to Val_SP.
Sts_ExtAvailable	BOOL	1 = PID is available for external control. 0 = PID is not available, initialize outer loop to Val_SP.
Sts_IntlkAvailable	BOOL	1 = Interlock availability OK. Device can be acquired by program and is available for control when interlocks are OK.
Sts_Bypass	BOOL	1 = Bypassable interlocks are bypassed.

Public output member	Data Type	Description
Sts_BypActive	BOOL	1 = Interlock bypassing is active (bypassed or Maintenance).
Sts_NotRdy	BOOL	1 = PPID is not ready, see detail bits for reason.
Sts_NrdyCfgErr	BOOL	1 = PPID is not ready: Configuration error.
Sts_NrdyInit	BOOL	1 = PPID is not ready while CV is initialized (Inp_UseCVInitVal = 1).
Sts_NrdyIntlk	BOOL	1 = PPID is not ready: Interlock Not OK (Shed requires reset).
Sts_NrdyFail	BOOL	1 = PPID is not ready: Fail status is on (Shed requires reset).
Sts_NrdyCVFail	BOOL	1 = PPID is not ready: CV Fail (Shed requires reset).
Sts_NrdyPVFail	BOOL	1 = PPID is not ready: PV Fail (Shed requires reset).
Sts_NrdySPFail	BOOL	1 = PPID is not ready: SP Fail (Shed requires reset).
Sts_NrdyOoS	BOOL	1 = PPID is not ready: Out of Service.
Sts_MaintByp	BOOL	1 = Device has a Maintenance bypass function active.
Sts_NrdyInner	BOOL	1 = Inner loop object is not available (Inp_InnerAvailable = 0) for this PPID.
Sts_Err	BOOL	1 = Error in PPID configuration: see detail Err bits for reason.
Sts_ErrRatioLim	BOOL	1 = Error in PPID configuration: Ratio clamping limits invalid. Cfg_RatioLoLim > Cfg_RatioHiLim.
Sts_ErrSPLim	BOOL	1 = Error in PPID configuration: SP clamping limits invalid. Cfg_SPLoLim < Cfg_PVEUMin or Cfg_SPHiLim > Cfg_PVEUMax or Cfg_SPLoLim > Cfg_SPHiLim.
Sts_ErrCVLim	BOOL	1 = Error in PPID configuration: CV clamping limits invalid. Cfg_CVLoLim < Cfg_CVEUMin or Cfg_CVHiLim > Cfg_CVEUMax or Cfg_CVLoLim > Cfg_CVHiLim.
Sts_ErrPVEU	BOOL	1 = Error in PPID configuration: PV scaling limits invalid. Cfg_PVEUMin >= Cfg_PVEUMax or Cfg_PVEUMin is +-Inf.
Sts_ErrCVEU	BOOL	1 = Error in PPID configuration: CV scaling limits invalid. Cfg_CVEUMin >= Cfg_CVEUMax or Cfg_CVEUMin is +-Inf.
Sts_ErrDevDB	BOOL	1 = Error in PPID configuration: PV deviation deadband invalid. Cfg_DevDBEnter > Cfg_DevDB, or Cfg_DevDB < 0, or Cfg_DevDBEnter < 0.
Sts_ErrExecTime	BOOL	1 = Error in PPID configuration: Execution time invalid, Cfg_ExecTime < 0.0 or Cfg_ExecTime > 2147483.0 seconds.
Sts_ErrPGain	BOOL	1 = Error in PPID configuration: PGain invalid, Cfg_PGain < 0.
Sts_ErrIGain	BOOL	1 = Error in PPID configuration: IGain invalid, Cfg_IGain < 0.
Sts_ErrDGain	BOOL	1 = Error in PPID configuration: DGain invalid, Cfg_DGain < 0.
Sts_ErrCVTrackGain	BOOL	1 = Error in PPID configuration: CV TrackGain invalid, Cfg_CVTrackGain < 0.
Sts_ErrPSPWeight	BOOL	1 = Error in PPID configuration: SP weight in proportional term invalid, Cfg_PSPWeight < 0 or Cfg_PSPWeight > 1.5.
Sts_ErrDSPWeight	BOOL	1 = Error in PPID configuration: SP weight in derivative term invalid, Cfg_DSPWeight < 0 or Cfg_DSPWeight > 1.5.
Sts_ErrSPRoCIncrLim	BOOL	1 = Error in PPID configuration: Maximum allowed SP rate of change increasing value invalid, Cfg_SPRoCIncrLim < 0.
Sts_ErrSPRoCDecrLim	BOOL	1 = Error in PPID configuration: Maximum allowed SP rate of change decreasing value invalid, Cfg_SPRoCDecrLim < 0.
Sts_ErrCVRoCIncrLim	BOOL	1 = Error in PPID configuration: Maximum allowed CV rate of change increasing value invalid, Cfg_CVRoCIncrLim < 0.
Sts_ErrCVRoCDecrLim	BOOL	1 = Error in PPID configuration: Maximum allowed CV rate of change decreasing value invalid, Cfg_CVRoCDecrLim < 0.
Sts_ErrSPIntlk	BOOL	1 = Error in PPID configuration: SP value to use with Interlock or bad value SP action invalid, value out of scale, Cfg_SPIntlk > Cfg_PVEUMax or < Cfg_PVEUMin.
Sts_ErrCVIntlk	BOOL	1 = Error in PPID configuration: CV value to use with Interlock or bad value CV action invalid, value out of scale, Cfg_CVIntlk > Cfg_CVEUMax or < Cfg_CVEUMin.

Public output member	Data Type	Description
Sts_ErrSPPwrUp	BOOL	1 = Error in PPID configuration: SP value to use in Powerup invalid, value out of scale, Cfg_SPPwrUp > Cfg_PVEUMax or < Cfg_PVEUMin.
Sts_ErrCVPwrUp	BOOL	1 = Error in PPID configuration: CV value to use in Powerup invalid, value out of scale, Cfg_CVIntlk > Cfg_CVEUMax or < Cfg_CVEUMin.
Sts_ErrAlm	BOOL	1 = Error in PPID configuration: At least one Logix tag-based alarm has invalid settings.
Sts_ErrHiHiDevLim	BOOL	1 = Error in PPID configuration: High-High PV deviation threshold invalid, Cfg_HiHiDevLim < 0.
Sts_ErrHiHiDevGateDly	BOOL	1 = Error in PPID configuration: Invalid High-High PV deviation gate delay timer preset (use 0.0 to 2147483.0).
Sts_ErrHiHiDevDB	BOOL	1 = Error in PPID configuration: High-High PV deviation deadband invalid, Cfg_HiHiDevDB < 0 or Cfg_HiHiDevDB > Cfg_HiHiDevLim.
Sts_ErrHiDevLim	BOOL	1 = Error in PPID configuration: High PV deviation threshold invalid, Cfg_HiDevLim < 0.0.
Sts_ErrHiDevGateDly	BOOL	1 = Error in PPID configuration: Invalid High PV deviation gate delay timer preset (use 0.0 to 2147483.0).
Sts_ErrHiDevDB	BOOL	1 = Error in PPID configuration: High PV deviation deadband invalid, Cfg_HiDevDB < 0 or Cfg_HiDevDB > Cfg_HiDevLim.
Sts_ErrLoDevLim	BOOL	1 = Error in PPID configuration: Low PV deviation threshold invalid, Cfg_LoDevLim > 0.
Sts_ErrLoDevGateDly	BOOL	1 = Error in PPID configuration: Invalid Low PV deviation gate delay timer preset (use 0.0 to 2147483.0).
Sts_ErrLoDevDB	BOOL	1 = Error in PPID configuration: Low PV deviation deadband invalid, Cfg_LoDevDB < 0, or > -Cfg_LoDevLim.
Sts_ErrLoLoDevLim	BOOL	1 = Error in PPID configuration: Low-Low PV deviation threshold invalid, Cfg_LoLoDevLim > 0.0.
Sts_ErrLoLoDevGateDly	BOOL	1 = Error in PPID configuration: Invalid Low-Low PV deviation gate delay timer preset (use 0.0 to 2147483.0).
Sts_ErrLoLoDevDB	BOOL	1 = Error in PPID configuration: Low-Low PV deviation deadband invalid, Cfg_LoLoDevDB < 0, or > -Cfg_LoLoDevLim.
Sts_eSrc	INT	The current command source enumerated values: 0 = Logic not in use, 4 = Hand, 8 = Maintenance, 16 = Override, 32 = Program, 33 = Program locked, 34 = Program by default (Normal), 64 = Operator, 65 = Operator locked, 66 = Operator by default (Normal), 128 = Maintenance Out of Service, 129 = Programmed Out of Service (rung false), 256 = External.
Sts_bSrc	INT	Active selection bitmap (for HMI totem pole with command source request selection): Sts_bSrc.0: Hand, Sts_bSrc.1: Programmed Out of Service (rung false), Sts_bSrc.2: Maintenance Out of Service, Sts_bSrc.3: Maintenance, Sts_bSrc.4: Override, Sts_bSrc.5: External, Sts_bSrc.6: Program locked, Sts_bSrc.7: Program, Sts_bSrc.8: Operator locked, Sts_bSrc.9: Operator.
Sts_Hand	BOOL	1 = Hand is selected, supersedes OoS, Maint, Ovrd, Ext, Prog, Oper.
Sts_OoS	BOOL	1 = Out of Service is selected, supersedes Maint, Ovrd, Ext, Prog, Oper.
Sts_Maint	BOOL	1 = Maintenance is selected, supersedes Ovrd, Ext, Prog, Oper.

Public output member	Data Type	Description
Sts_Ovrd	BOOL	1 = Override is selected, supersedes Ext, Prog, Oper.
Sts_Ext	BOOL	1 = External is selected, supersedes Prog, Oper.
Sts_Prog	BOOL	1 = Program is selected.
Sts_ProgLocked	BOOL	1 = Program is selected and locked.
Sts_Oper	BOOL	1 = Operator is selected.
Sts_OperLocked	BOOL	1 = Operator is selected and locked.
Sts_ProgOperSel	BOOL	Program/Operator selection (latch) state: 1 = Program, 0 = Operator.
Sts_ProgOperLock	BOOL	Program/Operator lock (latch) state: 1 = Locked, 0 = Unlocked.
Sts_Normal	BOOL	1 = Owner selection equals what is configured as Normal (Prog or Oper).
Sts_ExtReqInh	BOOL	1 = External request inhibited.
Sts_ProgReqInh	BOOL	1 = Program request inhibited, cannot get to Program from current owner.
Sts_MAcqRcvd	BOOL	1 = Maintenance acquire command received this scan.
Sts_Alm	BOOL	1 = An alarm is active.
Sts_AlmInh	BOOL	1 = An alarm is shelved or disabled.
Sts_HiHiDevCmp	BOOL	1 = PV deviation is above High-High limit, (Val_PV-Val_SP)>Cfg_HiHiDevLim.
Sts_HiHiDevGate	BOOL	1 = PV deviation High-High gate is open.
Sts_HiHiDev	BOOL	1 = PV deviation is above High-High limit, (Val_PV-Val_SP)>Cfg_HiHiDevLim, for gate open. There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PPIDTag.@Alarms.Alm_HiHiDev.AlarmElement
Sts_HiDevCmp	BOOL	1 = PV deviation is above High limit, (Val_PV-Val_SP)>Cfg_HiDevLim.
Sts_HiDevGate	BOOL	1 = PV deviation High gate is open.
Sts_HiDev	BOOL	1 = PV deviation is above High limit, (Val_PV-Val_SP)>Cfg_HiDevLim, for gate open. There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PPIDTag.@Alarms.Alm_HiDev.AlarmElement
Sts_LoDevCmp	BOOL	1 = PV deviation is below Low limit, (Val_PV-Val_SP)<Cfg_LoDevLim.
Sts_LoDevGate	BOOL	1 = PV deviation Low gate is open.
Sts_LoDev	BOOL	1 = PV deviation (loop error) is below Low limit, (Val_PV-Val_SP)<Cfg_LoDevLim, for gate open. There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PPIDTag.@Alarms.Alm_LoDev.AlarmElement
Sts_LoLoDevCmp	BOOL	1 = PV deviation is below Low-Low limit, (Val_PV-Val_SP)<Cfg_LoLoDevLim.
Sts_LoLoDevGate	BOOL	1 = PV deviation Low-Low gate is open.
Sts_LoLoDev	BOOL	1 = PV deviation (loop error) is below Low-Low limit, (Val_PV-Val_SP)<Cfg_LoLoDevLim, for gate open. There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PPIDTag.@Alarms.Alm_LoLoDev.AlarmElement
Sts_Fail	BOOL	1 = Loop failure: PV bad, SP bad, CV bad or Hand feedback bad statuses are on or are latched on without reset. There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PPIDTag.@Alarms.Alm_Fail.AlarmElement
Sts_IntlkTrip	BOOL	1 = Interlock Not OK caused loop output to hold or change. There is a predefined default discrete tag-based alarm for the status. Set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PPIDTag.@Alarms.Alm_IntlkTrip.AlarmElement
Sts_RdyReset	BOOL	1 = A latched alarm or shed condition is ready to be reset.
Sts_RdyAck	BOOL	1 = An alarm is ready to be acknowledged.
XRdy_Acq	BOOL	1 = Ready for XCmd_Acq, enable HMI button.
XRdy_Rel	BOOL	1 = Ready for XCmd_Rel, enable HMI button.
XRdy_Casc	BOOL	1 = Ready for XCmd_Casc, enable HMI button.

Public output member	Data Type	Description
XRdy_Auto	BOOL	1 = Ready for XCmd_Auto, enable HMI button.
XRdy_Man	BOOL	1 = Ready for XCmd_Man, enable HMI button.
XRdy_NormLM	BOOL	1 = Ready for XCmd_NormLM, enable HMI button.
XRdy_SPRampStart	BOOL	1 = Ready for XCmd_SPRampStart, enable HMI button.
XRdy_SPRampStop	BOOL	1 = Ready for XCmd_SPRampStop, enable HMI button.
XRdy_Reset	BOOL	1 = Ready for XCmd_Reset, enable HMI button.
XRdy_ResetAckAll	BOOL	1 = Ready for XCmd_ResetAckAll, enable HMI button.
Val_Owner	DINT	Current object owner ID, 0 = Not owned.

Private input member	Data Type	Default Value	Description
HMI_BusObjIndex	DINT	0	HMI bus object index. Default is 0.
OCmd_SPRampStart	BOOL	0	Operator command to initiate SP ramping. The instruction clears this operand automatically. Default is 0.
OCmd_SPRampStop	BOOL	0	Operator command to stop SP ramping. The instruction clears this operand automatically. Default is 0.
OSet_Ratio	REAL	1.0	Operator setting for Ratio, loop mode Cascade/Ratio and Ratio enabled (unitless). Valid = 0.0 to maximum positive float. Default is 1.0.
OSet_SP	REAL	0.0	Operator setting for Setpoint, loop mode Auto (PVEU). Valid any float. Default is 0.0.
OSet_SPTarget	REAL	0.0	Operator setting for Setpoint ramp target, endpoint for ramp wizard (PVEU). Valid any float. Default is 0.0.
OSet_SPRampTime	REAL	0.0	Operator setting for Setpoint ramp time, time to reach target for ramp wizard (second). Valid = 0.0 to 2147483.0 seconds. Default is 0.0.
OSet_CV	REAL	0.0	Operator setting for CV, loop mode Manual (CVEU). Valid any float. Default is 0.0.
MCmd_Bypass	BOOL	0	Maintenance command to bypass all bypassable interlocks. The instruction clears this operand automatically. Default is 0.
MCmd_Check	BOOL	0	Maintenance command to check, not bypass, all interlocks. The instruction clears this operand automatically. Default is 0.
MCmd_OoS	BOOL	0	Maintenance command to select Out of Service. The instruction clears this operand automatically. Default is 0.
MCmd_IS	BOOL	0	Maintenance command to select In Service. The instruction clears this operand automatically. Default is 0.
MCmd_Acq	BOOL	0	Maintenance command to acquire ownership (Operator/Program/External/Override to Maintenance). The instruction clears this operand automatically. Default is 0.

Private input member	Data Type	Default Value	Description
MCmd_Rel	BOOL	0	Maintenance command to release ownership (Maintenance to Operator/Program/External/Override). The instruction clears this operand automatically. Default is 0.
OCmd_Oper	BOOL	0	Operator command to select Operator (Program to Operator). The instruction clears this operand automatically. Default is 0.
OCmd_Prog	BOOL	0	Operator command to select Program (Operator to Program). The instruction clears this operand automatically. Default is 0.
OCmd_Lock	BOOL	0	Operator command to lock Operator (disallow Program). The instruction clears this operand automatically. Default is 0.
OCmd_Unlock	BOOL	0	Operator command to unlock or release (allow Program to acquire) ownership. The instruction clears this operand automatically. Default is 0.
OCmd_Normal	BOOL	0	Operator command to select Normal (Operator or Program). The instruction clears this operand automatically. Default is 0.
OCmd_Auto	BOOL	0	Operator command to select Automatic loop mode. The instruction clears this operand automatically. Default is 0.
OCmd_Casc	BOOL	0	Operator command to select Cascade/Ratio loop mode. The instruction clears this operand automatically. Default is 0.
OCmd_Man	BOOL	0	Operator command to select Manual loop mode. The instruction clears this operand automatically. Default is 0.
OCmd_NormLM	BOOL	0	Operator command to select loop mode configured as Normal (see Cfg_NormLM). The instruction clears this operand automatically. Default is 0.
OCmd_Reset	BOOL	0	Operator command to reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is 0.
OCmd_ResetAckAll	BOOL	0	Operator command to acknowledge and reset all alarms and latched shed conditions. The use of OCmd_ResetAckAll is restricted to HMI. The instruction clears this operand automatically. Default is 0.

Private output member	Data Type	Description
MRdy_Bypass	BOOL	1 = Ready for MCmd_Bypass, enable HMI button.
MRdy_Check	BOOL	1 = Ready for MCmd_Check, enable HMI button.
MRdy_OoS	BOOL	1 = Ready for MCmd_OoS, enable HMI button.
MRdy_IS	BOOL	1 = Ready for MCmd_IS, enable HMI button.
MRdy_Acq	BOOL	1 = Ready for MCmd_Acq, enable HMI button.
MRdy_Rel	BOOL	1 = Ready for MCmd_Rel, enable HMI button.
ORdy_Oper	BOOL	1 = Ready for OCmd_Oper, enable HMI button.
ORdy_Prog	BOOL	1 = Ready for OCmd_Prog, enable HMI button.
ORdy_Lock	BOOL	1 = Ready for OCmd_Lock, enable HMI button.
ORdy_Unlock	BOOL	1 = Ready for OCmd_Unlock, enable HMI button.

Private output member	Data Type	Description
ORdy_Normal	BOOL	1 = Ready for OCmd_Normal, enable HMI button.
ORdy_Auto	BOOL	1 = Ready for OCmd_Auto, enables HMI button.
ORdy_Casc	BOOL	1 = Ready for OCmd_Casc, enables HMI button.
ORdy_CV	BOOL	1 = Ready for OSet_CV, enables data entry field.
ORdy_Man	BOOL	1 = Ready for OCmd_Man, enables HMI button.
ORdy_NormLM	BOOL	1 = Ready for OCmd_NormLM, enables HMI button.
ORdy_Ratio	BOOL	1 = Ready for OSet_Ratio, enables data entry field.
ORdy_SP	BOOL	1 = Ready for OSet_SP, enables data entry field.
ORdy_SPRampStart	BOOL	1 = Ready for OCmd_SPRampStart, enables HMI button.
ORdy_SPRampStop	BOOL	1 = Ready for OCmd_SPRampStop, enables HMI button.
ORdy_SPTarget	BOOL	1 = Ready for OSet_SPTarget, enables data entry field.
ORdy_SPRampTime	BOOL	1 = Ready for OSet_SPRampTime, enables data entry field.
ORdy_Reset	BOOL	1 = Ready for OCmd_Reset, enables HMI button.
ORdy_ResetAckAll	BOOL	1 = At least one alarm or latched shed condition requires Reset or Acknowledgement.

Public InOut member	Data Type	Description
BusObj	BUS_OBJ	Bus component

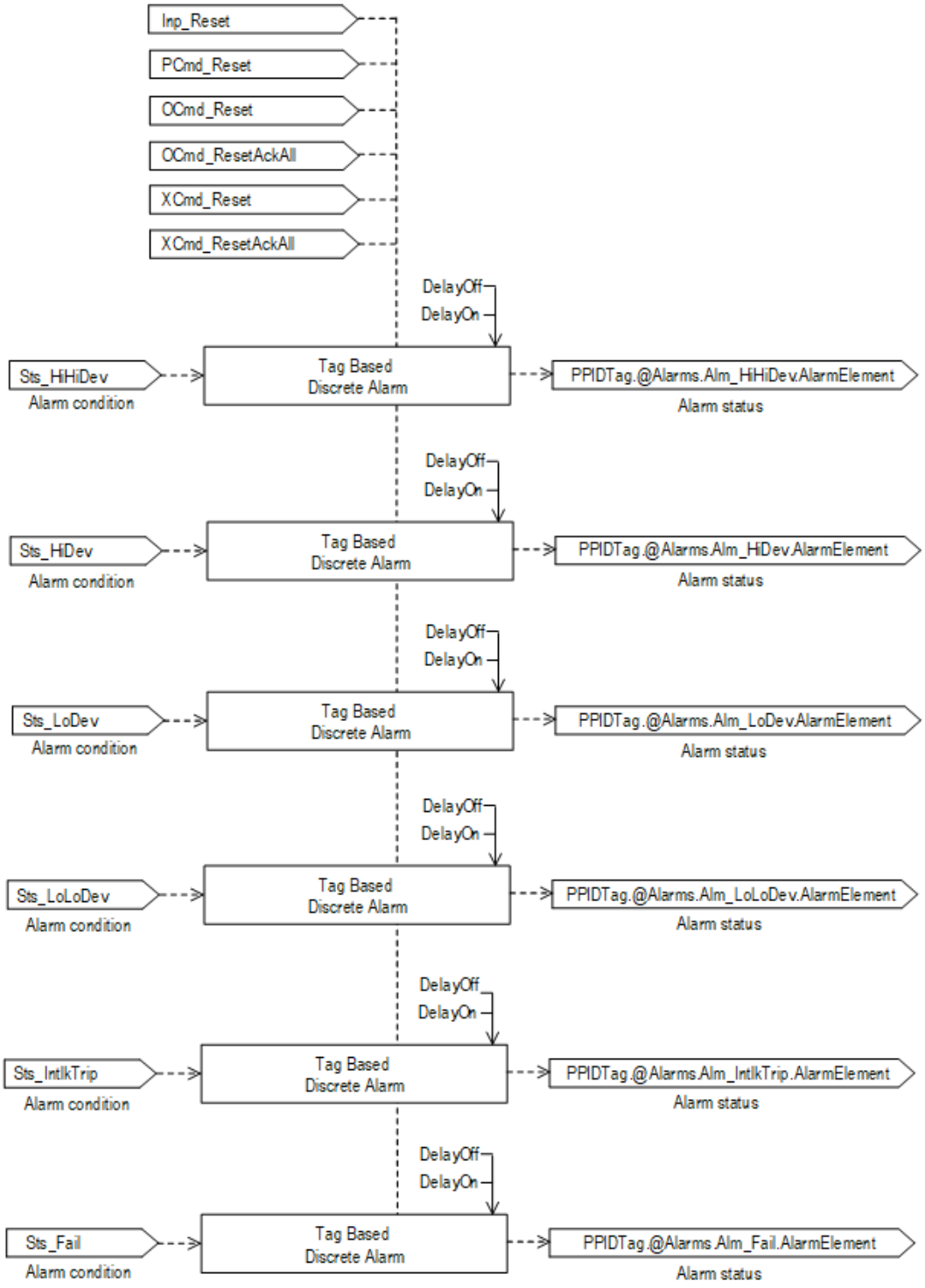
Alarms

Discrete tag-based alarms are defined for these members.

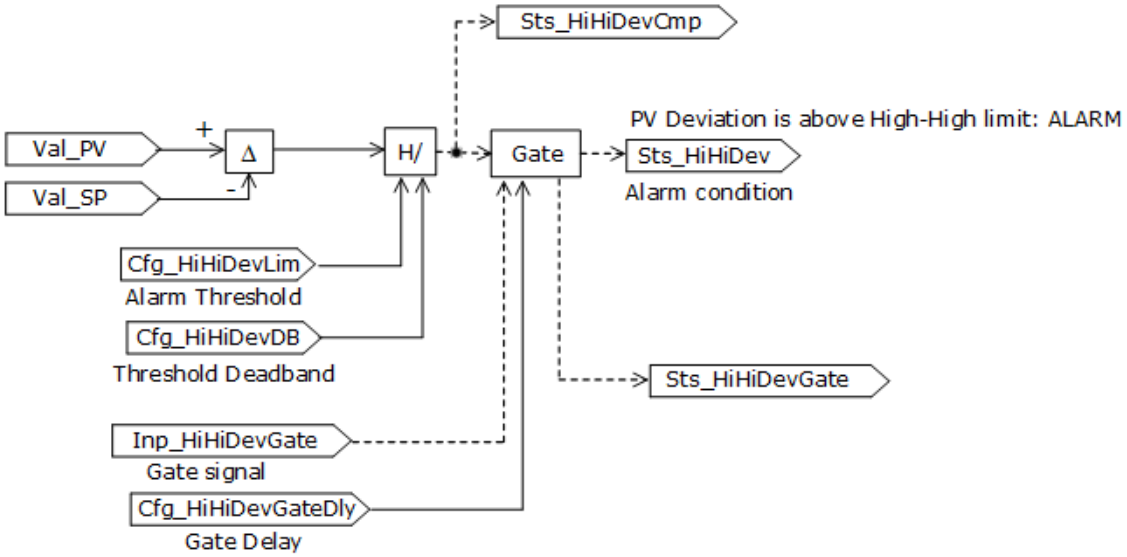
Member	Alarm Name	Description
Sts_Fail	Alm_Fail	Instruction Failure. Raised when the PPID instruction reports an Instruction Fault. The Fault is reported under any of these conditions: • Setpoint (SP) bad quality, Sts_SPBad = 1, • Process variable (PV) bad quality, Sts_PVBad = 1, • Control variable (CV) bad quality, Sts_CVBad = 1, • Hand feedback (HandFdbk) bad quality, Sts_HandFdbkBad = 1.
Sts_IntlkTrip	Alm_IntlkTrip	Interlock Trip alarm. Raised when non-bypassable interlocks are not OK or bypassable interlocks are not OK when not bypassed. NAND(Inp_NBIntlkOK,(OR(Inp_IntlkOK,Stst_BypActive))).
Sts_HiHiDev	Alm_HiHiDev	High-High Deviation. Raised when the amount by which the PV exceeds the setpoint or reference is above the High-High Deviation threshold. The threshold, deadband, gating, and timing are set in configuration.
Sts_HiDev	Alm_HiDev	High Deviation. Raised when the amount by which the PV exceeds the setpoint or reference is above the High Deviation threshold. The threshold, deadband, gating, and timing are set in configuration.
Sts_LoDev	Alm_LoDev	Low Deviation. Raised when the amount by which the PV exceeds the setpoint or reference is below the Low Deviation threshold. (Since the threshold is a negative number, this reading is the amount the PV falls below the setpoint or reference.) The threshold, deadband, gating, and timing are set in configuration.
Sts_LoLoDev	Alm_LoLoDev	Low-Low Deviation. Raised when the amount by which the PV exceeds the setpoint or reference is below the Low-Low Deviation threshold. (Since the threshold is a negative number, this reading is the amount the PV falls below the setpoint or reference.) The threshold, deadband, gating, and timing are set in configuration.

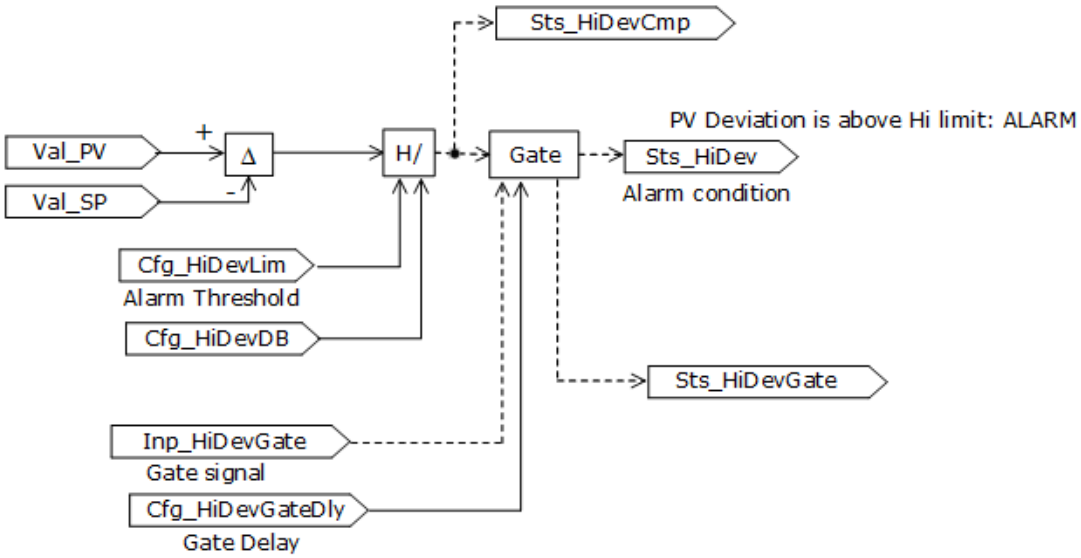
Mark the alarm as used or unused and set standard configuration members of the discrete tag-based alarm. Access alarm elements using this format: PPIDTag.@Alarms.AlarmName.AlarmElement.

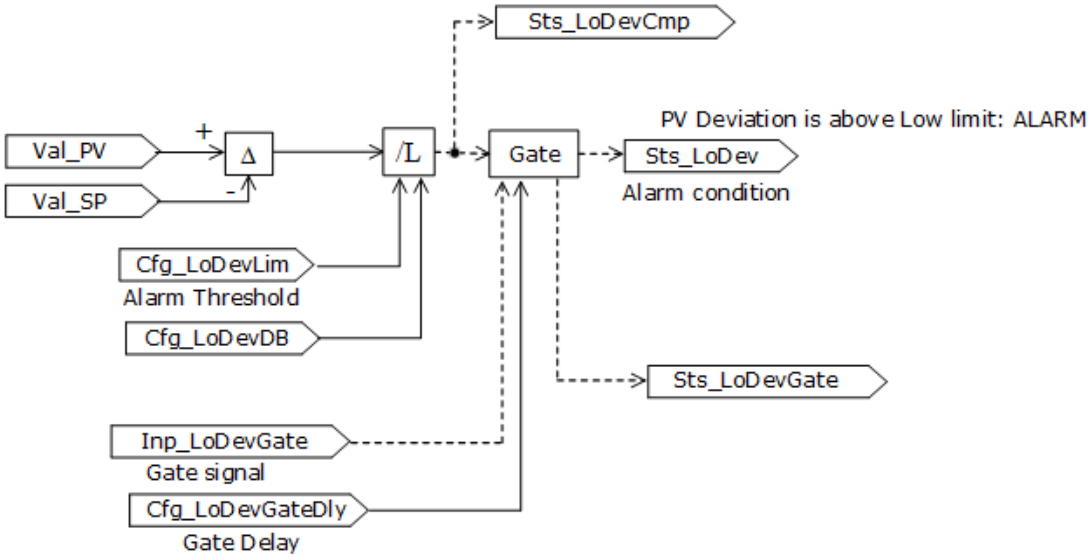
There are Program, Operator, and External commands that enable the Reset and Reset & Acknowledge of all alarms of the instruction (Alarm Set) at the same time. This diagram shows how alarm condition interact with alarm command.

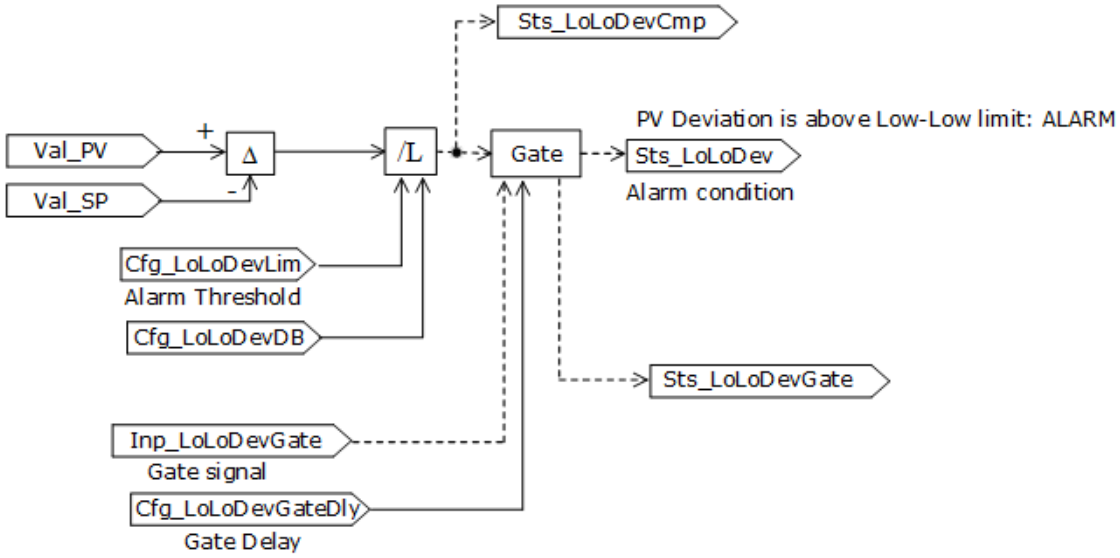


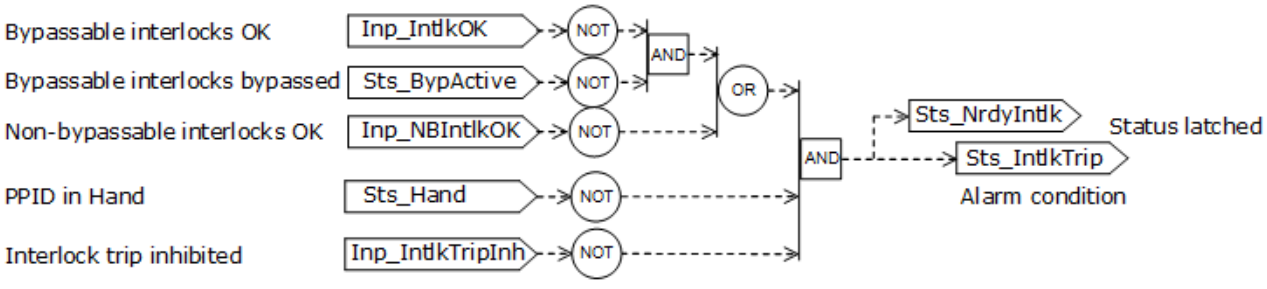
These diagrams explain how alarm condition high-high deviation, high deviation, low deviation, low-low deviation, interlock trip and fail are derived.

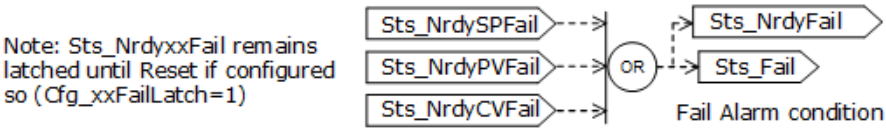





Sts_LoDevCmp
PV Deviation is above Low limit: ALARM
Val_PV
+
Δ
/L
Gate
Sts_LoDev
Alarm condition
Val_SP
-
Cfg_LoDevLim
Alarm Threshold
Cfg_LoDevDB
Sts_LoDevGate
Inp_LoDevGate
Gate signal
Cfg_LoDevGateDly
Gate Delay


Sts_LoLoDevCmp
PV Deviation is above Low-Low limit: ALARM
Val_PV
+
Δ
/L
Gate
Sts_LoLoDev
Alarm condition
Val_SP
-
Cfg_LoLoDevLim
Alarm Threshold
Cfg_LoLoDevDB
Sts_LoLoDevGate
Inp_LoLoDevGate
Gate signal
Cfg_LoLoDevGateDly
Gate Delay


Bypassable interlocks OK
Inp_IntlkOK
NOT
Bypassable interlocks bypassed
Sts_BypActive
NOT
AND
OR
Non-bypassable interlocks OK
Inp_NBIntlkOK
NOT
Sts_NrdyIntlk
Status latched
AND
Sts_IntlkTrip
Alarm condition
PPID in Hand
Sts_Hand
NOT
Interlock trip inhibited
Inp_IntlkTripInh
NOT


Note: Sts_NrdyxxFail remains latched until Reset if configured so (Cfg_xxFailLatch=1)
Sts_NrdySPFail
Sts_NrdyPVFail
Sts_NrdyCVFail
OR
Sts_NrdyFail
Sts_Fail
Fail Alarm condition

Operation

In principle the PID core algorithm calculates CV with a formula configured for dependent gains,

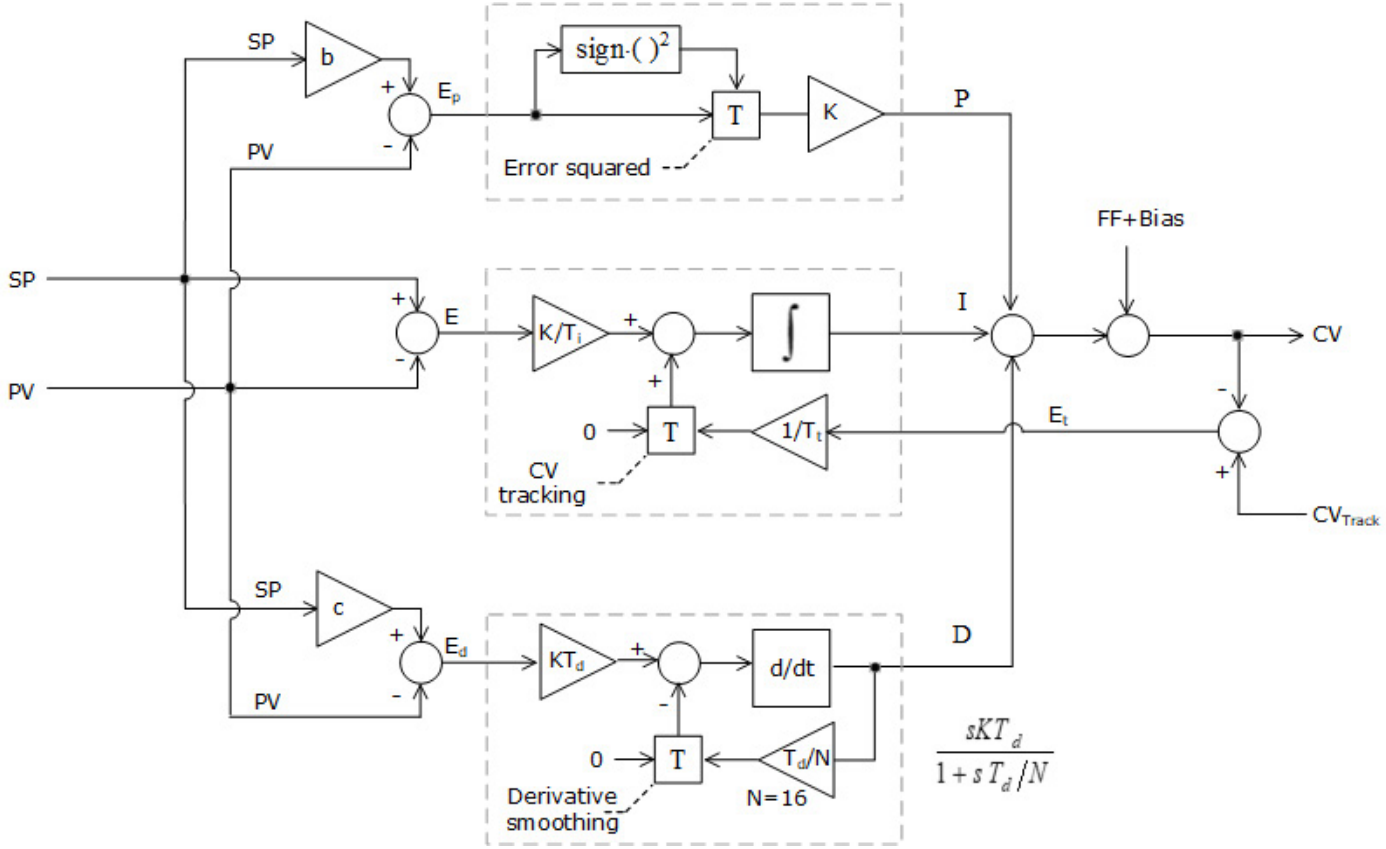
$$CV(t) = K\left[e(t) + \frac{1}{Ti}\int_{\square}^{t} e(\tau)d\tau + Td\frac{de(t)}{dt}\right] + FF(t) + Bias$$

where K is controller gain [-], Ti is reset time [minutes], Td is rate time [minutes]. Alternatively the instruction is configured for independent gains,

$$CV(t) = K_P e(t) + K_I\int_{\square}^{t} e(\tau)d\tau + K_D\frac{de(t)}{dt} + FF(t) + Bias$$

where K_P is proportional gain [-], K_I is integral gain [1/minute], and K_D is derivative gain [minutes]. Use independent gains when you want the three gains for the proportional, integral and derivative terms to operate independently.

This principle diagram illustrates additional configuration options with derivative smoothing and weighted setpoint used in error calculations entering proportional and derivative term, and error squared in proportional term and CV tracking:

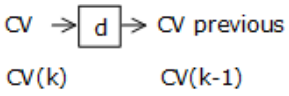


The following section uses exact tag names from P_PID structure. Variables without a prefix, such as SP, PV, CVPrev, CVTrack, FF, FFPrev, and CtrlAction, are internal variables used in calculation. Variables with prefix Inp_, Val_, of

data type REAL are in engineering units. Variables without prefix of data type REAL are in % of span.

Ts [second] is a current period of PID algorithm execution, i.e. time elapsing from the previous scan of the PPID instruction.

d is a one-scan delay operator. For example:

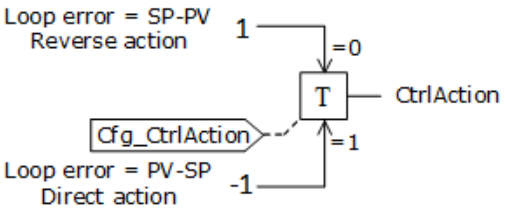


Variables entering the PID formula must be in percent of span so the PID gains do not depend on engineering units used for PV and CV. Scaling of these parameters is calculated as Perc=min(100,max(0,(EU-EUMin)/(EUMax-EUMin)*100)).

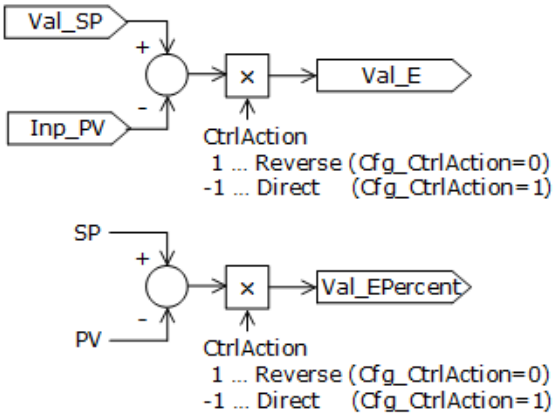
- Val_SP,
- Inp_PV,
- Inp_CVPrev,
- Inp_CVTrack,
- Cfg_CVHiLim,
- Cfg_LoLim,
- Cfg_CVRoCIncrLim,
- Cfg_CVRoCDecrLim

Scaling of Inp_FF and Inp_FFPrev is calculated as Perc=min(100,max(-100,(EU-EUMin)/(EUMax-EUMin)*100)).

Reverse/Direct control action affects loop error sign that will be used in further calculations:

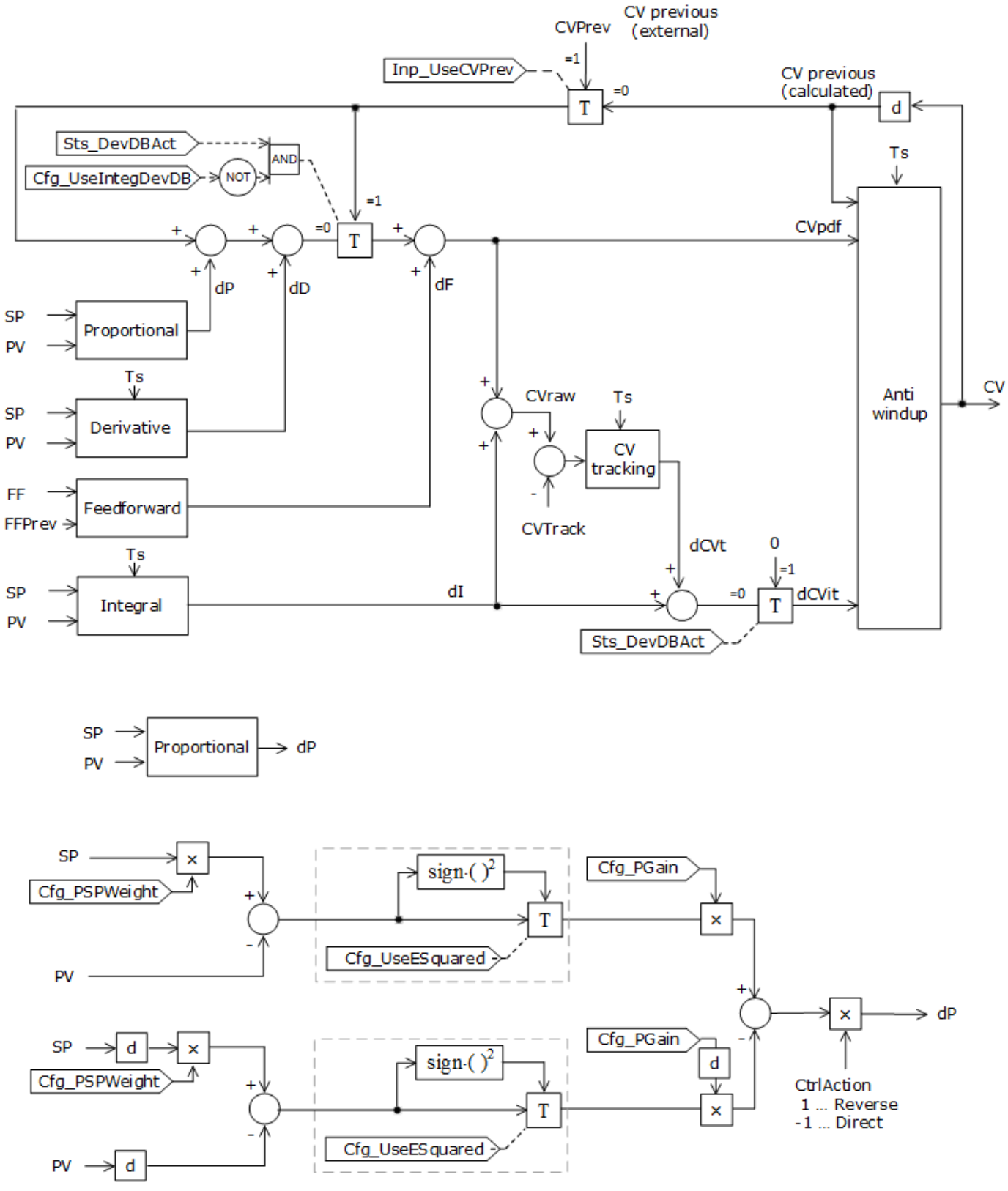


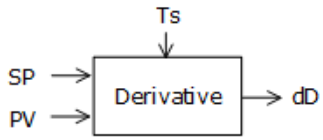
Loop error calculations:



Velocity PID algorithm

This diagram of the velocity PID algorithm describes the CV calculation. Additional detailed diagrams in this section describe all PPID configuration options.





Calculation of derivative term:

Modified error calculation:

$E_c(k)$ = Cfg_DSPCoef(k) * $SP(k) - PV(k)$

$E_c(k)$ = Cfg_DSPCoef(k) * $SP(k-1) - PV(k-1)$

$\Delta E_c(k) = E_c(k) - E_c(k-1)$

= (Cfg_DSPCoef(k) * $SP(k) - PV(k)$) – (Cfg_DSPCoef(k) * $SP(k-1) - PV(k-1)$)

$\Delta E_c(k-1) = E_c(k-1) - E_c(k-2)$

= (Cfg_DSPCoef(k) * $SP(k-1) - PV(k-1)$) – (Cfg_DSPCoef(k) * $SP(k-2) - PV(k-2)$)

Contribution of derivative term if Cfg_UseDSmooting=0, Cfg_Dependent=1 and Cfg_GainBumpless=1.

$dD(k)$ = Cfg_PGain(k) * Cfg_DGain(k) * ($\Delta E_c(k) - \Delta E_c(k-1)$)

Contribution of derivative term if Cfg_UseDSmooting=0, Cfg_Dependent=1 and Cfg_GainBumpless=0.

$dD(k)$ = Cfg_PGain(k) * Cfg_DGain(k) * $\Delta E_c(k)$ – Cfg_PGain(k–1) * Cfg_DGain(k–1) * $\Delta E_c(k-1)$

Contribution of derivative term if Cfg_UseDSmooting=0, Cfg_Dependent=0 and Cfg_GainBumpless=1.

$dD(k)$ = Cfg_DGain(k) * ($\Delta Ec(k) - \Delta E_c(k-1)$)

–

Contribution of derivative term if Cfg_UseDSmooting=0, Cfg_Dependent=0 and Cfg_GainBumpless=0.

$dD(k)$ = Cfg_DGain(k) * $\Delta Ec(k)$ – Cfg_DGain(k–1) * $\Delta Ec(k-1)$

If derivative filter is enabled (Cfg_UseDSmooting=1), the calculation is as follows.

DCoef parameter for dependent gains Cfg_Dependent=1.

$$DCoef(k) = \frac{\text{Cfg_PGain(k)} \cdot \text{Cfg_DGain(k)} * 60}{Ts + \frac{\text{Cfg_DGain(k)} * 60}{N}}$$

For independent gains Cfg_Dependent=0.

$$DCoef(k) = \frac{\text{Cfg_DGain(k)} * 60}{Ts + \frac{\text{Cfg_DGain(k)} * 60}{N * \text{Cfg_PGain(k)}}}$$

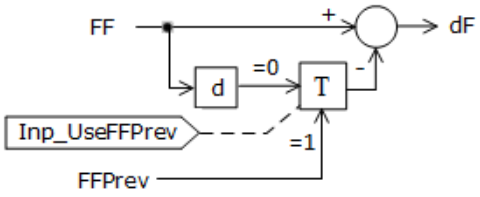
Contribution of derivative term if Cfg_UseDSmooting=1 and Cfg_GainBumpless=1.

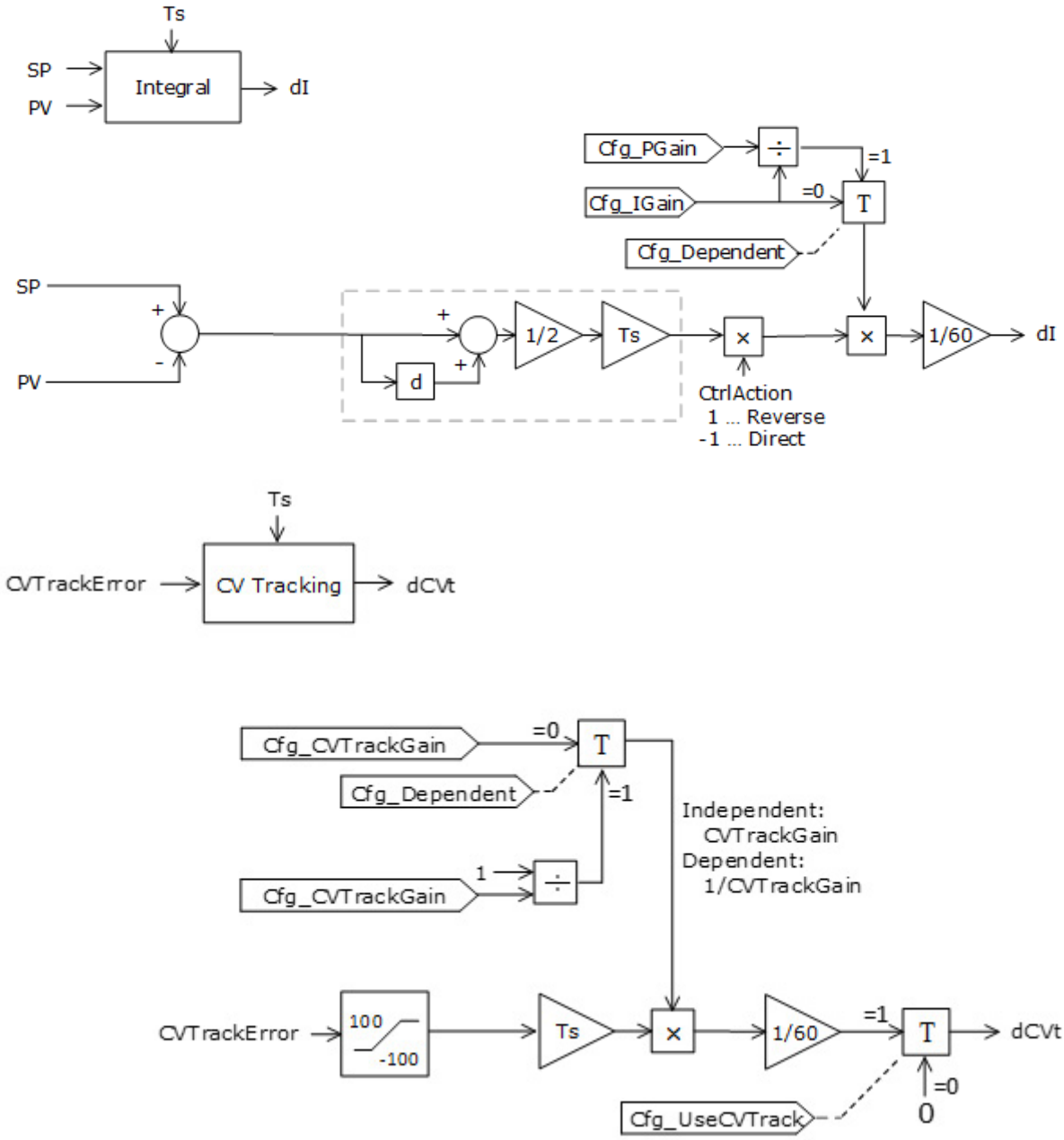
$$dD(k) = DCoef(k)\left(\left(\Delta E_c(k) - \Delta E_c(k-1)\right) + \frac{1}{N \cdot \text{Cfg_PGain(k)}} dD(k-1)\right)$$

Contribution of derivative term if Cfg_UseDSmooting=1 and Cfg_GainBumpless=0.

$$dD(k) = DCoef(k) \cdot \Delta E_c(k) - DCoef(k-1) \cdot \Delta E_c(k-1) + \frac{DCoef(k)}{N \cdot \text{Cfg_PGain(k)}} dD(k-1)$$

Feedforward term.

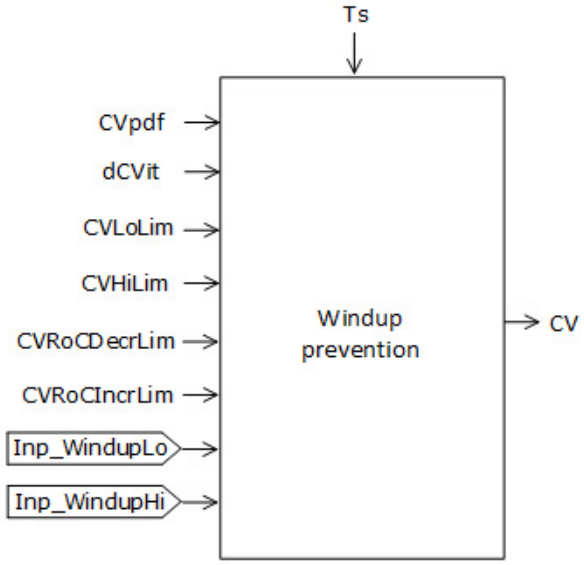


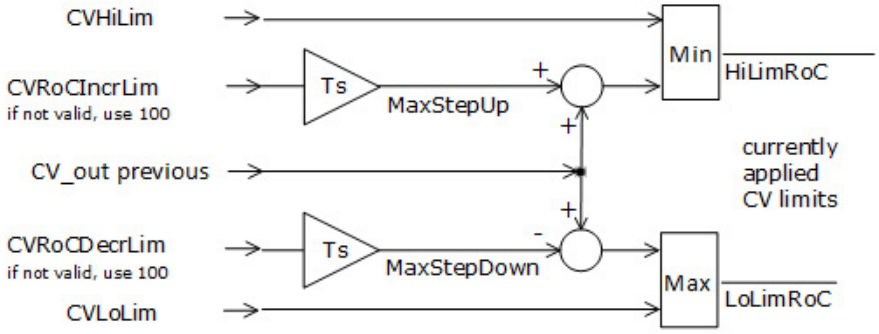


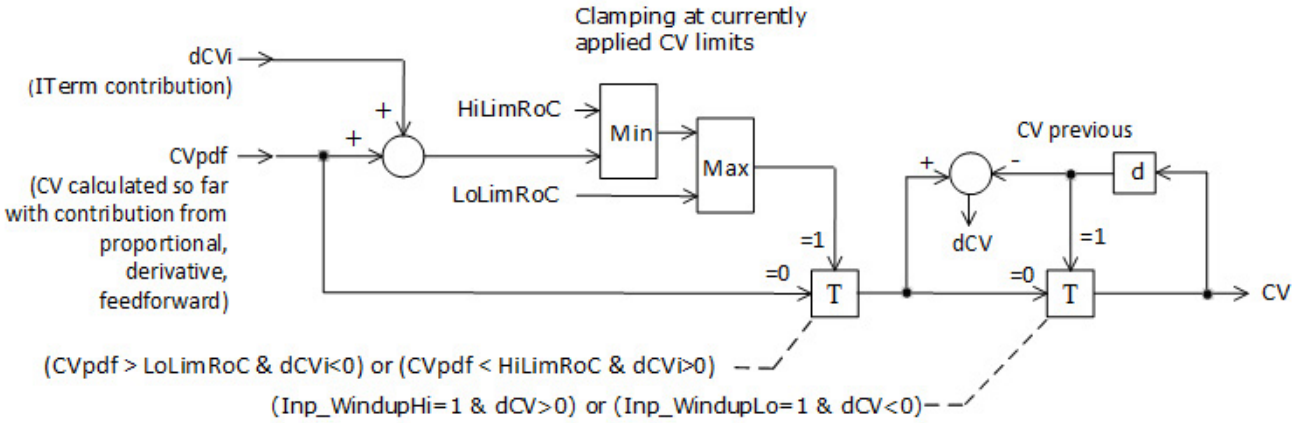
The windup prevention mechanism monitors the internal CV value and drives the integral term. The windup prevention mechanism uses these rules:

- If the internal CV is above the currently applied upper limit, do not integrate upward.
- If the internal CV is within the currently applied limits, integrate but do not violate limits.

- If the internal CV is below the currently applied lower limit, do not integrate downward.





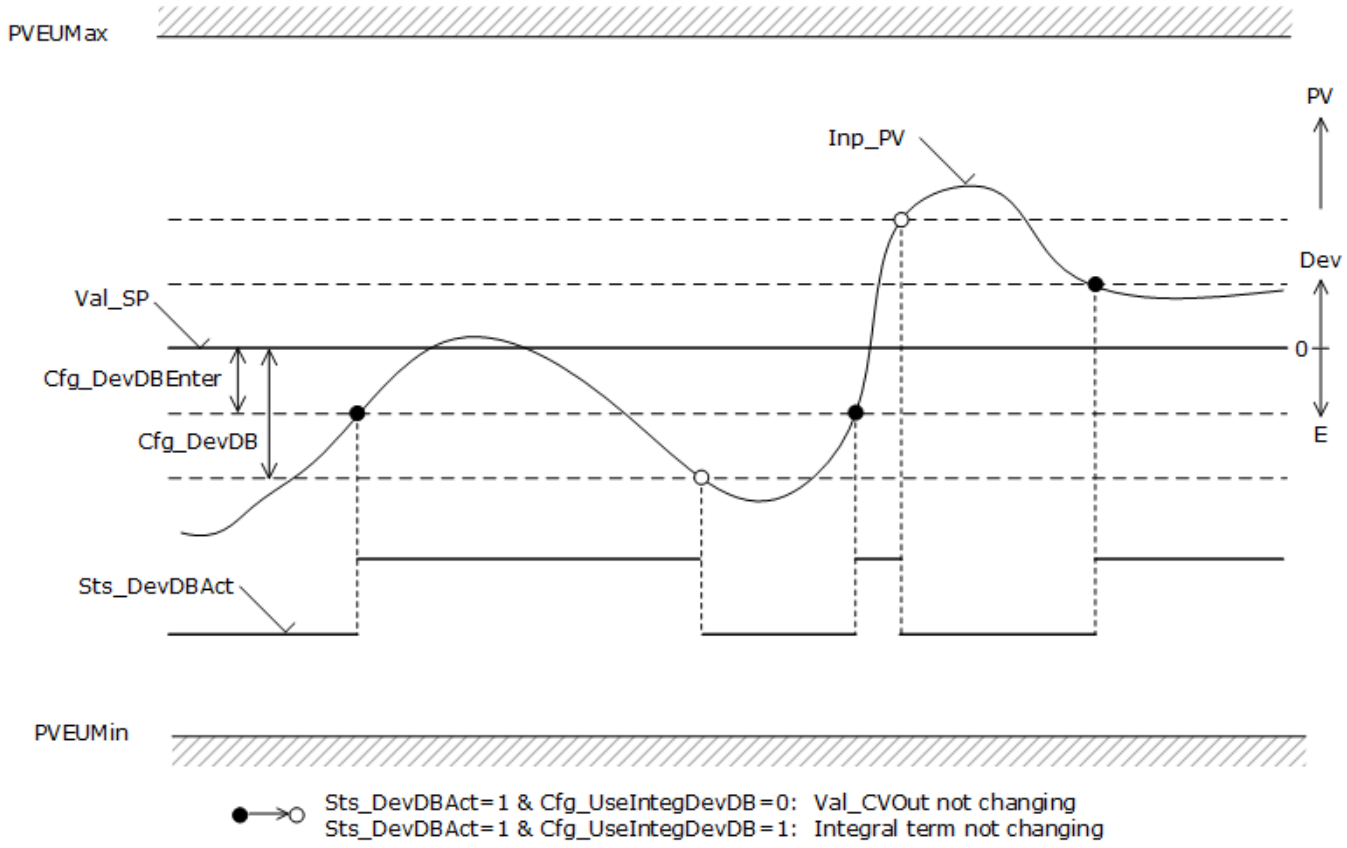


Position PID algorithm

The PPID instruction is configured for the position form of the PD algorithm without bumpless transfer from Manual to Auto or Cascade if Cfg_PositionBump = 1. This option is selected when Cfg_PositionBump = 1 and Cfg_IGain = 0 and change of proportional gain is not bumpless even if Cfg_GainBumpless =1. This feature can also be applied to the velocity PID algorithm to behave as it would work on error and not error change.

Deviation deadband

CV is not sensitive to loop error variation when within the band around zero error. Configured band levels allow for additional hysteresis. Deadband level for PV approaching SP (Cfg_DevDBEnter) may be set lower than deadband level for PV going away from SP (Cfg_DevDB). Active deadband status (Sts_DevDBAct) is set for PV within deadband. An option to stop CV moves or just stop the integration while leaving proportional and derivative action live when in deadband is provided (Cfg_UseIntegDevDB).



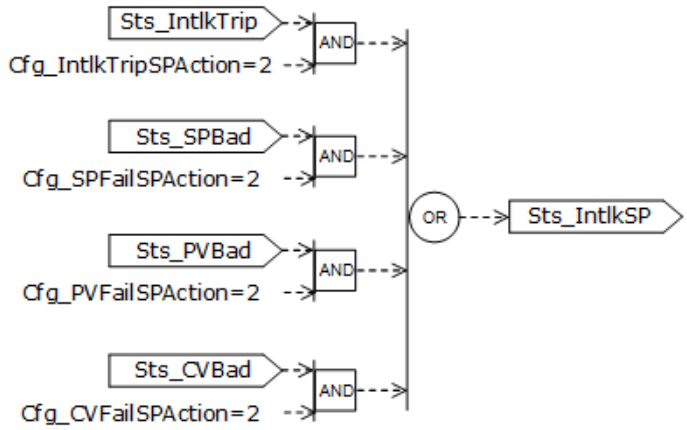
SP handling

The PPID instruction checks the setpoint for validity before the SP value is provided to the PID algorithm for processing. SP fails if:

- SP clamping limits are invalid (Sts_ErrSPLim=1), or
- PV scaling limits are invalid (Sts_ErrPVEU=1), or
- The loop mode is Cascade and cascade SP is NaN or Inf, or
- Cascade is Ratio (Cfg_HasRatio=1) and ratio clamping limits are invalid.

SP value (Val_SPSet) is shed to the configured setpoint value Cfg_SPIntlk and the status flag is set (Sts_IntlkSP=1) if:

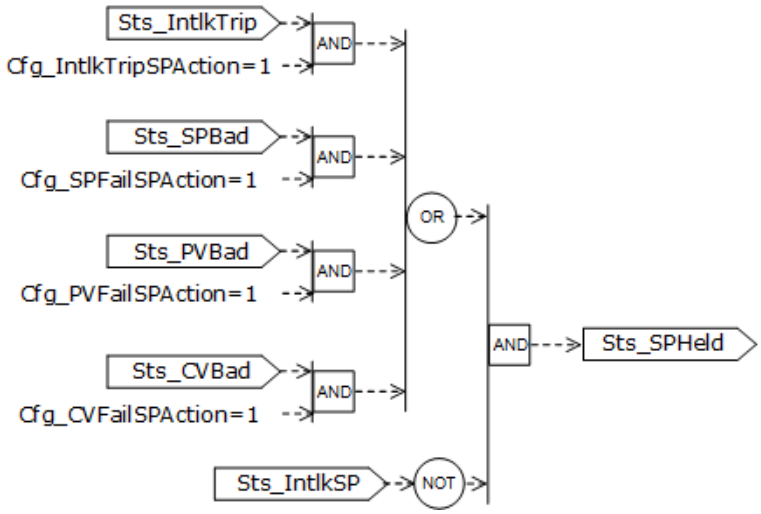
- Interlock trips and the follow up action is to use Cfg_SPIntlk as the setpoint value(Cfg_IntlkTripSPAction=2), or
- SP fails and the instruction is configured to follow with using Cfg_SPIntlk as the setpoint (Cfg_SPFailSPAction=2), or
- PV fails and the instruction is configured to use Cfg_SPIntlk as the setpoint value (Cfg_PVFailSPAction=2), or
- CV fails and the instruction is configured to use Cfg_SPIntlk as the setpoint value (Cfg_CVFailSPAction=2).



SP value holds, shed to the current SP value and the status flag is set (Sts_SPHeld=1), if:

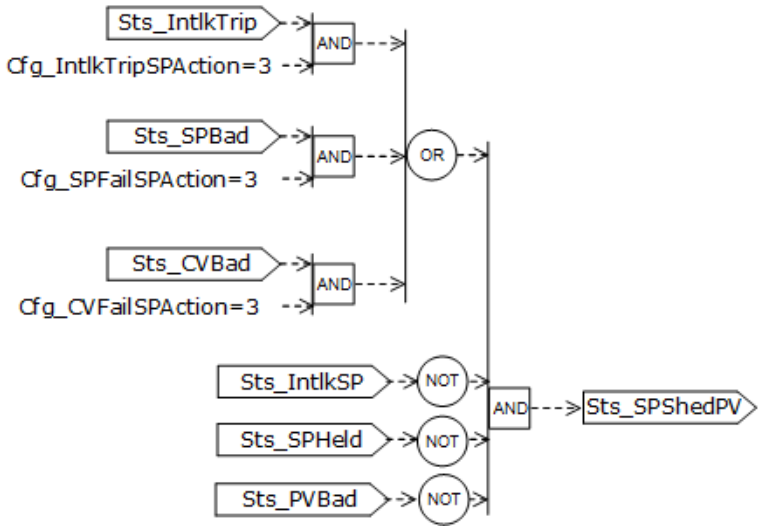
- Interlock trips and the follow up action is to leave the value unchanged (Cfg_IntlkTripSPAction=1), or
- SP fails and the instruction is configured to follow with the setpoint not changed (Cfg_SPFailSPAction=1), or
- PV fails and the instruction is configured to follow with the setpoint not changed (Cfg_PVFailSPAction=1), or

- CV fails and the instruction is configured to follow with the setpoint not changed (Cfg_CVFailSPAction=1) and higher shed priority is not applied (Sts_IntlkSP=0).



SP value sets to the current PV value (Val_SPSet=Val_PV) and the status flag is set (Sts_SPShedPV=1) if:

- Interlock trips and the follow up action is to leave the value unchanged (Cfg_IntlkTripSPAction=3), or
- SP fails and the instruction is configured to follow with the setpoint not changed (Cfg_SPFailSPAction=3), or
- CV fails and the instruction is configured to follow with the setpoint not changed (Cfg_CVFailSPAction=3) and higher shed priority is not applied (Sts_IntlkSP=0 & Sts_SPHeld=0) and PV is good (Sts_PVBad=0).



Shed conditions win over other SP selections, in Auto from Program, Operator, Override SP, in Cascade from CascSP, in Manual from PV to track.

If SP is not from shed and SP clamping limits are valid, other sources for SP value are checked.

If the loop is not Cascade and Operator has SP, OSet_SP is used.

If the loop is not Cascade and Program has SP, PSet_SP is used.

If the loop is not Cascade and External has SP, XSet_SP is used.

If the loop is not Cascade and Override is selected, Inp_OvrdSP is used.

For the loop in Cascade Inp_CascSP × Val_Ratio is used. If Cascade is not Ratio Val_Ratio=1.

The instruction is ready to receive a new SP from the operator (ORdy_SP) if:

- Tracking is not enabled (Cfg_SetTrack=0), or
- Operator has loop mode in Auto, or Operator has loop mode in Manual and SP does not track PV in Manual (Cfg_PVTrack=0), or
- Command source is Hand in Auto, or command source is Hand in Manual and SP does not track PV in Manual (Cfg_PVTrack=0), or
- Command source is Override and SP is not configured for tracking in Override (Cfg_SetTrackOvrdHand=0).

SP tracks PV value (Val_PV) and sets tracking status Sts_SPTrackPV=1 if loop mode is Manual, and instruction is configured for tracking in Manual (Cfg_PVTrack=1), and no SP shed condition is active (Sts_SPShed=0), and PV is not bad (Sts_PVBad=0).

SP is clamped at Cfg_SPHiLim and Cfg_SPLoLim and clamping status is set when clamped (Sts_SPClamped=1) if SP is not held by shed (Sts_SPHeld=0), and SP is not set to interlock SP by shed (Sts_IntlkSP=0), and SP clamping limits are valid (Sts_ErrSPLim=0).

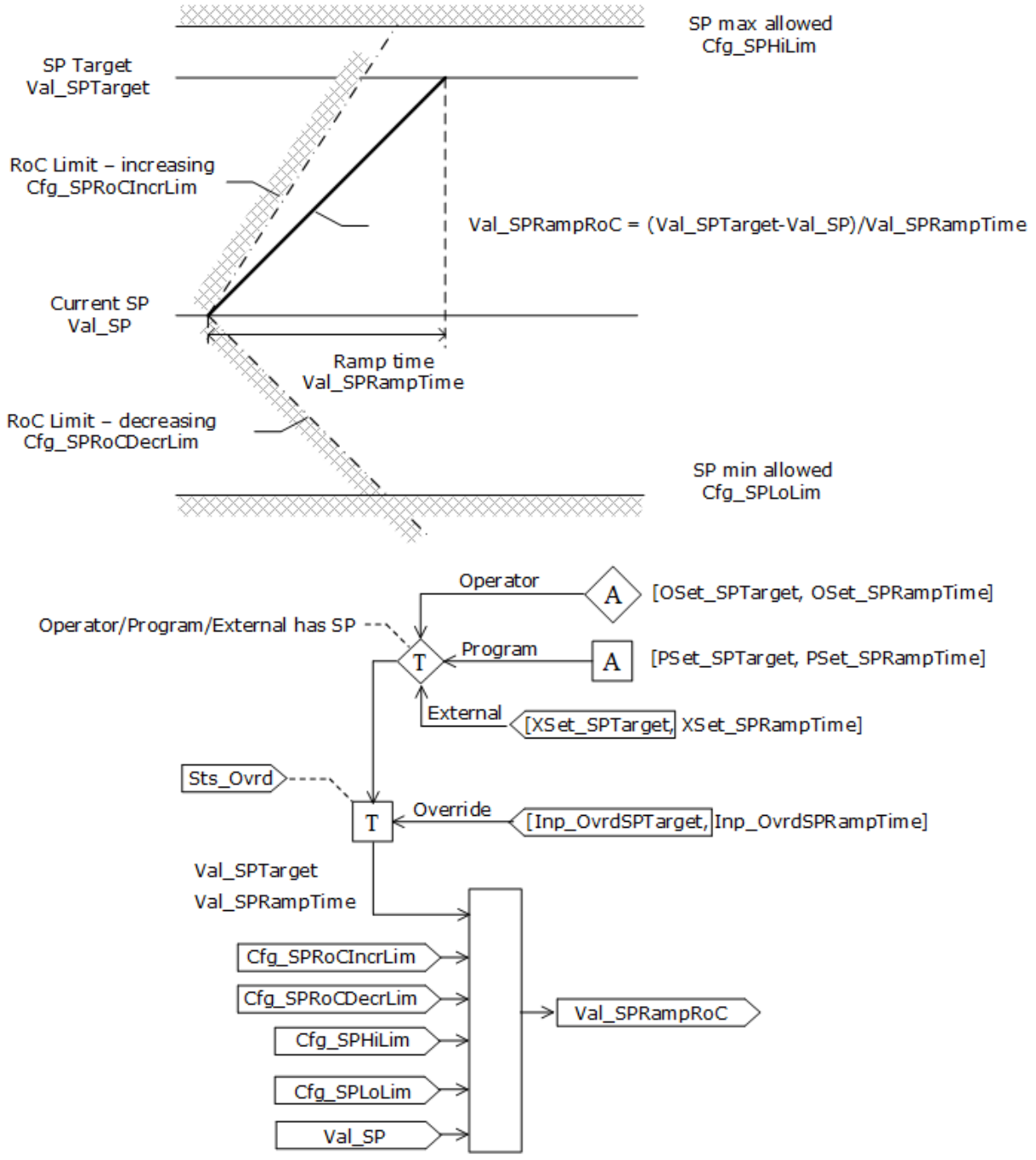
SP ramp wizard

SP ramp wizard allows Operator/Program/External/Override to enter the SP target and time to reach target (ramp time) and calculate SP moves to reach entered target in entered time when starting from current SP when the wizard is commanded to start. Ramping characteristic is defined by the SP ramp target (SPTarget) and SP ramp time (SPRampTime) as described below.

SP target and ramp time entry is permitted (ORdy_SPTarget=1, ORdy_SPRampTime=1) and Val_SPTarget copied to Val_SPSet if:

- Setpoint ramp wizard is permitted (Cfg_HasSPRamp=1), and Program/Operator/External setting is not tracked (Cfg_SetTrack=0), or

- Operator has SP in Auto, or command source is Hand in Auto loop mode.



SP ramping

Setpoint ramping prevents both CV spikes and bumps by eliminating sudden setpoint changes. Although setpoint ramping can be an advantage for single loop control or for the primary (outer) controller of a cascade configuration, neither setpoint ramping nor proportional-on-PV should be used on the secondary (inner) controller of a cascade loop, as that would degrade the responsiveness of the secondary loop.

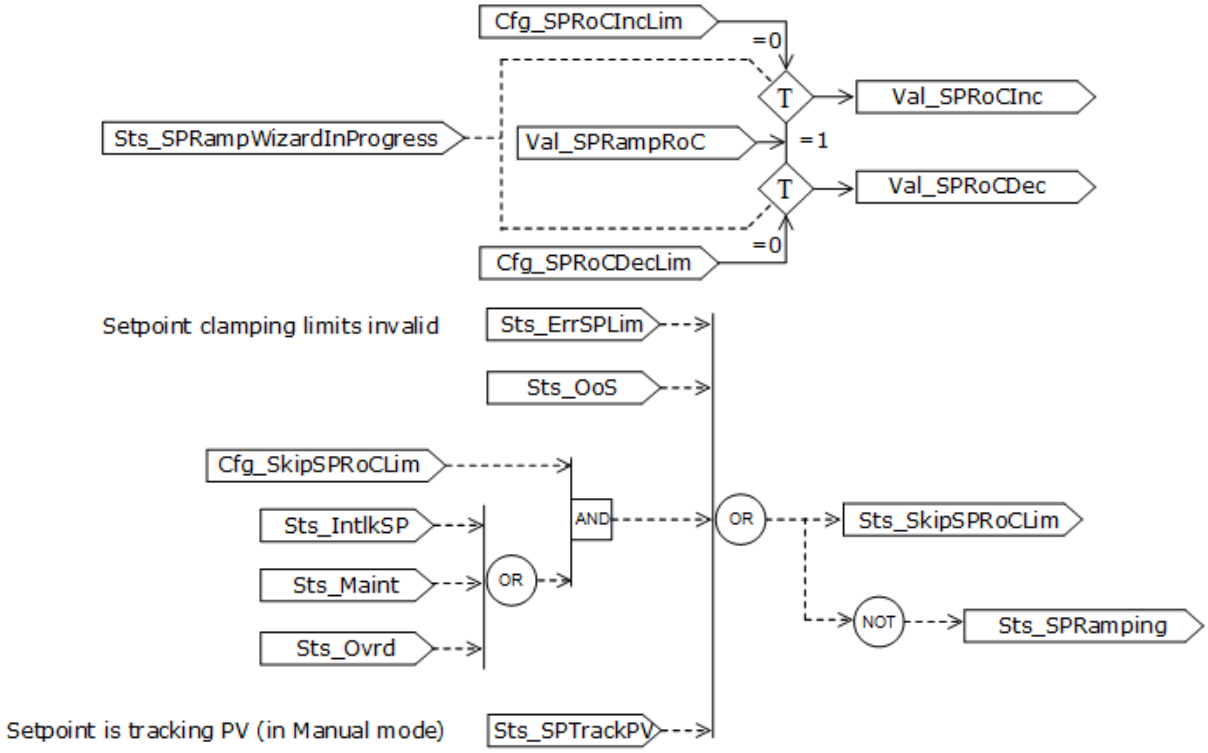
SP ramping is skipped, SP target (Val_SPSet) is used directly for final setpoint (Val_SP), skip status is set Sts_SkipSPRoCLim=1, and ramp reported complete Sts_SPRamping=0 under these conditions:

- Setpoint clamping limits are invalid, Sts_ErrSPLim=1, or
- PPID is Out of Service, or
- PPID is configured to skip RoC limiting Cfg_SkipSPRoCLim=1 and the PPID is in Interlock/Maintenance/Override, or
- SP value tracks PV value in Manual loop mode (Sts_SPTrackPV=1), or
- Current value of SP rate of change limit increasing is zero (Val_SPRoCIncr=0) and current SP target (Val_SPSet) is above current SP, or
- Current value of SP rate of change limit decreasing is zero (Val_SPRoCDecr=0) and current SP target (Val_SPSet) is below current SP.

SP ramping executes if these bullets are all true:

- not skipped (Sts_SkipSPRoCLim=0),
- the setpoint has not reached its target Val_SPSet,
- absolute value of loop deviation (|Val_E|) does not exceed configured maximum ramp deviation, |Val_E|≤ Cfg_SPRampMaxDev.

If |Val_E|> Cfg_SPRampMaxDev, then actual RoC limits Val_SPRoCDecr and Val_SPRoCIncr are set to zero and the SP does not change.



SP scaling

If PV scaling limits are valid (Sts_ErrPVEU=0) SP in PVEU is scaled to percent used in PID calculation, SP = (SPEU-Cfg_PVEUMin)/(Cfg_PVEUMax-Cfg_PVEUMin)×100.

Ratio selection and clamping

The PPID instruction is ready for the Operator to enter a Ratio if the Cascade loop mode is Ratio (Cfg_HasRatio=1), ratio clamping limits are not invalid, and the PPID instruction is not configured for tracking, or Operator has Ratio, or in Hand.

If the PPID instruction allows Cascade-Ratio (Cfg_HasRatio=1) and ratio clamping limits are not invalid, then the source for Ratio is selected based on the command source:

- PSet_Ratio if Program has Ratio,
- OSet_Ratio if Operator has Ratio,
- XSet_Ratio if External has Ratio,
- Inp_OvrdRatio if Override is selected.

If the PPID instruction allows Cascade-Ratio (Cfg_HasRatio=1) and ratio clamping limits are not invalid, Ratio is clamped at Cfg_RatioHiLim and Cfg_RatioLoLim. If clamping is active Sts_RatioClamped is set.

If the loop does not have Ratio or Ratio clamping limits are invalid, the Ratio is set to 1.0 (Val_Ratio = 1.0).

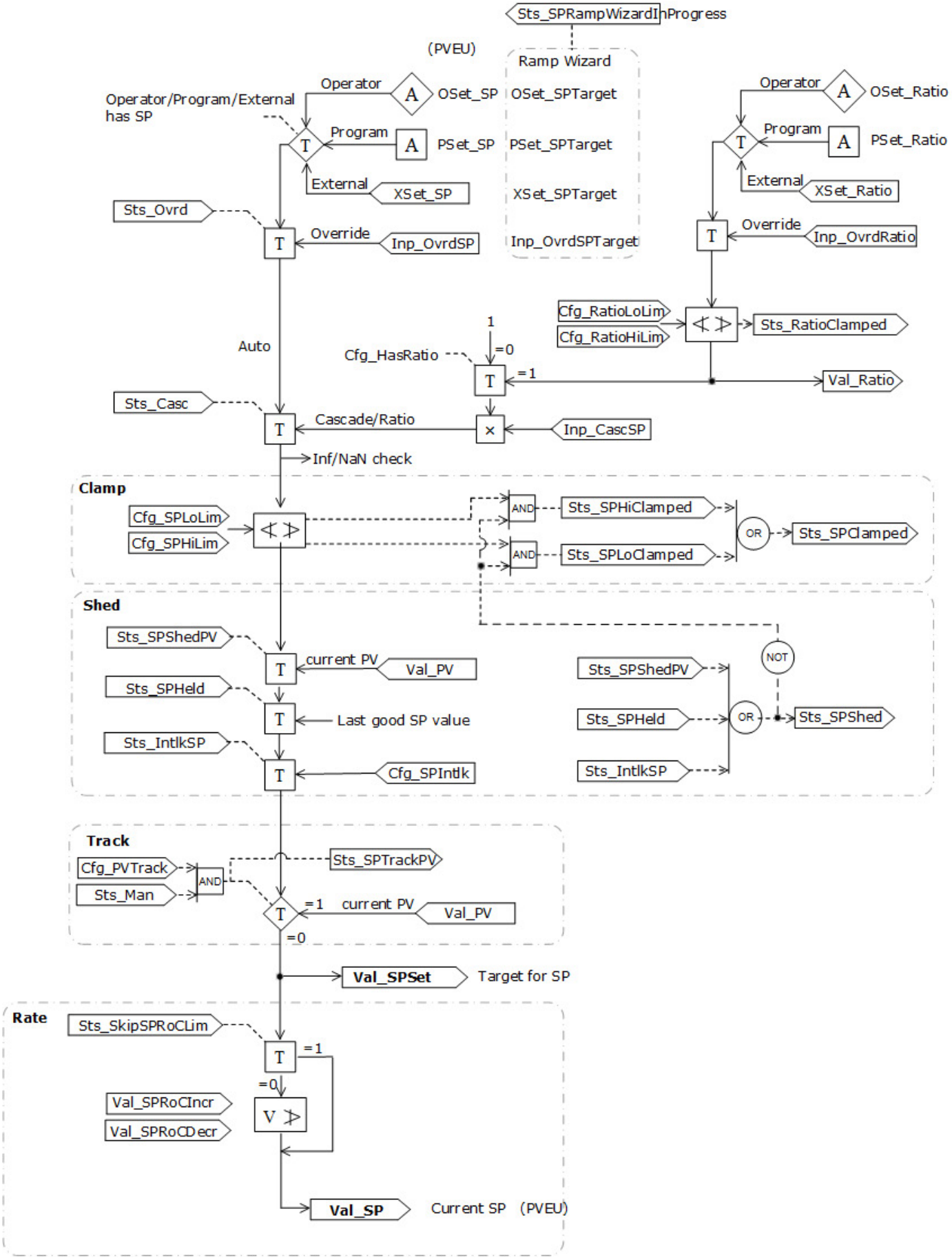
SP and Ratio tracking for bumpless transfer

Val_SPSet is copied back to all inactive SP settings to allow bumpless transfer when the SP owner changes.

SP tracking is applied if PPID is configured for it (Cfg_SetTrack=1) and:

- Loop mode is not in Override nor Hand, or
- Loop mode is in Override or Hand and PPID is configured to track in Override and Hand loop mode (Cfg_SetTrackOvrdHand=1), or
- Loop mode is in Cascade, or
- Loop mode is in Manual and Cfg_PVTrack=1.

This diagram shows the main steps in SP processing before the SP value enters the PID formula for calculating CV.


Sts_SPRampWizardInProgress
(PVEU)
Ramp Wizard
Operator
A
OSet_SP
OSet_SPTarget
Operator/Program/External has SP
Program
PSet_SP
PSet_SPTarget
External
XSet_SP
XSet_SPTarget
Sts_Ovrd
Override
Inp_OvrdSP
Inp_OvrdSPTarget
Operator
OSet_Ratio
Program
PSet_Ratio
External
XSet_Ratio
Override
Inp_OvrdRatio
Cfg_RatioLoLim
Cfg_RatioHiLim
Sts_RatioClamped
Val_Ratio
Auto
Cfg_HasRatio
1
=0
=1
Sts_Casc
Cascade/Ratio
Inp_CascSP
Inf/NaN check
Clamp
Cfg_SPLoLim
Cfg_SPHiLim
AND
Sts_SPHiClamped
Sts_SPLoClamped
OR
Sts_SPClamped
Shed
Sts_SPShedPV
current PV
Val_PV
Sts_SPHeld
Last good SP value
Sts_IntlkSP
Cfg_SPIntlk
NOT
Sts_SPShed
Track
Cfg_PVTrack
Sts_Man
Sts_SPTrackPV
Val_SPSet
Target for SP
Rate
Sts_SkipSPRoCLim
Val_SPRoCIncr
Val_SPRoCDecr
Val_SP
Current SP (PVEU)

CV handling

CV selection in Manual

In Manual loop mode, if CV clamping limits are valid, the CV is selected from the active source (Program/ Operator/ External/ Override). If Program has CV, PSet_CV is selected, and so on. The PPID instruction is ready for OSet_CV (ORdy_CV=1) under the following conditions: operator entry is not configured for tracking in Program/Operator/External (Cfg_SetTrack=0), or command source is Hand or Override and operator entry is not configured for tracking in Hand or Override (Cfg_SetTrackOvrdHand=0).

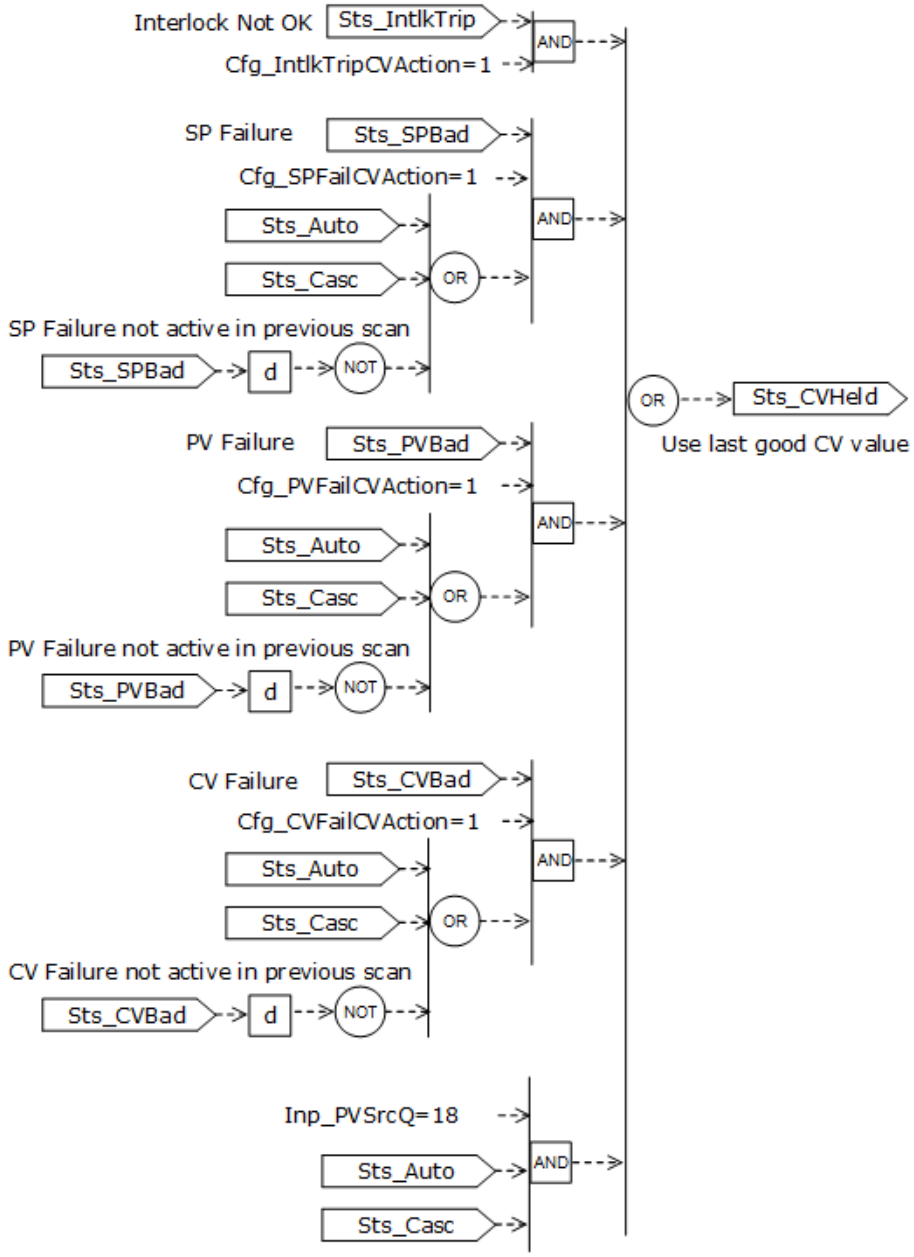
CV shed

Val_CVSet holds the last good CV value and the CV value replacement status is set (Sts_CVHeld=1) when

- Interlock trips and CV action is configured for last good CV value (Cfg_IntlkTripCVAction=1), or
- SP fails and follow up action on CV is configured for last good CV value (Cfg_SPFailCVAction=1) and the loop mode is Auto or Cascade or SP did not fail in previous scan (SP fail rising edge), or
- PV fails and follow up action on CV is configured for last good CV value (Cfg_PVFailCVAction=1) and the loop mode is Auto or Cascade or PV did not fail in previous scan (PV fail rising edge), or
- PV substituted at PAI (Inp_PVSrcQ=18) and the loop mode is Auto or Cascade, or
- CV fails and follow up action on CV is configured for last good CV value (Cfg_CVFailCVAction=1) and the loop mode is Auto or Cascade or CV did not fail in previous scan (CV fail rising edge).

For Sts_CVHeld=1 the CV is not updated.

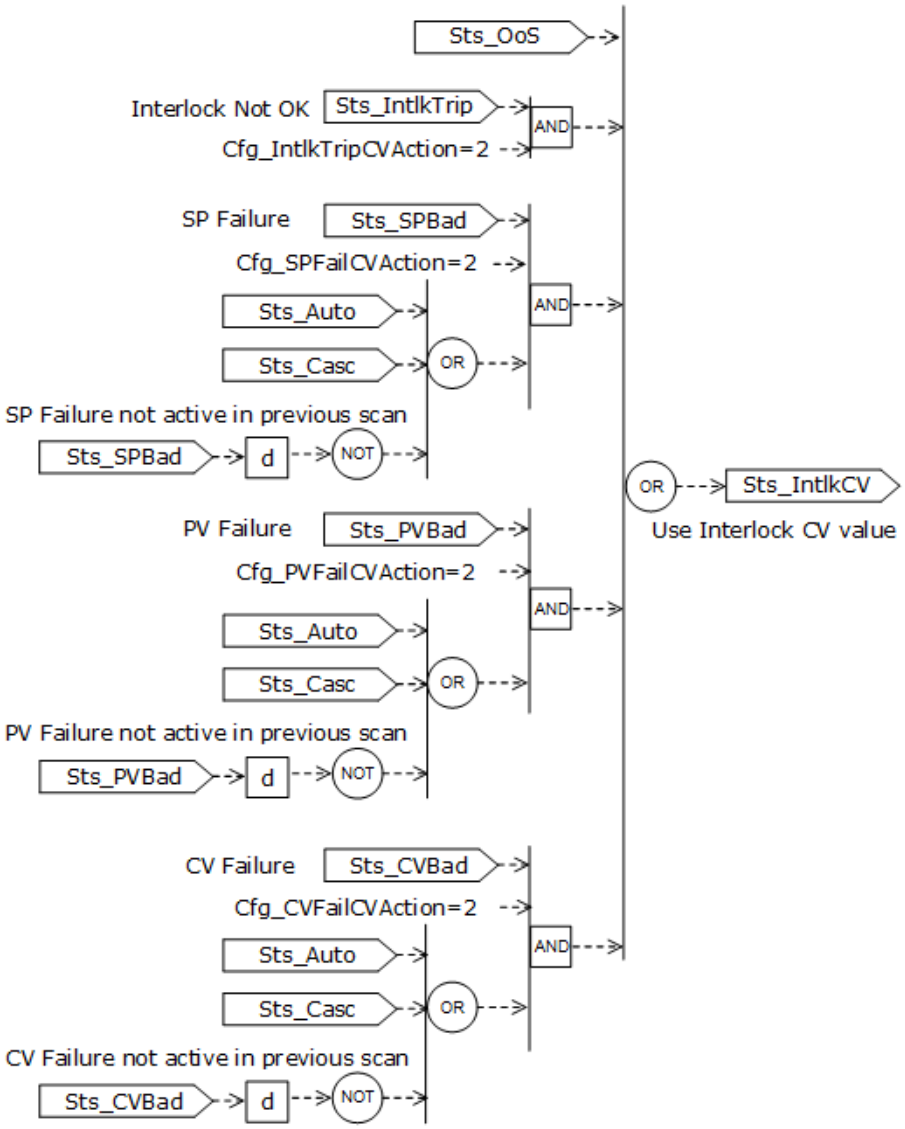
Note: When PV, CV, or SP fails, the last good CV value stays at output, but it can be still overwritten in Manual loop mode.



Val_CVSet sets to Cfg_CVIntlk and the CV value replacement status is set (Sts_IntlkCV=1) when:

- Interlock trips and CV action is configured for Cfg_CVIntlk (Cfg_IntlkTripCVAction=2), or
- SP fails and follow up action on CV is configured for Cfg_CVIntlk (Cfg_SPFailCVAction=2) and loop mode is Auto or Cascade or SP did not fail in previous scan (SP fail rising edge), or
- PV fails and follow up action on CV is configured for Cfg_CVIntlk (Cfg_PVFailCVAction=2) and loop mode is Auto or Cascade or PV did not fail in previous scan (PV fail rising edge), or
- CV fails and follow up action on CV is configured for Cfg_CVIntlk (Cfg_CVFailCVAction=2), and loop mode is Auto or Cascade or CV did not fail in previous scan (CV fail rising edge), or
- PPID is out of service.

Note: When PV, CV, or SP fails, the Cfg_CVIntlk value stays at output, but it can be still overwritten in Manual loop mode.



If CV value is being set by shed, to interlock CV or hold last good CV value, Sts_CVShed=1.

CV clamping

If CV clamping limits are valid (Sts_ErrCVLim=0) and the loop mode is not Manual, or is Manual but the configuration is set not to skip CV clamping (Cfg_SkipCVManLim=0), the calculated or set value of CV (CVSet) is clamped at Cfg_CVHiLim and Cfg_CVLoLim. The corresponding status bits (Sts_CVHiClamped, Sts_CVLoClamped, and Sts_CVClamped) are set if the CV is clamped.

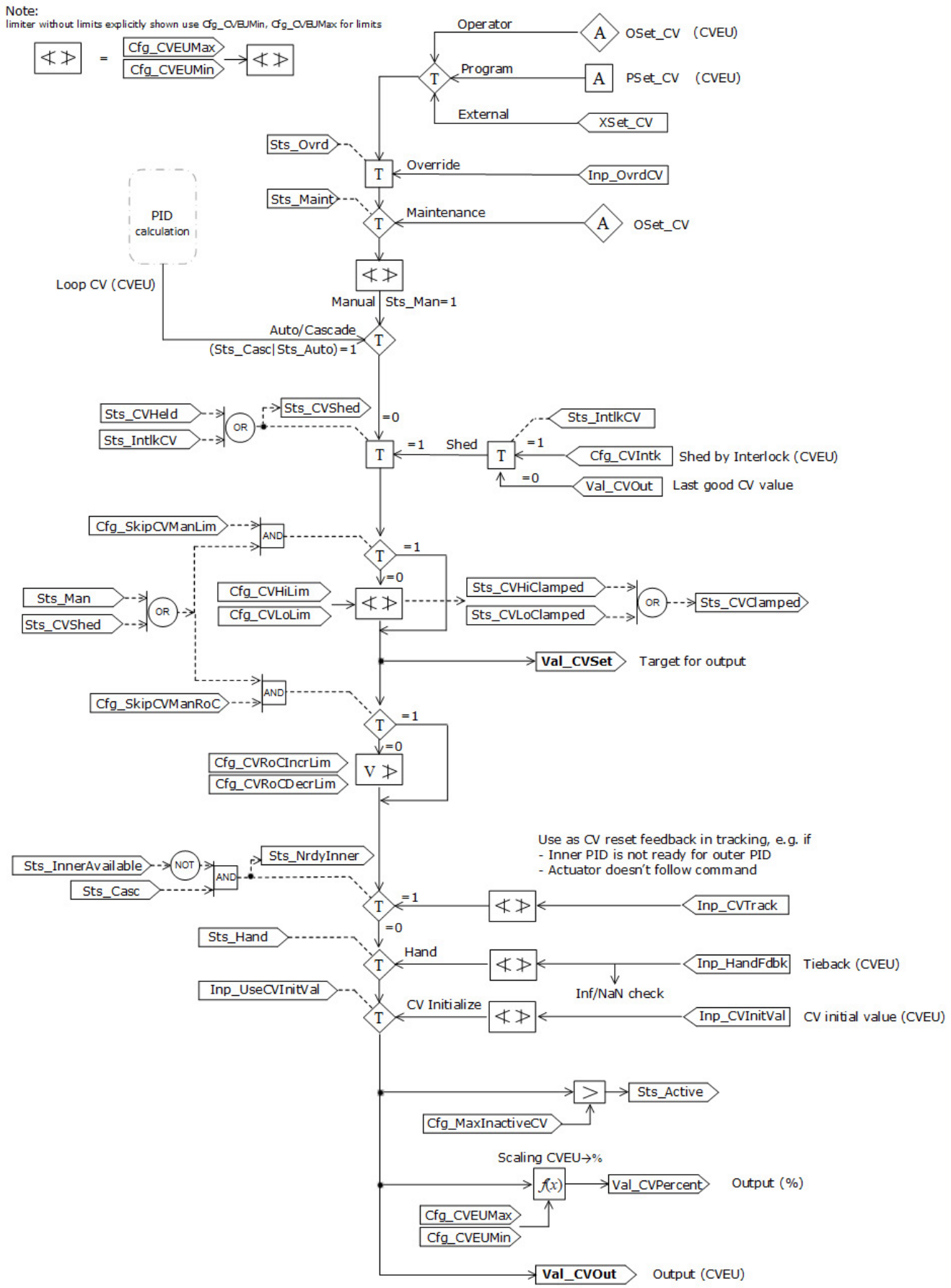
CV rate of change limiting

If the PPID instruction is not configured to skip CV RoC limiting in Manual (Cfg_SkipCVManRoC=0) and the loop mode is Manual and PPID is configured for RoC limiting (Cfg_CVRoCIncrLim>0, Cfg_CVRoCDecrLim>0) and the target value of CV is greater than CVOut_previous+Cfg_CVRoCIncrLim*Ts (Ts is current PPID scan time), the CVOut is calculated as CVOut= CVOut_previous+Cfg_CVRoCIncrLim*Ts. If the target value of CV is lower than CVOut_previous-Cfg_CVRoCDecrLim*Ts (Ts is current PPID scan time) then CVOut= CVOut_previous-Cfg_CVRoCDecrLim*Ts. If ramping is active, status bits are set (Sts_CVRampingUp/Sts_CVRampingDown=1, Sts_CVRamping=1). The status bits are cleared if CVOut reaches the target. If ramping is not active the CV target is copied to CVOut.

CV output in percent

CVOut, which is calculated in Auto/Cascade or entered from various sources in Manual, is scaled to percent and made available as Val_CVOutPercent. Val_CVOutPercent=0 if CV scaling limits are invalid.

This diagram shows the main steps in CV processing.


Note:
limiter without limits explicitly shown use Cfg_CVEUMin, Cfg_CVEUMax for limits
Cfg_CVEUMax
Cfg_CVEUMin
Operator
OSet_CV (CVEU)
Program
PSet_CV (CVEU)
External
XSet_CV
Sts_Ovrd
Override
Inp_OvrdCV
Sts_Maint
Maintenance
OSet_CV
PID calculation
Loop CV (CVEU)
Manual Sts_Man=1
Auto/Cascade
(Sts_Casc|Sts_Auto)=1
Sts_CVHeld
Sts_IntlkCV
OR
Sts_CVShed
=0
=1
Shed
Sts_IntlkCV
Cfg_CVIntk
Shed by Interlock (CVEU)
Val_CVOut
Last good CV value
Cfg_SkipCVManLim
AND
Sts_Man
Sts_CVShed
Cfg_CVHiLim
Cfg_CVLoLim
Sts_CVHiClamped
Sts_CVLoClamped
Sts_CVClamped
Val_CVSet
Target for output
Cfg_SkipCVManRoC
Cfg_CVRoCIncrLim
Cfg_CVRoCDecrLim
Use as CV reset feedback in tracking, e.g. if
- Inner PID is not ready for outer PID
- Actuator doesn't follow command
Sts_InnerAvailable
NOT
Sts_NrdyInner
Sts_Casc
Inp_CVTrack
Sts_Hand
Hand
Inp_HandFdbk
Tieback (CVEU)
Inf/NaN check
Inp_UseCVInitVal
CV Initialize
Inp_CVInitVal
CV initial value (CVEU)
Sts_Active
Cfg_MaxInactiveCV
Scaling CVEU→%
f(x)
Val_CVPercent
Output (%)
Cfg_CVEUMax
Cfg_CVEUMin
Val_CVOut
Output (CVEU)

Interlock handling

Maintenance commands to bypass or check bypassable interlocks are processed. Interlock bypassing is active if requested (MCmd_Bypass=1). The bypassing request remains active (Sts_Bypass=1) until a maintenance command to check bypassable alarms (MCmd_Check=1) is received. Bypassing is active (Sts_BypActive=1) if requested (Sts_Bypass=1), or in Maintenance (Sts_Maint=1), or in Override (Sts_Ovrd) if the instruction is configured for bypassing interlocks in Override (Cfg_OvrdIntlk). If an Interlock is NOT OK (bypassable or not), and if any action is configured on the Interlock, the Interlock NOT OK latch is shed.

Initialization and Powerup

The instruction must be initialized to execute properly. The instruction is normally initialized automatically in the instruction first run (for example, after power up). Re-initialization can be requested any time by setting Inp_InitializeReq = 1. This operand is cleared in the instruction automatically. For proper initialization, when adding the instruction while performing an online edit of the code, make sure that Inp_InitializeReq = 1 (default value). Action performed in initialization:

- Owner command set to None,
- Override command set to None,
- Maintenance commands to bypass and check interlocks are cleared,
- Operator, Program and External commands are cleared,
- latched shed faults are cleared,
- all timers are reset.

This section defines PPID actions on Loop mode, SP and CV in initialization (Power up).

Loop mode handling in Power up Normal:

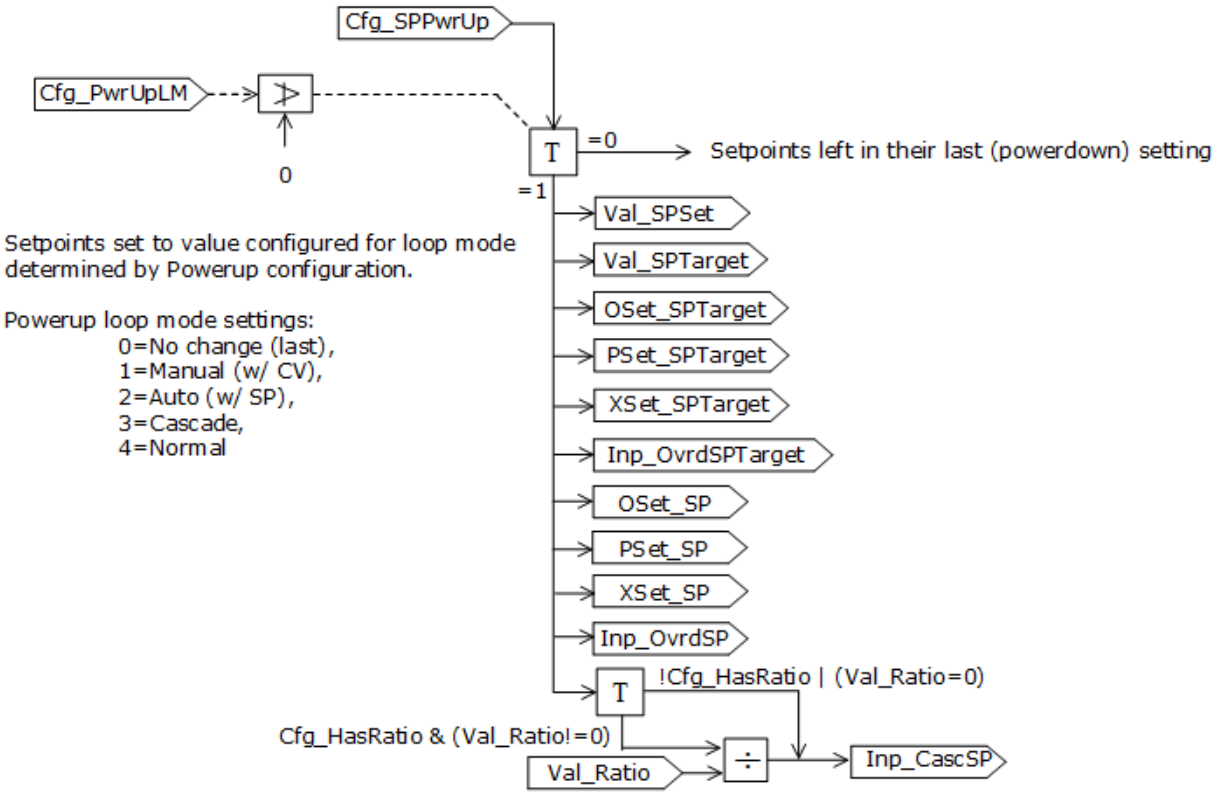
- Normal loop mode is disabled (Cfg_NormLM=0) if loop mode defined as normal is invalid (Cfg_NormLM<0 | >3), or normal loop mode is set Cascade but the instruction does not allow Cascade in configuration (Cfg_HasCasc ≠ 1 & Cfg_NormLM = 3),
- Loop mode is set to Manual if loop mode for Power up (Cfg_PwrUpLM) is Normal and normal loop mode is Manual, or loop mode for Power up in Manual,
- Loop mode is set to Auto if normal loop mode for Power up is Normal and normal loop mode is Auto, or loop mode for Power up in Auto,
- Loop mode is set to Cascade if normal loop mode for Power up is Normal and normal loop mode is Cascade, or loop mode for Power up in Cascade.

Cascade SP handling in Power up:

- Inp_CascSP = Cfg_SPPwrUp if loop mode in Powerup (Cfg_PwrUpLM) is explicitly provided as Manual/ Auto/ Cascade/ Normal and the Cascade loop mode is not Ratio, or Val_Ratio = 0,
- Inp_CascSP = Cfg_SPPwrUp × Val_Ratio if loop mode in Powerup is explicitly provided as Manual/ Auto/ Cascade/ Normal and the Cascade loop mode is Ratio and Val_Ratio ≠ 0.

SP handling in Power up. All SP inputs, interval SP values, and output SP values are overwritten by SP value configured for Power up:

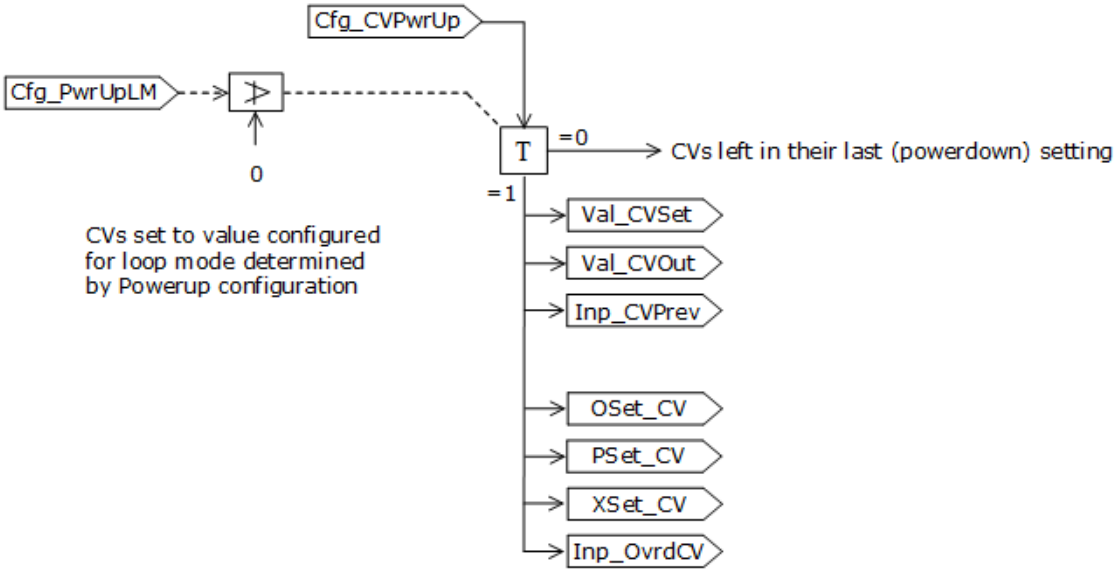
- Cfg_SPPwrUp → PSet_SP, OSet_SP, XSet_SP, Inp_OvrdSP,
- Cfg_SPPwrUp → PSet_SPTarget, OSet_SPTarget, XSet_SPTarget, Inp_OvrdSPTarget,
- Cfg_SPPwrUp → Val_SPTarget, Val_SPSet, Val_SP.



CV handling in Power up. All CV inputs, internal CV values, and output CV values are overwritten by CV value configured for Power up:

- Cfg_CVPwrUp → PSet_CV, OSet_CV, XSet_CV, Inp_OvrdCV, Inp_CVPrev,

- Cfg_CVPwrUp → Val_CVSet, Val_CVOut.

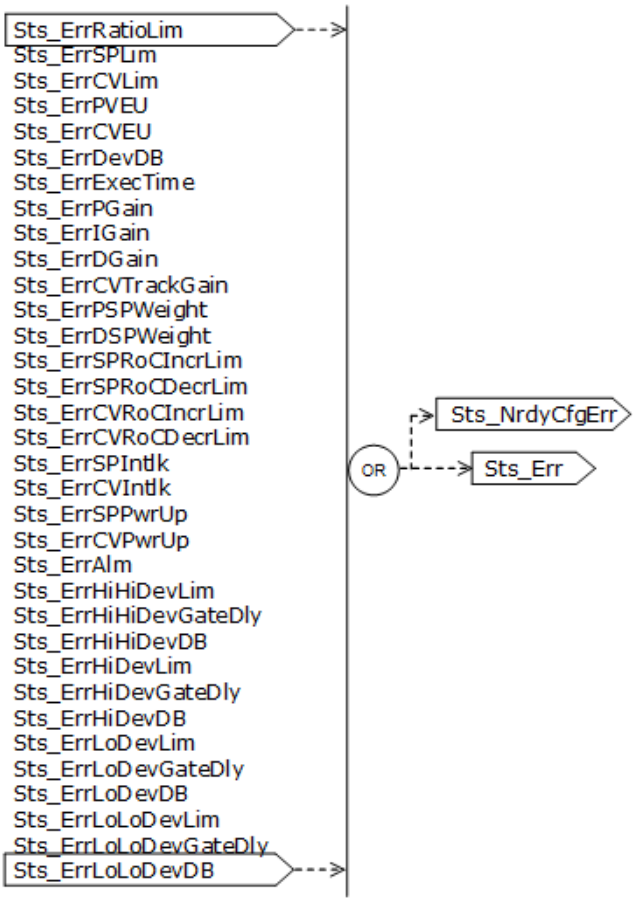


When in Auto or Cascade, CV can be initialized by inner loop CV set in Inp_CVTrack if Inp_InnerAvailable=0 and Cfg_CVPwrupSel=1. If Cfg_CVPwrupSel=0 then Inp_InnerAvailable is ignored in first run and Cfg_CVPwrup is used as initial CV.

If the loop mode was not zero for Power up (Cfg_PwrUpLM) and multiple loop modes were set internally, the instruction makes sure only one loop mode is set at a time, using this order of priority: Manual, then Auto, then Cascade. If the Powerup loop mode is 0, the loop mode, CV, and SP are left in their last (powerdown) states. Once initialization is complete the initialization request is cleared and the instruction is reported initialized (Sts_Initialized = 1).

Configuration error report

Selected parameters with Cfg_ prefix and data type REAL are examined in a validity check to prevent false actions. An error bit specific to a particular parameter is set if the value is not valid. If an error bit is set, the global error (Sts_Err) is reported and a Fail alarm is raised.



The PPID instruction ensures a valid number of decimal places for PV and CV display in the HMI. Cfg_xxDecPlcs =2 if an invalid number is entered. The PPID instruction ensures valid action on SP/PV/CV fail. Cfg_SPFailSPAction = 1, Cfg_PVFailPVAction = 1, and Cfg_CVFailCVAction = 1 for any invalid number entered. The PPID instruction ensures valid Keep configuration on eKeepLM, CV, Ratio and SP. Cfg_eKeepxx = 0 (follows command source) if an invalid value is entered.

CV, SP, Ratio, Loop Mode ownership

Owned by Operator

The CV is owned by Operator if:

- The current command source is Operator (Sts_Oper=1) and the configuration is set to follow the command source (Cfg_eKeepCV=0), or
- the command source is Operator/Program/External and the configuration keeps the CV for Operator (Cfg_eKeepCV=1),
- or the command source is Maintenance (Sts_Maint=1).

The loop mode is owned by Operator if:

- The current owner is Operator (Sts_Oper=1) and the configuration is set to follow the command source (Cfg_eKeepLM=0), or
- the owner is Operator/Program/External and the configuration keepd the loop mode for Operator (Cfg_eKeepLM=1), or
- the owner is Maintenance (Sts_Maint=1).

The Ratio is owned by Operator if:

- The current owner is Operator (Sts_Oper=1) and the configuration is set to follow the command source (Cfg_eKeepRatio=0), or
- the owner is Operator/Program/External and the configuration keeps the Ratio for Operator (Cfg_eKeepRatio=1), or
- the owner is Maintenance (Sts_Maint=1).

The SP is owned by Operator if:

- The current owner is Operator (Sts_Oper=1) and the configuration is set to follow the command source (Cfg_eKeepSP=0), or
- the owner is Operator/Program/External and the configuration keeps the SP for Operator (Cfg_eKeepSP=1), or
- the owner is Maintenance (Sts_Maint=1).

Owned by Program

The CV is owned by Program if:

- The current owner is Program (Sts_Prog=1) and the configuration is set to follow the command source (Cfg_eKeepCV=0), or
- the owner is Operator/Program/External and the configuration keeps the CV for Program (Cfg_eKeepCV=2).

The loop mode is owned by Program if:

- The current owner is Program (Sts_Prog=1) and the configuration is set to follow the command source (Cfg_eKeepLM=0), or
- the owner is Operator/Program/External and the configuration keeps the loop mode for Program (Cfg_eKeepLM=2).

The Ratio is owned by Program if:

- The current owner is Program (Sts_Prog=1) and the configuration is set to follow the command source (Cfg_eKeepRatio=0), or
- the owner is Operator/Program/External and the configuration keeps the Ratio for Program (Cfg_eKeepRatio=2).

The SP is owned by Program if:

- The current owner is Program (Sts_Prog=1) and the configuration is set to follow the command source (Cfg_eKeepSP=0), or
- the owner is Operator/Program/External and the configuration keeps the SP for Program (Cfg_eKeepSP=2).

Owned by External

The CV is owned by External if:

- The current owner is External (Sts_Ext=1) and the configuration is set to follow the command source (Cfg_eKeepCV=0), or
- the owner is Operator/Program/External and the configuration keeps the CV for External (Cfg_eKeepCV=3).

The Loop mode is owned by External if:

- The current owner is External (Sts_Ext=1) and the configuration is set to follow the command source (Cfg_eKeepLM=0), or
- the owner is Operator/Program/External and the configuration keeps the loop mode for External (Cfg_eKeepLM=3).

The Ratio is owned by External if:

- The current owner is External (Sts_Ext=1) and the configuration is set to follow the command source (Cfg_eKeepRatio=0), or
- the owner is Operator/Program/External and the configuration keeps the Ratio for External (Cfg_eKeepRatio=3).

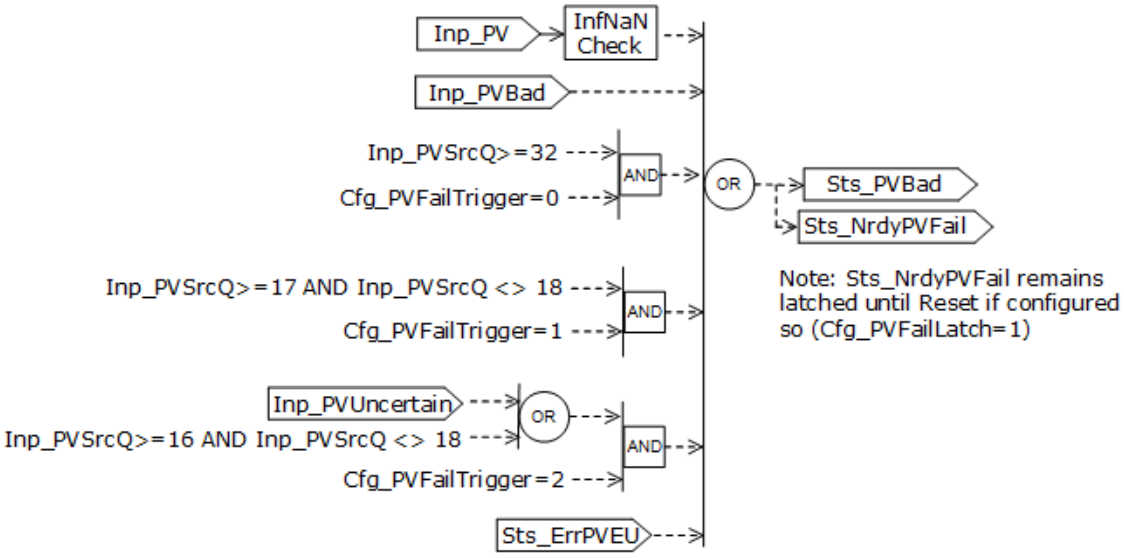
The SP is owned by External if:

- The current owner is Program (Sts_Prog=1) and the configuration is set to follow the command source (Cfg_eKeepSP=0), or
- the owner is Operator/Program/External and the configuration keeps the SP for External (Cfg_eKeepSP=3).

PPID statuses

PV bad status

The Process variable fails in case of invalid PV scaling limits, and the PV is reported bad at source via Inp_PVBad input or quality index Inp_PVSrcQ. The PV cannot be used in PID calculations if the source quality is bad. The PV is also treated as failing if the PV is live but reported uncertain or just off-spec and the PPID is configured for treating this situation as unacceptable (Cfg_PVFailTrigger=1 or 2).

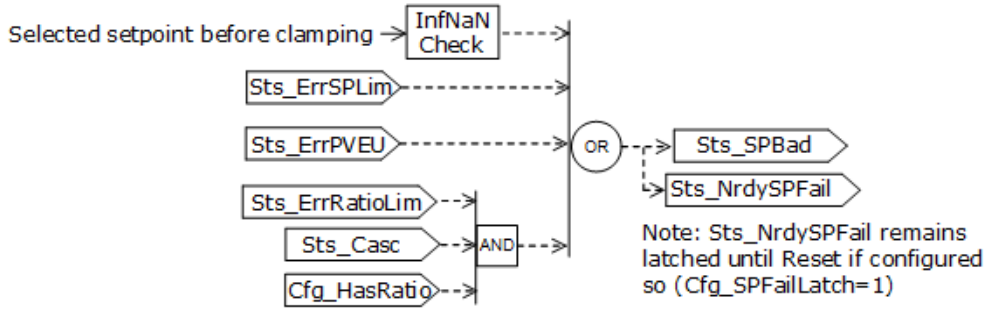


SP bad status

Setpoint fails if:

- SP clamping limits are invalid (Sts_ErrSPLim=1), or
- PV limits used in scaling are invalid (Sts_ErrPVEU=1, Max<=Min).

The failure occurs because SP is derived from PV if the loop mode is Ratio, or ratio clamping limits are invalid when the loop mode is Ratio (Sts_ErrRatioLim=1 & Cfg_HasRatio=1 & loop mode is Cascade).

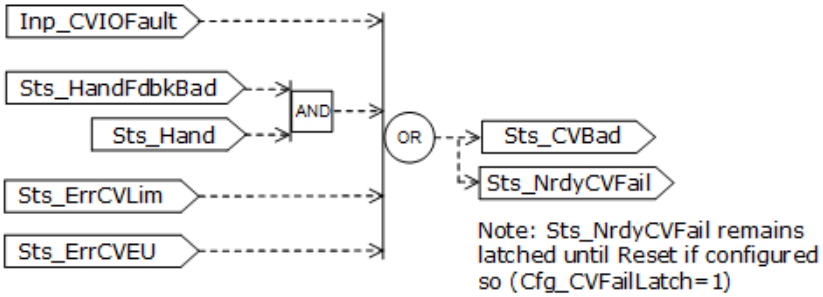


CV bad status

CV is bad if:

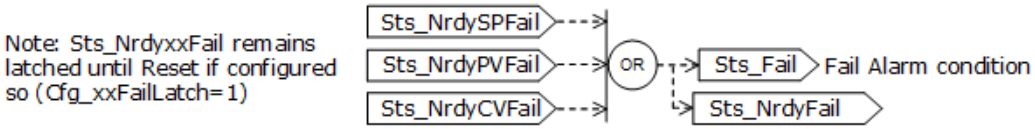
- Inp_CVIOFault=1, or
- CV clamping limits are invalid (Sts_ErrCVLim=1), or
- CV scaling limits are invalid (Sts_ErrCVEU=1),or

- Hand feedback is bad and the PPID is in Hand mode.



Loop failure status

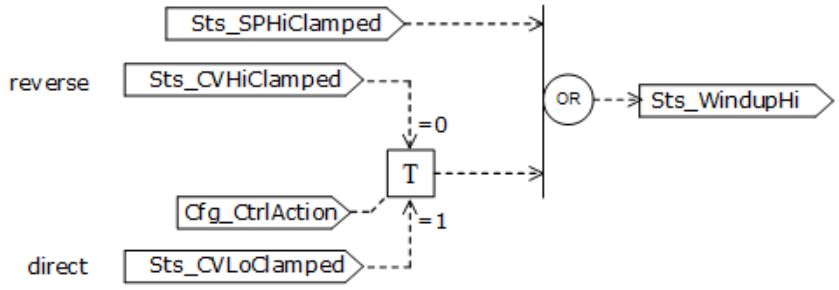
Loop failure (Sts_Fail) is set if SP, PV, or CV fails and failure shed is latched. If Sts_Fail is set the loop cannot operate normally.



Windup status

Windup status High (Sts_WindupHi) is set if:

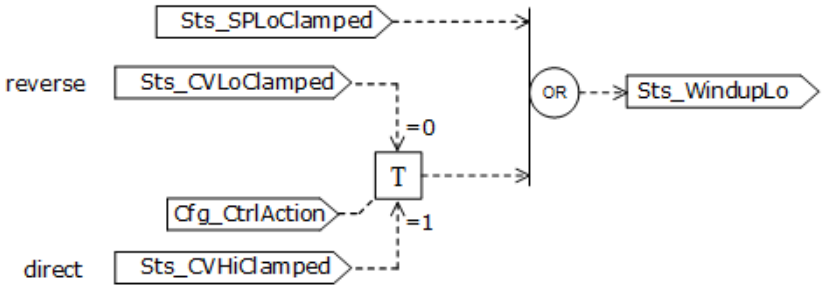
- Selected SP is clamped at high limit (Sts_SPHiClamped=1), or
- CV is clamped at high limit (Sts_CVHiClamped=1) if the PPID is configured for reverse control action (Cfg_CtrlAction=0), or
- CV is clamped at low limit (Sts_CVLoClamped=1) if the PPID is configured for direct control action (Cfg_CtrlAction=1).



Windup status Low (Sts_WindupLo) is set if:

- Selected SP is clamped at low limit (Sts_SPLoClamped=1), or
- CV is clamped at low limit (Sts_CVLoClamped=1) if the PPID is configured for reverse control action (Cfg_CtrlAction=0), or

- CV is clamped at high limit (Sts_CVHiClamped=1) if the PPID is configured for direct control action (Cfg_CtrlAction=1).



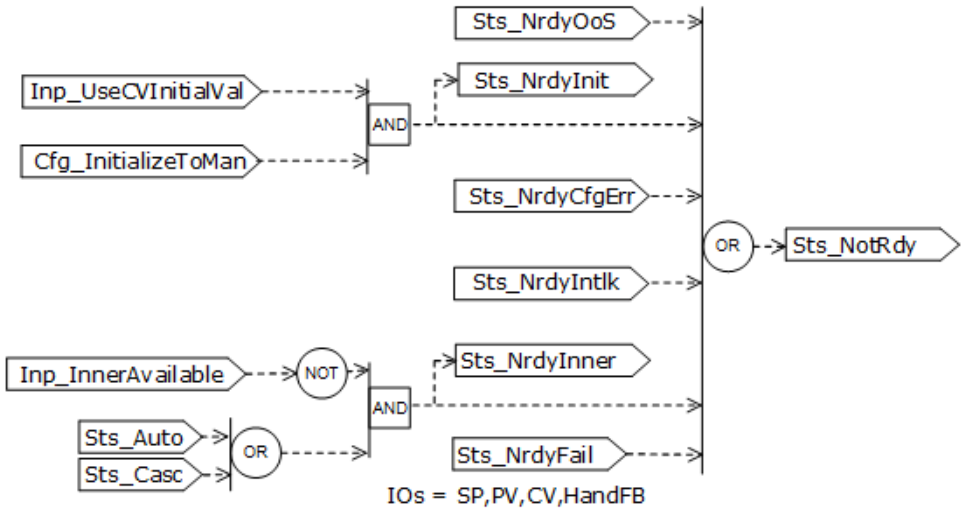
Active status

Status CV active (Sts_Active) is set if Val_CVOut is greater than Cfg_MaxInactiveCV. The HMI graphic symbol, for example, valve open, is shown for Sts_Active=1, and the inactive HMI graphic symbol, for example, valve closed, is shown when Sts_Active=0.

Not ready status

The PPID instruction is not ready (Sts_NotRdy=1) under these conditions:

- PPID is out of service (Sts_NrdyOoS=1), or
- PPID goes to Manual loop mode when initialized and Inp_CVInitialVal is used to initialize CV (Sts_NrdyInit=1), or
- PPID configuration error (Sts_NRdyCfgErr=1), or Interlock not OK (Sts_NrdyIntlk=1), or
- Inner object to this PPID, typically PID of the secondary loop, is not available (Inp_InnerAvailable=0) for Auto or Cascade mode of the PPID (Sts_Auto=1 | Sts_Casc=1), or
- Loop fails (Sts_Fail=1) because of PV/SP/CV bad value or communication failure, or Hand feedback value is invalid.



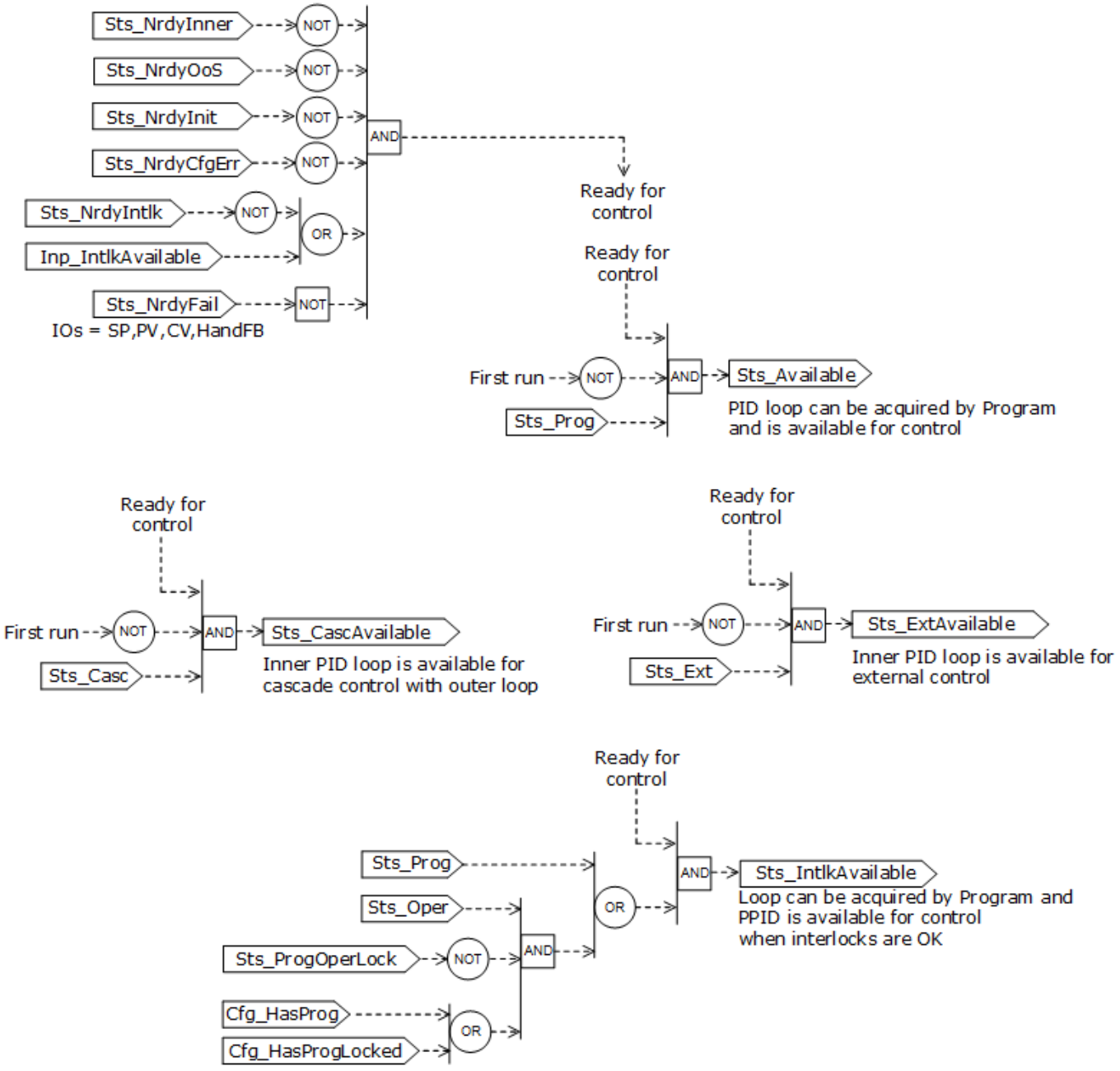
Available statuses

The PPID instruction is available (Sts_Available=1) to other objects when in Program mode, not in the first scan, and generally ready for control.

Inner loop is available for cascade control (Sts_CascAvailable=1) when in Cascade mode, not in the first scan, and generally ready for control.

Inner loop is available for external control (Sts_ExtAvailable=1) when in External mode, not in the first scan, and generally ready for control.

This diagram shows the conditions required for the instruction to be generally ready for control.



Configuration of strings for HMI

Configure strings for HMI faceplates, as seen in FactoryTalk View, and for the Logix Designer configuration dialog box. The strings are set to extended properties of tag items. Configure strings in the Logix Designer application only.

- Description
- Label for graphic symbol
- Display Library for HMI Faceplate call-up
- Instruction name
- Area name
- URL link
- Path to an object with more information
- Path to an object providing Cascade SP
- Path to an object providing PV
- Path to an object consuming CV
- PV/SP engineering units
- CV engineering units

Command source

The instruction uses these command sources. The command sources are prioritized in order from highest to lowest in this table.

Command source	Description
Hand	Hardwired logic or other logic outside the instruction owns control of the device. The instruction tracks the state of the device for bumpless transfer back to one of the other command sources. While in Hand mode the PID algorithm does not compute the change in CV. Val_CVOut = Inp_HandFdbk (Inp_Tieback), regardless of the control mode. Hand mode is typically used to indicate that control of the final control element was taken over by a field hand/auto station. Set Inp_Hand to request hand mode. This value is usually read as a digital input from a hand/auto station. (Highest Priority command source)
Out-of-Service	The instruction is disabled and has no owner.
Maintenance	Maintenance owns control of the device and supersedes Operator, Program, External and Override control. Operator commands and settings from the HMI are accepted. Bypassable interlocks are bypassed, and device timeout checks are not processed.
Override	Priority logic owns control of the device and supersedes Operator, Program and External control. Override Input (Inp_OvrdCmd) is accepted. If so configured, bypassable interlocks are bypassed.
External	External logic owns control of the device. External commands (XCmd_) are accepted.
Program locked	Program logic owns control of the device. Program commands (PCmd_) are accepted. Operator cannot take from Program. Override cannot take from Program unless Cfg_OvrdOverLock = 1.
Program	Program logic owns control of the device. Program commands (PCmd_) are accepted.
Operator locked	The Operator owns control of the device. Operator commands (OCmd_) from the HMI are accepted. Program cannot take from Operator. Override cannot take from Operator unless Cfg_OvrdOverLock = 1.
Operator	The Operator owns control of the device. Operator commands (OCmd_) from the HMI are accepted. (Lowest Priority command source)

The instruction enables or disables these operations:

- Maintenance Out-of-Service exists
- Maintenance exists
- External exists
- Program (unlocked) exists
- Program locked exists
- Operator (unlocked) exists
- Operator locked exists
- PCmd_Lock used as a Level (1 = Lock, 0 = Unlock)
- XCmd_Acq used as a Level (1 = Acquire External, 0 = Release External)

The instruction checks for invalid configurations of control and forces the nearest valid configuration.

The core control model arbitrates the source of the commands and parameters that the receiving function accepts. The core control model determines if the source is:

- A programmatic entity which resides entirely within the processing environment, or
- An external interface entity which issues commands and parameters external and asynchronously to the processing environment.

Locking a control source prevents the other control source from acquiring privilege.

Core command source model

The core command source model consists of these control sources:

- Oper
- OperLocked
- Prog
- ProgLocked

The control model defaults to this configuration. Other control sources may be present in the model but act as overriding control sources, acting independent of the base Operator/Program state machine.

Enabling control sources in configuration

The individual control sources may be enabled or disabled by the user. The default configuration uses the entire base model; upon power-up of the processing environment the control source will be the designated default. Some combinations of enabled control sources are disallowed as they are either unnecessary or could create unintended changes.

Ownership default and priority

Configuration allows a user to specify whether Operator or Program will be the power-up default and specify whether Operator or Program commands will win when simultaneously asserted.

Automatic reset of commands

All commands are treated as one-shot-latched. This means that all commands are automatically cleared when the instruction executes and processes them.

Changing destination states

Under certain configurations the destination command source for some commands may change. This is in keeping with the intent of the command. For example, if the Prog is disabled, the destination of the OCmd_Prog command is directed to the ProgLocked state instead of the Prog state. This maintains the intent of the OCmd_Prog command: the operator entity wishes to place the function in control of the program. If the command was eliminated then there would be no way to accomplish this. This is only done in configurations where it would cause no conflict or race condition, but serves to preserve as much user functionality as is practical.

Higher priority command sources

These higher priority command sources operate independently within the model: External, Override, Maintenance, Out-of-Service, In-Service, and Hand.

Command source processing and ownership arbitration

Maintenance and operator commands (MCmd_OoS, MCmd_IS, MCmd_Acq, MCmd_Rel, OCmd_Oper, OCmd_Prog, Ocmd_Lock, OCmd_Unlock, OCmd_Normal) are forwarded to the contained PCmdSrc instruction.

Maintenance and ready bits (MRdy_OoS, MRdy_IS, MRdy_Acq, MRdy_Rel, ORdy_Oper, ORdy_Prog, ORdy_Lock, ORdy_Unlock, ORdy_Normal) are copied from the contained PCmdSrc instruction in response.

The instruction sets ownership (Val_Owner) according to the order in which requests are received. If the requestor supplies a non-zero owner ID (PSet_Owner) and the current owner is none (Val_Owner = 0), the instruction assigns ownership to the requesting ID.

Monitor the PPID Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

The PID algorithm is only scanned at the configured execution rate Cfg_ExecTime. Configuring the instruction for execution period = 0.0 (default) or for a period shorter than the instruction scan time has no effect and the PID algorithm executes every scan. For the real execution period check Val_ExecTime.

Ladder Diagram

Condition/State	Action Taken
Prescan	Rung-condition-out is cleared to false. Sts_eSrc is set to 0. Sts_bSrc is set to 0.
Instruction first run	All commands that are automatically cleared each execution are cleared and ignored. The Program/Operator selection is set based on the configuration (Cfg_ProgPwrUp). The Program or Operator lock selection is set to unlocked. Loop mode is set according to configuration (Cfg_PwrUpLM). Initial value pf SP and CV are set according to Cfg_SPPwrUp and Cfg_CVPwrUp and selected loop mode. If Cfg_PwrUpLM=0 (no change), SP and CV initial values are equal to values from last scan (e.g. before power down). PSet_Owner and Sts_Owner are set to 0.
Rung-condition-in is false	The instruction is put Out of Service if Inp_Hand=0. The output is de-energized. All alarms are cleared. Command source selection processing proceeds as normal except that all ownership status bits (Sts_Maint, Sts_Ovrd, Sts_Ext, Sts_Prog and Sts_Oper) are cleared to 0. When rung-condition-in becomes true, the instruction takes into account the commands received and sets the active command source accordingly.
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Instruction first run	See Instruction first run in the Ladder Diagram table.
Instruction first scan	See Instruction first run in the Function Block Diagram table.
EnableIn is false	See Rung-condition-in is false in the Ladder Diagram table.
EnableIn is true	See Rung-condition-in is true in the Ladder Diagram table.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

In Structured Text, EnableIn is always true during normal scan. The instruction executes when it is in the control path activated by the logic.

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Instruction first run	See Instruction first run in the Ladder Diagram table.
EnableIn is true	See Rung-condition-in is true in the Ladder Diagram table.
Postscan	EnableIn and EnableOut bits are cleared to false.

Examples

Example 1: PID feedback control

This example demonstrates wiring analog input and analog output instructions with a PPID instruction.

Pressure system tank level control is considered as an example of wiring analog input and analog output instruction(s) with a PPID instruction. Consider a section of water distribution system with a station pumping water from a collector tank to a tank maintaining system pressure at distribution nodes. Due to demand variation water level fluctuates so the system pressure varies. The PPID instruction helps stabilize the pressure by measuring and processing the water level as a process variable. PPID calculates the reference speed for a motor driving a pump to compensate for demand variations. The actual motor rpm is measured and fed back to the controller. This feedback enables the control scheme to read the actual motor speed when the service personnel takes over the control and manipulates with pump directly from the control panel (motor is in Hand). Components of the PID loop should track the actual rpm and be ready to take over control without any bump.

Analog input module provides level in raw units and fault signals processed by the PAI instruction (LI_30). Level in engineering units (LI_30.Val), level signal quality (LI_30.SrcQ) and alarm notification (LI_30.Sts_eNotifyAll) are wired to corresponding inputs of a PPID (LIC_31.Inp_PV, LIC_31.Inp_PVSrcQ, LIC_31.Inp_PVNotify). PPID is configured so the proper response occurs when the level signal is not reliable and should not enter PID formula for CV calculation. Set Cfg_PVFailTrigger to properly classify PV source quality. Use Cfg_PVFailSPAction, Cfg_PVFailCVAction, and Cfg_PVFailLMAction to specify the setpoint value, the value of the control variable, and the loop mode the PPID instruction uses if the PV source quality is reported as bad.

The PPID instruction should receive a signal indicating whether the downstream object driven by the PPID (the analog output instruction SZ_31 driving a motor) is active and is responding properly to the PPID instruction CV. In other words, the PPID instruction should receive a signal if the control loop is not open.

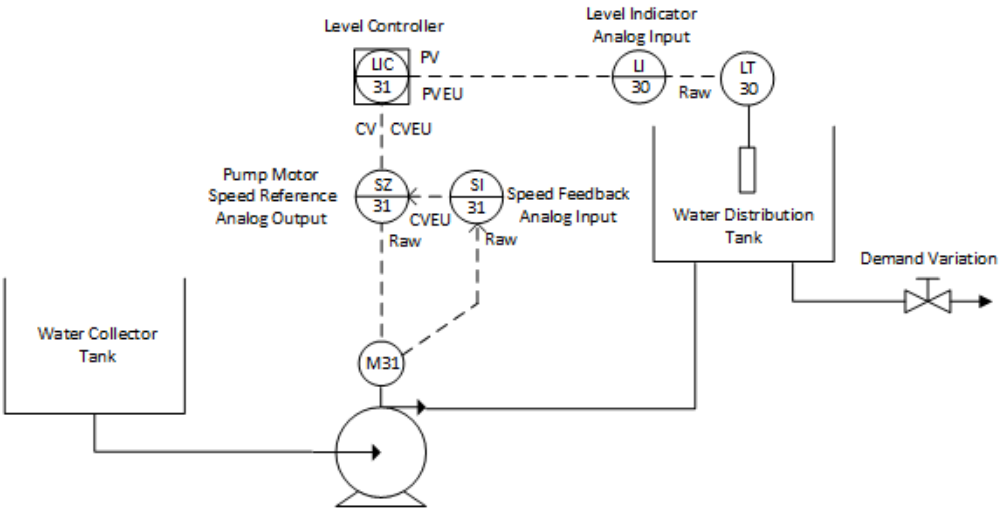
If the CV output of SZ_31 saturates (SZ_31.Sts_WindupHi=1 or SZ_31.Sts_WindupLo=1), the PPID instruction is out of control. To prevent windup, SZ_31.Sts_WindupHi and SZ_31.Sts_WindupLo are wired to LIC_31.Inp_WindupHi and LIC_31.Inp_WindupLo.

If the pump motor, the device driven by SZ_31, is in failure (M31_Fail=1), the loop is also open and both SZ_31 and LIC_31 should follow the actual device status to avoid any bump when put back in control. Actual motor rpm is measured, the rpm raw signal is scaled to engineering units by analog input instruction SI_31, and scaled value of rpm (SI_31.Val) is wired as a feedback signal to SZ_31.Inp_PosFdbk.

If service personnel have the pump motor in hand mode (M31_Hand=1), actual speed is provided separately in the M31_Fdbk tag, wired to SZ_31.Inp_HandFdbk.

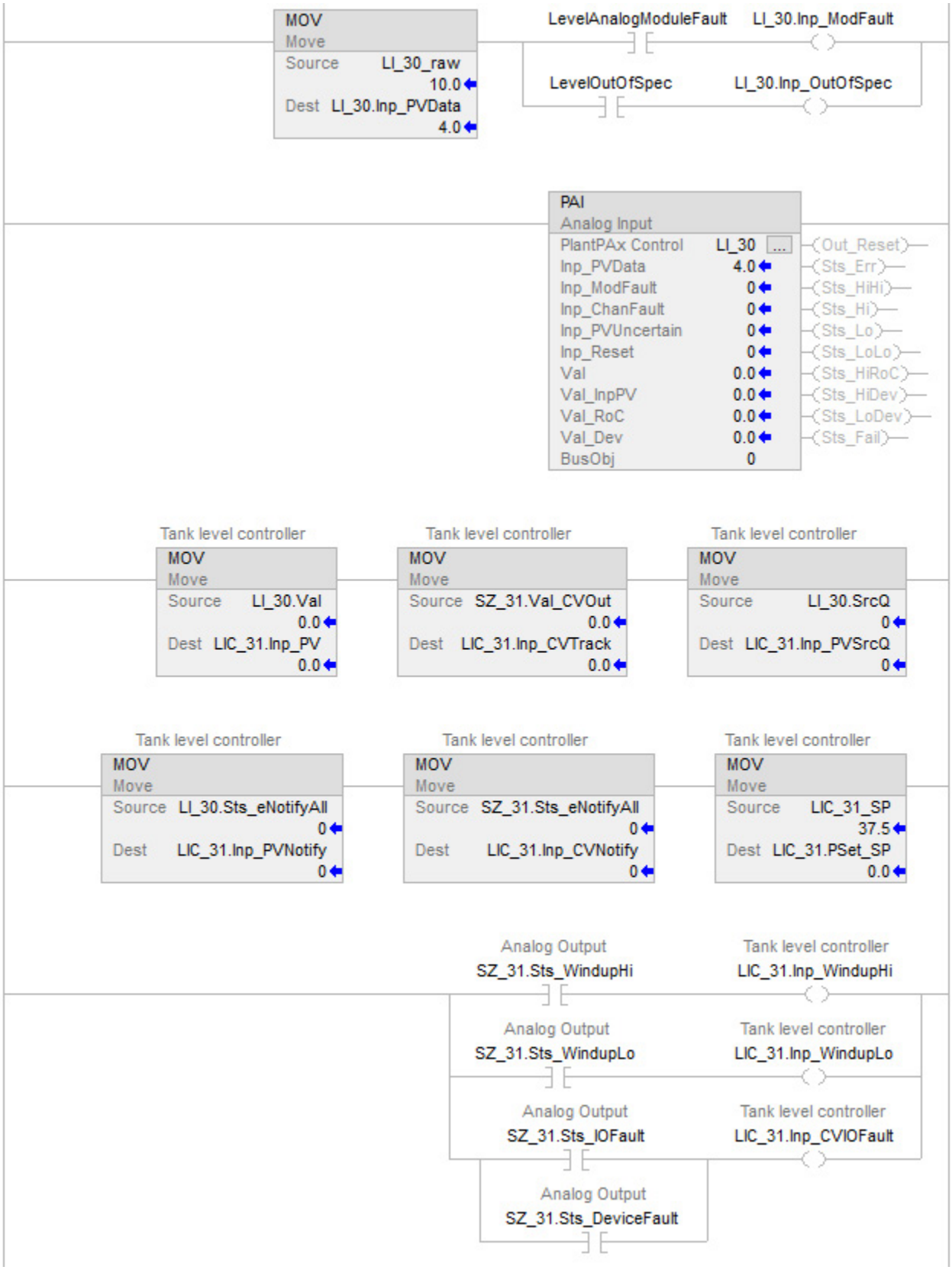
Device status is collected in analog output instruction SZ_31 and communicated upstream to the PPID instruction. The PPID instruction should be informed that the control loop is open somewhere downstream and thus the SZ_31 is not available for LIC_31. Wire SZ_31.Sts_Available to LIC_31.Inp_InnerAvailable to pass the information to the PPID. Actual speed reference should be provided to PPID for tracking purposes when control loop opens. Wire SZ_31.Val_CVOut to LIC_31.Inp_CVTrack. Other downstream problems like communication or device fault should also be fed back to the PPID so the instruction takes a configured action in response. Merge SZ_31.Sts_IOFault with SZ_31.Sts_DeviceFault, wire the result to LIC_31.Inp_CVIOFault and use LIC_31.Cfg_CVFailSPAction and LIC_31.Cfg_CVFailCVAction to configure the PPID for the follow up action.

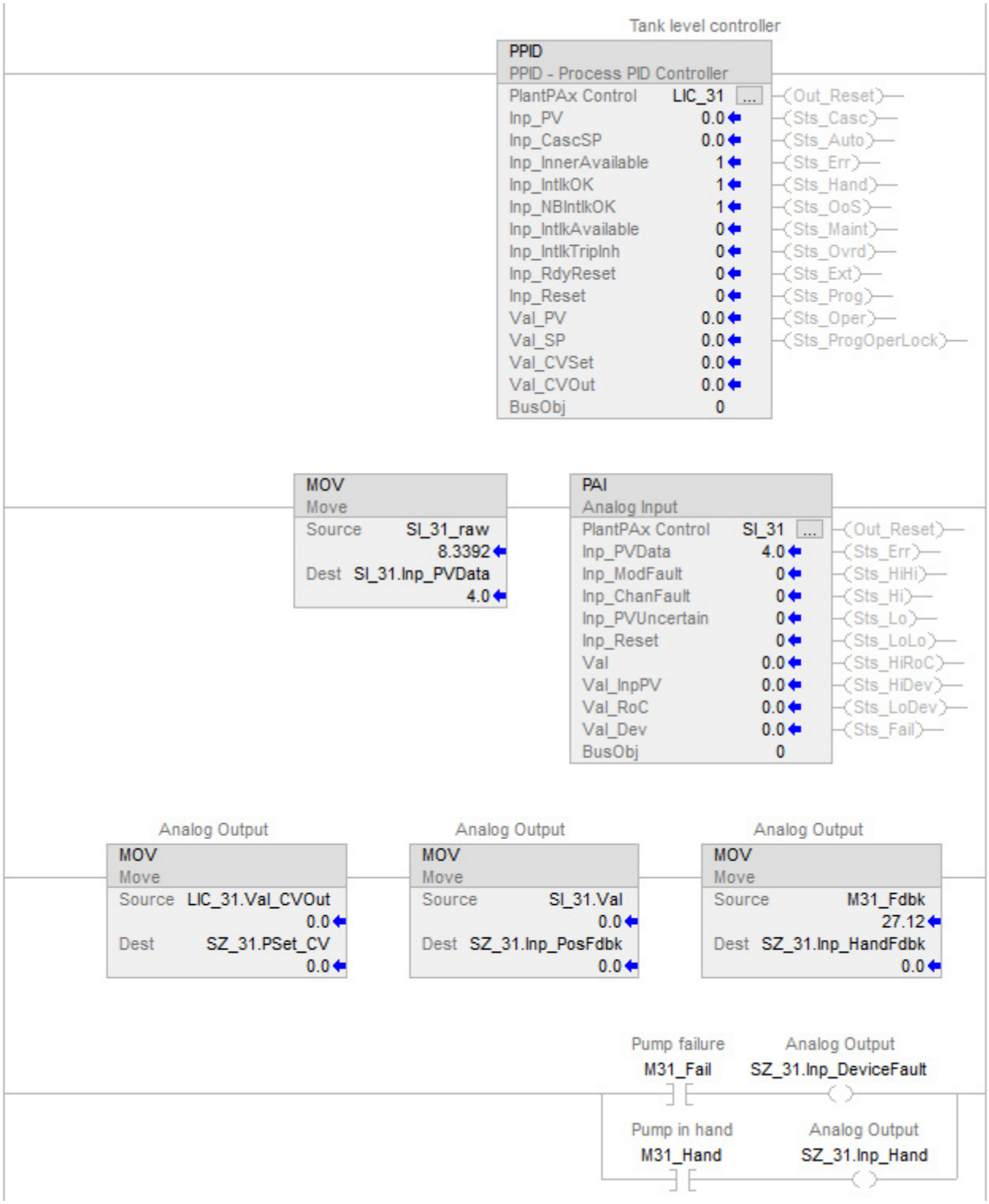
Finally update the alarm notification for the PPID and propagate the highest severity notification from SZ_31 upstream. Wire SZ_31.Sts_eNotifyAll to LIC_31.Inp_CVNotify.

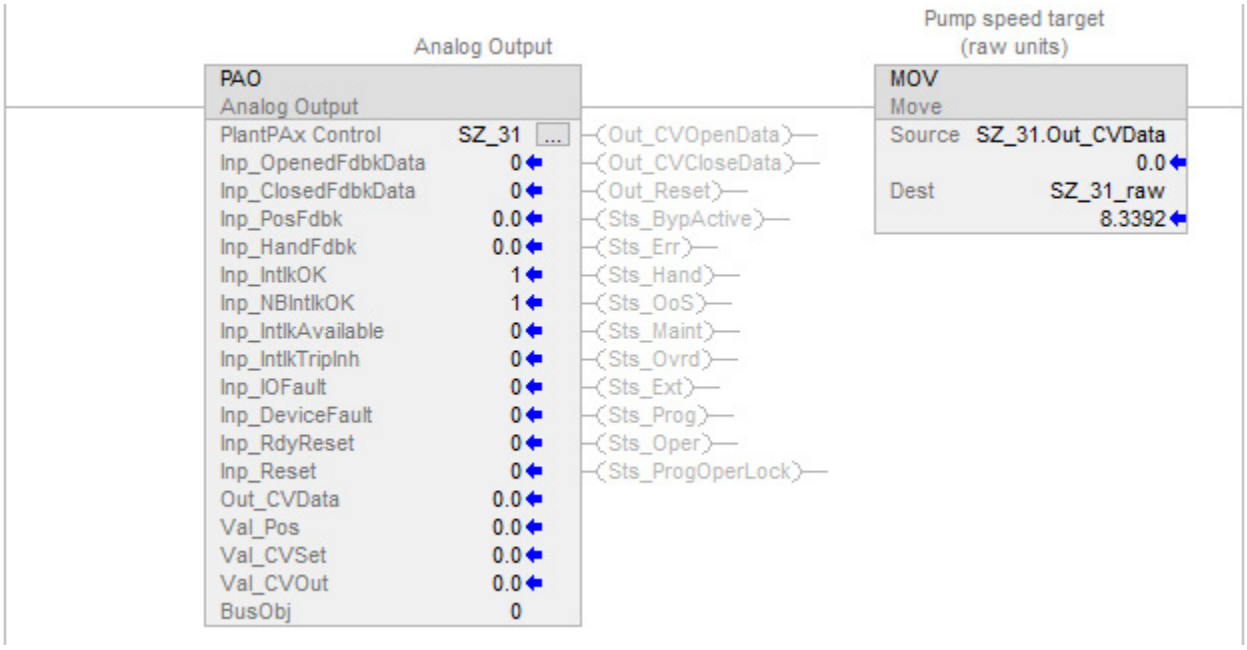


The example is shown in all three language editors.

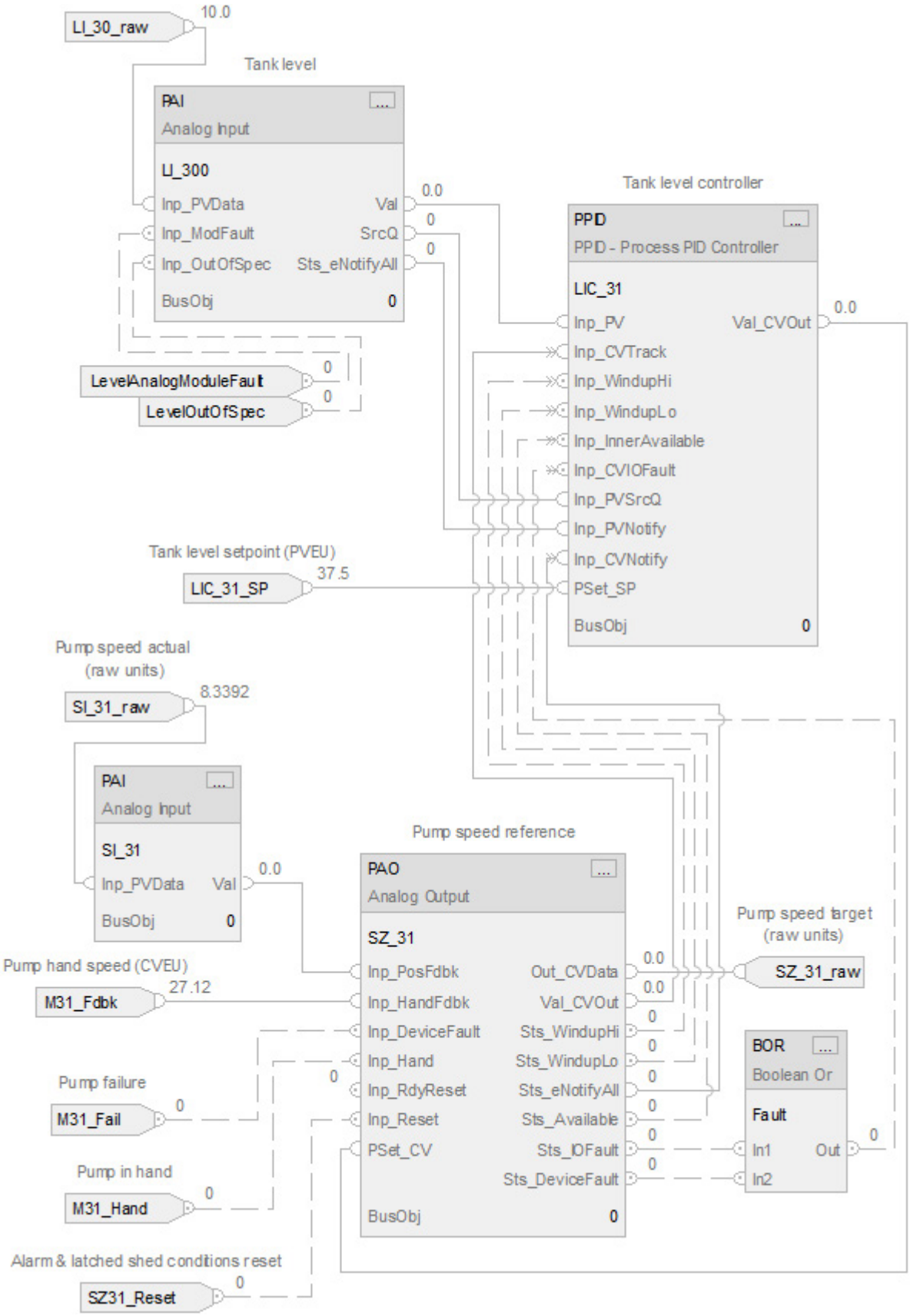
Ladder Diagram




Tank level controller
PPID
PPID - Process PID Controller
PlantPAx Control LIC_31
Inp_PV 0.0
Inp_CascSP 0.0
Inp_InnerAvailable 1
Inp_IntlkOK 1
Inp_NBIntlkOK 1
Inp_IntlkAvailable 0
Inp_IntlkTripInh 0
Inp_RdyReset 0
Inp_Reset 0
Val_PV 0.0
Val_SP 0.0
Val_CVSet 0.0
Val_CVOut 0.0
BusObj 0
Out_Reset
Sts_Casc
Sts_Auto
Sts_Err
Sts_Hand
Sts_OoS
Sts_Maint
Sts_Ovrd
Sts_Ext
Sts_Prog
Sts_Oper
Sts_ProgOperLock
MOV
Move
Source SI_31_raw
8.3392
Dest SI_31.Inp_PVData
4.0
PAI
Analog Input
PlantPAx Control SI_31
Inp_PVData 4.0
Inp_ModFault 0
Inp_ChanFault 0
Inp_PVUncertain 0
Inp_Reset 0
Val 0.0
Val_InpPV 0.0
Val_RoC 0.0
Val_Dev 0.0
BusObj 0
Out_Reset
Sts_Err
Sts_HiHi
Sts_Hi
Sts_Lo
Sts_LoLo
Sts_HiRoC
Sts_HiDev
Sts_LoDev
Sts_Fail
Analog Output
MOV
Move
Source LIC_31.Val_CVOut
0.0
Dest SZ_31.PSet_CV
0.0
Analog Output
MOV
Move
Source SI_31.Val
0.0
Dest SZ_31.Inp_PosFdbk
0.0
Analog Output
MOV
Move
Source M31_Fdbk
27.12
Dest SZ_31.Inp_HandFdbk
0.0
Pump failure
M31_Fail
Analog Output
SZ_31.Inp_DeviceFault
Pump in hand
M31_Hand
Analog Output
SZ_31.Inp_Hand


Analog Output
PAO
Analog Output
PlantPAx Control SZ_31 ...
Inp_OpenedFdbkData 0
Inp_ClosedFdbkData 0
Inp_PosFdbk 0.0
Inp_HandFdbk 0.0
Inp_IntlkOK 1
Inp_NBIntlkOK 1
Inp_IntlkAvailable 0
Inp_IntlkTripInh 0
Inp_IOFault 0
Inp_DeviceFault 0
Inp_RdyReset 0
Inp_Reset 0
Out_CVData 0.0
Val_Pos 0.0
Val_CVSet 0.0
Val_CVOut 0.0
BusObj 0
Out_CVOpenData
Out_CVCloseData
Out_Reset
Sts_BypActive
Sts_Err
Sts_Hand
Sts_OoS
Sts_Maint
Sts_Ovrd
Sts_Ext
Sts_Prog
Sts_Oper
Sts_ProgOperLock
Pump speed target
(raw units)
MOV
Move
Source SZ_31.Out_CVData
0.0
Dest SZ_31_raw
8.3392

Function Block Diagram



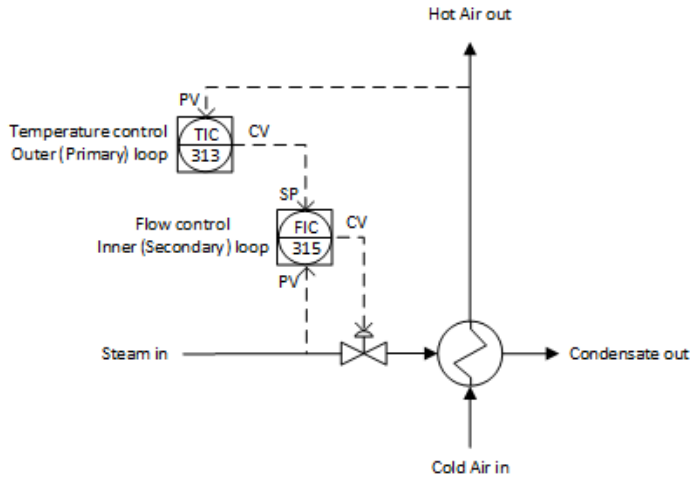
Structured Text

```
LI_30.Inp_PVData := LI_30_raw;
LI_30.Inp_ModFault := LevelAnalogModuleFault;
LI_30.Inp_OutOfSpec := LevelOutOfSpec;
PAI(LI_30,0);
LIC_31.Inp_PV := LI_30.Val;
LIC_31.Inp_CVTrack := SZ_31.Val_CVOut;
LIC_31.Inp_PVSrcQ := LIC_31.Inp_PVSrcQ;
LIC_31.Inp_PVNotify := LI_30.Sts_eNotifyAll;
LIC_31.Inp_CVNotify := SZ_31.Sts_eNotifyAll;
LIC_31.PSet_SP := LIC_31_SP;
LIC_31.Inp_WindupHi := SZ_31.Sts_WindupHi;
LIC_31.Inp_WindupLo := SZ_31.Sts_WindupLo;
LIC_31.Inp_CVIOFault := SZ_31.Sts_IOFault OR SZ_31.Sts_DeviceFault;
PPID(LIC_31,0);
SI_31.Inp_PVData := SI_31_raw;
PAI(SI_31,0);
SZ_31.PSet_CV := LIC_31.Val_CVOut;
SZ_31.Inp_PosFdbk := SI_31.Val;
SZ_31.Inp_HandFdbk := M31_Fdbk;
SZ_31.Inp_DeviceFault := M31_Fail;
SZ_31.Inp_Hand := M31_Hand;
PAO(SZ_31,0);
SZ_31_raw := SZ_31.Out_CVData;
```

Example 2: Cascade control

Cascade control is useful when externally-caused upsets to the controlled variable occur often, which then cause upsets to the process variable you are trying to control. For example, try to control the temperature of liquid in a tank by varying the amount of steam fed into a heating jacket around the tank. If the steam flow suddenly drops because of an upstream process, the temperature of the liquid in the tank eventually drops and the PPID

instruction then opens the steam valve to compensate for the drop in temperature.



In this example, a cascaded loop provides better control by opening the steam valve when the steam flow drops before the liquid temperature in the tank drops. To implement a cascaded loop, use a PPID instruction to control the steam valve opening based on a process variable signal from a steam flow transmitter. This is the inner loop of the cascaded pair. A second PPID instruction (called the outer or primary loop) uses the liquid temperature as a process variable and sends its CV output into the setpoint of the inner loop. In this manner, the outer temperature loop asks for a certain amount of steam flow from the inner steam flow loop. The steam flow loop is then responsible for providing the amount of steam requested by the temperature loop in order to maintain a constant liquid temperature.

For a cascaded pair of loops to work correctly, the inner loop must have a faster process response than the primary loop. This is because the inner loop's process must be able to compensate for any upsets before these upsets affect the outer loop's process. In this example, if steam flow drops, the steam flow must be able to increase as a result of the inner controller's action before the liquid temperature is affected.

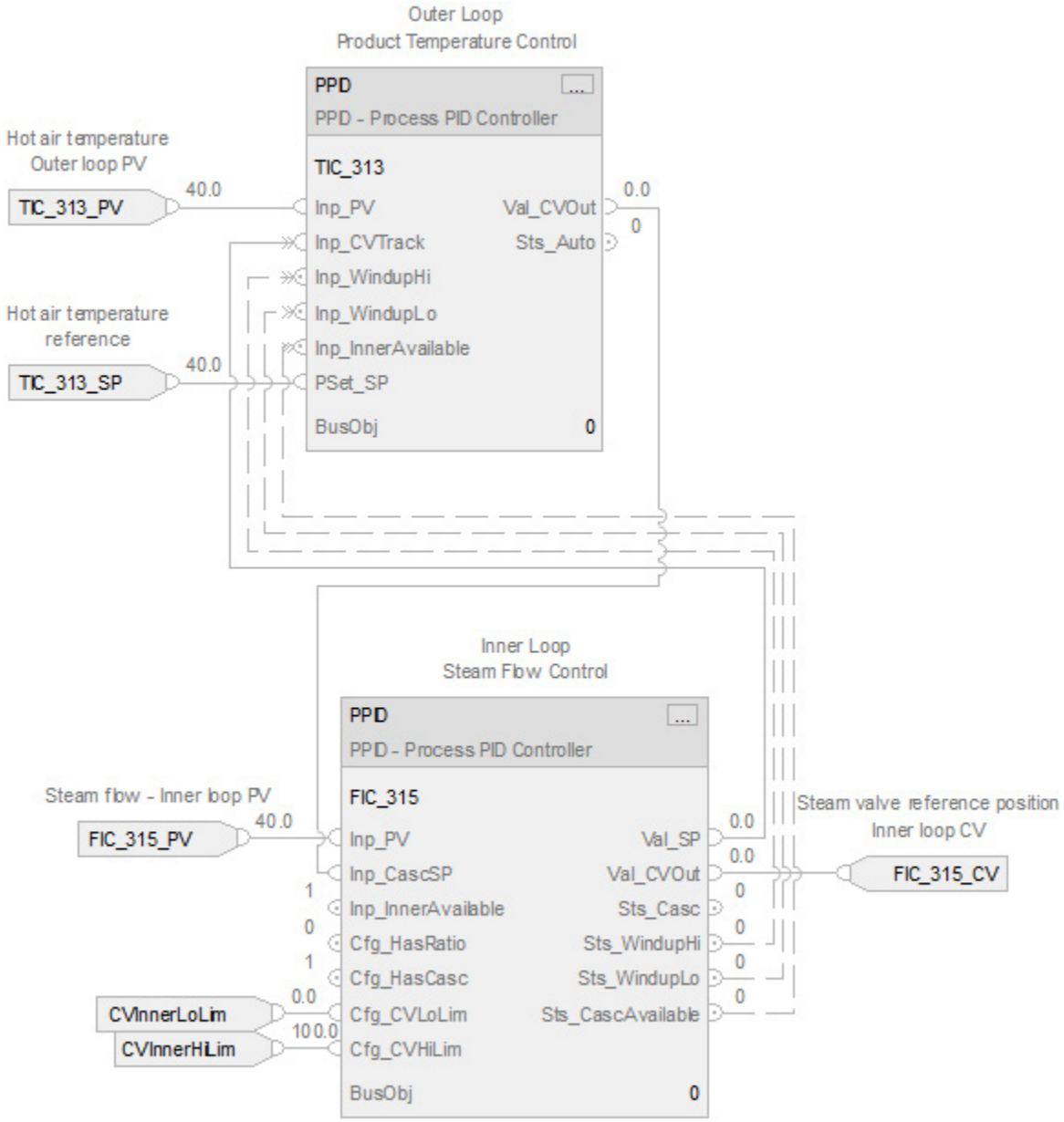
To set up a pair of cascaded PPID instructions, set the Cfg_HasCasc input parameter in the inner loop. This allows the inner loop to be placed into Cascade/Ratio mode. Next, wire the Val_CVOut from the outer loop into the Inp_CascSP parameter on the inner loop. The Inp_CascSP value is used as the SP on the inner loop when the inner loop is placed into Cascade/Ratio mode. The engineering unit range of the Val_CVOut on the outer loop should match the engineering unit range of the PV on the inner loop. This lets the outer loop scale its 0-100% value of CV into the matching engineering units used for the setpoint on the inner loop.

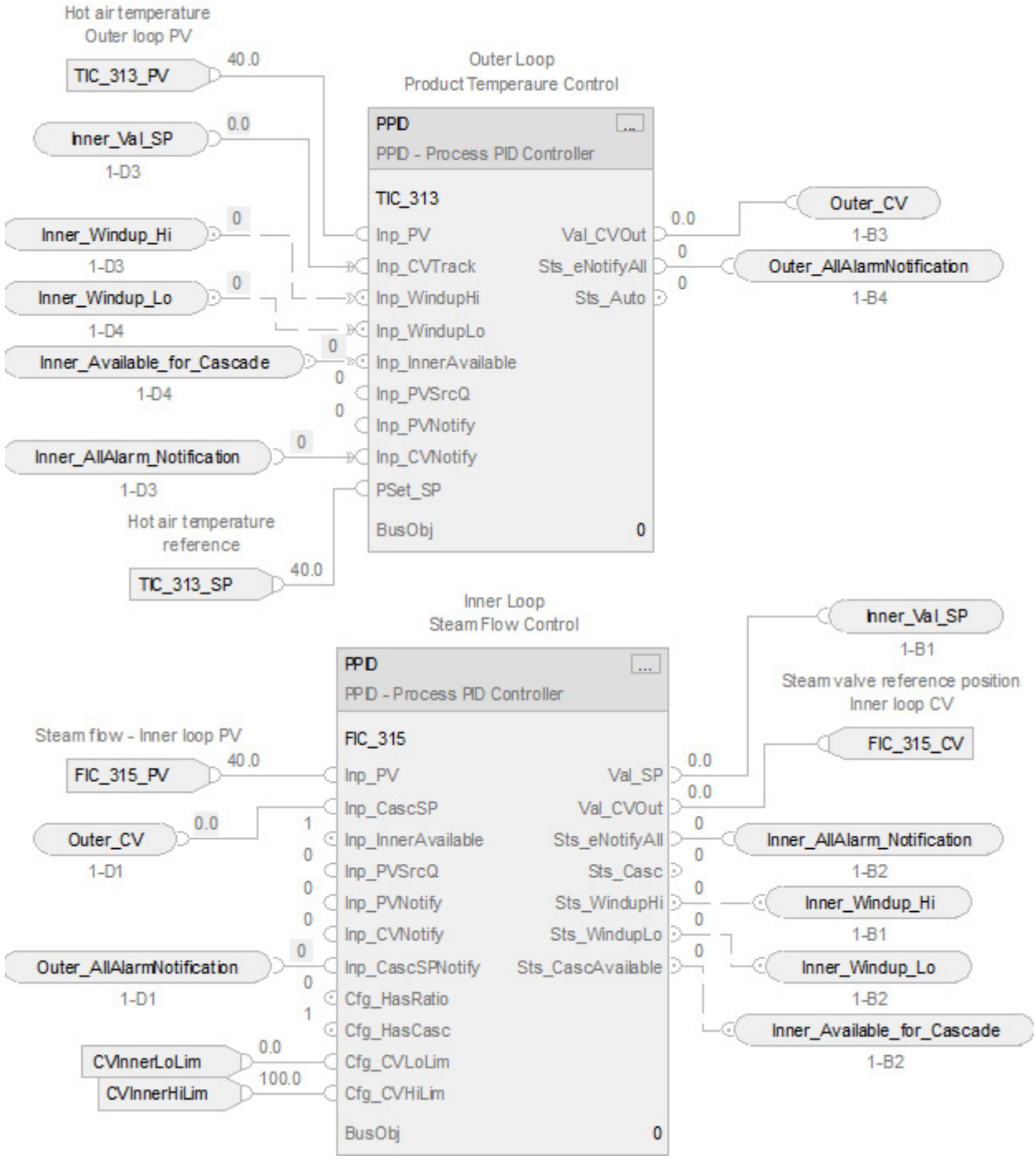
The PPID instruction has several other features to more effectively support cascade control. Wire the Sts_CascAvailable output on the inner loop PPID into the Inp_InnerAvailable input on the outer loop PPID and wire the Val_SP

output of the inner into the Inp_CVTrack input on the outer. This sets the Val_CVOut value of the outer loop to track the SP of the inner loop when the inner loop is not in Cascade/Ratio mode. This allows a bumpless transfer when you place the inner loop back into Cascade/Ratio mode. Also, wire the Sts_WindupHi and Sts_WindupLo outputs on the inner loop into the Inp_WindupHi and Inp_WindupLo inputs on the outer loop. This causes the outer loop to stop increasing or decreasing, as appropriate, its Val_CVOut value if the inner loop hits a SP limit or CV limit and eliminates any windup on the outer loop if these conditions occur.

The example is shown in FBD in two versions, with minimum wiring and extended wiring. The extended wiring shows passing alarm notification between outer and inner PID and makes signal quality input pins visible for immediate use in the application.

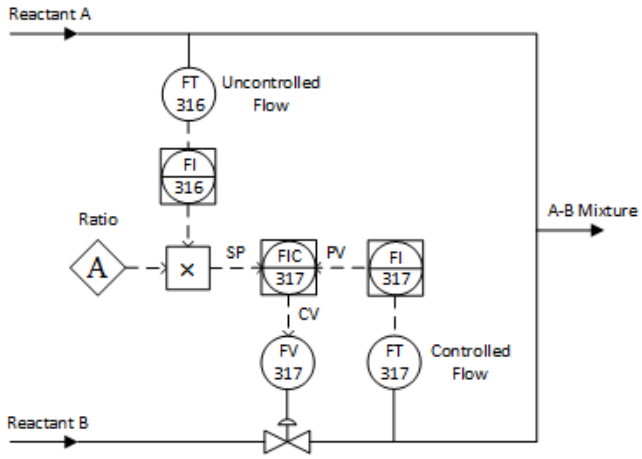
Function Block Diagram





Example 3: Ratio control

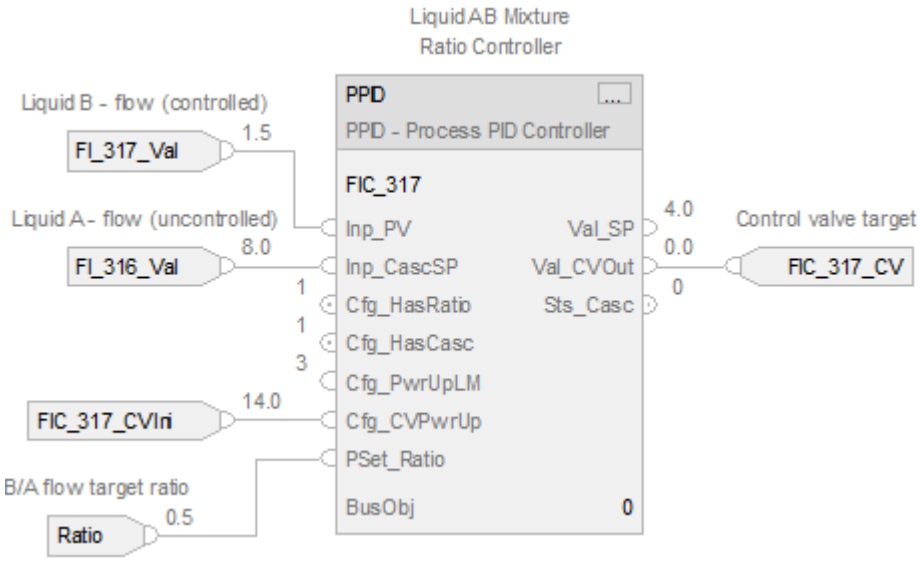
Ratio control is typically used to add a fluid in a set proportion to another fluid. For example, if you want to add two reactants (say A and B) to a tank in a constant ratio, and the flow rate of reactant A may change over time because of some upstream process upsets, you can use a ratio controller to automatically adjust the rate of reactant B addition. In this example, reactant A is often called the uncontrolled or wild flow since it is not controlled by the PPID instruction. The flow of reactant B is then called the controlled flow.



To perform ratio control with a PPID instruction, set the Cfg_HasCasc and Cfg_HasRatio input parameters. Wire the uncontrolled flow into the Inp_CascSP input parameter. When in Cascade/Ratio mode, the uncontrolled flow is multiplied by either the OSet_Ratio, when in Operator control, or the PSet_Ratio, when in Program control, and the resulting value is used by the PPID instruction as the setpoint.

The example is shown in FBD.

Function Block Diagram



Example 4: Feedforward control

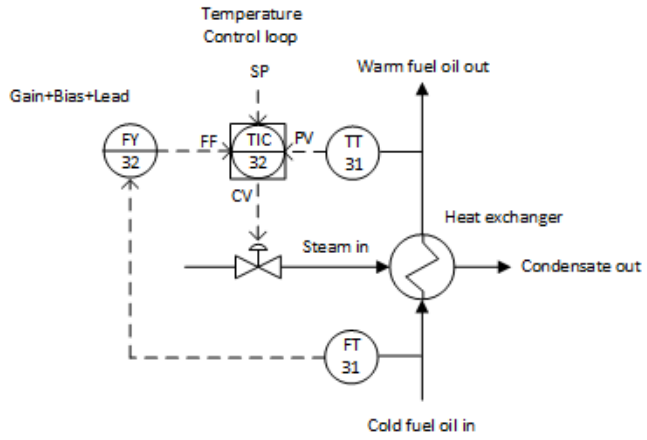
Feedforward control is a disturbance rejection strategy to deal with load change. Rather than rely on feedback to make corrective changes to a process only after some load change has driven the process variable away from setpoint, control schemes with feedforward monitor the relevant load(s) and

use that information to preemptively make stabilizing changes to the final control element such that the process variable will not be affected.

Consider a control system manipulating steam flow to the heat exchanger to maintain the discharge temperature of the oil at a constant setpoint value. The outlet temperature will suffer temporary deviations from setpoint if load conditions change. The feedback control system may be able to eventually bring the exiting oil's temperature back to setpoint, but it cannot begin corrective action until after a load has driven the oil temperature off setpoint. To improve control, build feedforward action and feedback action into the design. The feedforward action allows the control system to take corrective action in response to load changes before the process variable is affected.

In this example, the dominant load in the system is oil flow rate, caused by changes in demand at the combustion furnace where the oil is being used as fuel. Adapting this control system to include feedforward requires installing an oil flow transmitter and a gain/bias function providing feedforward action to the PID controller maintaining temperature. With feedforward control action in place, the steam flow rate immediately changes with oil flow rate, preemptively compensating for the increased or decreased heat demand of the oil. The time constant of the process with regard to steam flow changes is greater than the time constant of the process with regard to oil flow changes.

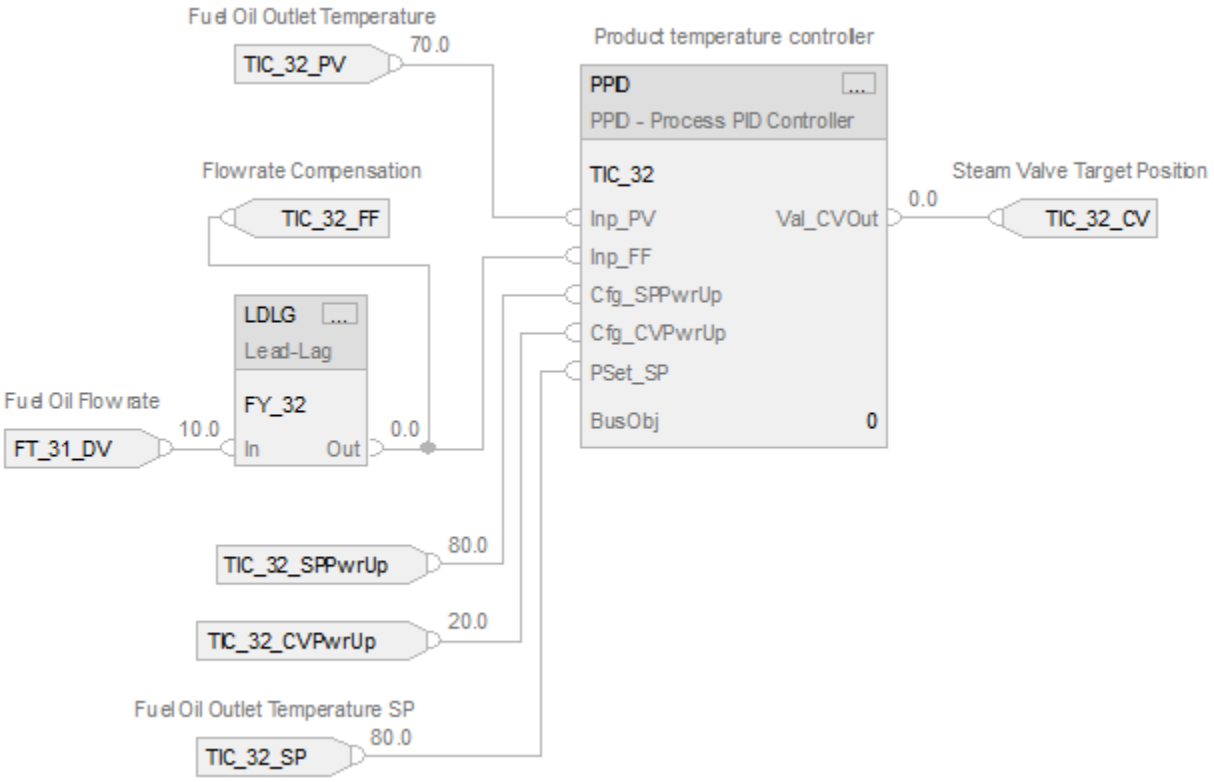
Oil flow is a wild variable. The feedforward control system can only manipulate the steam valve position in response to oil flow. The best method to help control it is to speed up the time constant of the steam flow variable, which the system can influence. The solution is to wire the output of the user-specified lead function FY_32 to Inp_FF of the PPID.



Now, when the oil flow rate to this heat exchanger suddenly increases, the lead function will add a surge to the feedforward signal, quickly opening the steam valve and sending a surge of steam to the exchanger to help overcome the naturally sluggish response of the oil temperature to changes in steam flow. The feedforward action won't be perfect with this lead function added, but it will be substantially better than if there was no dynamic compensation added to the feedforward signal.

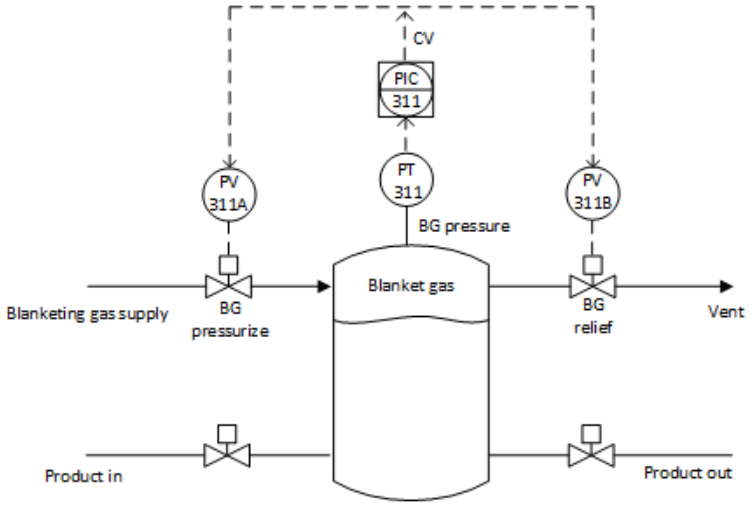
The example is shown in FBD.

Function Block Diagram



Example 5: Split-range control

Split-range control allows using a single PID Control Variable to drive more than one final control element. An application example is shown in the following figure.



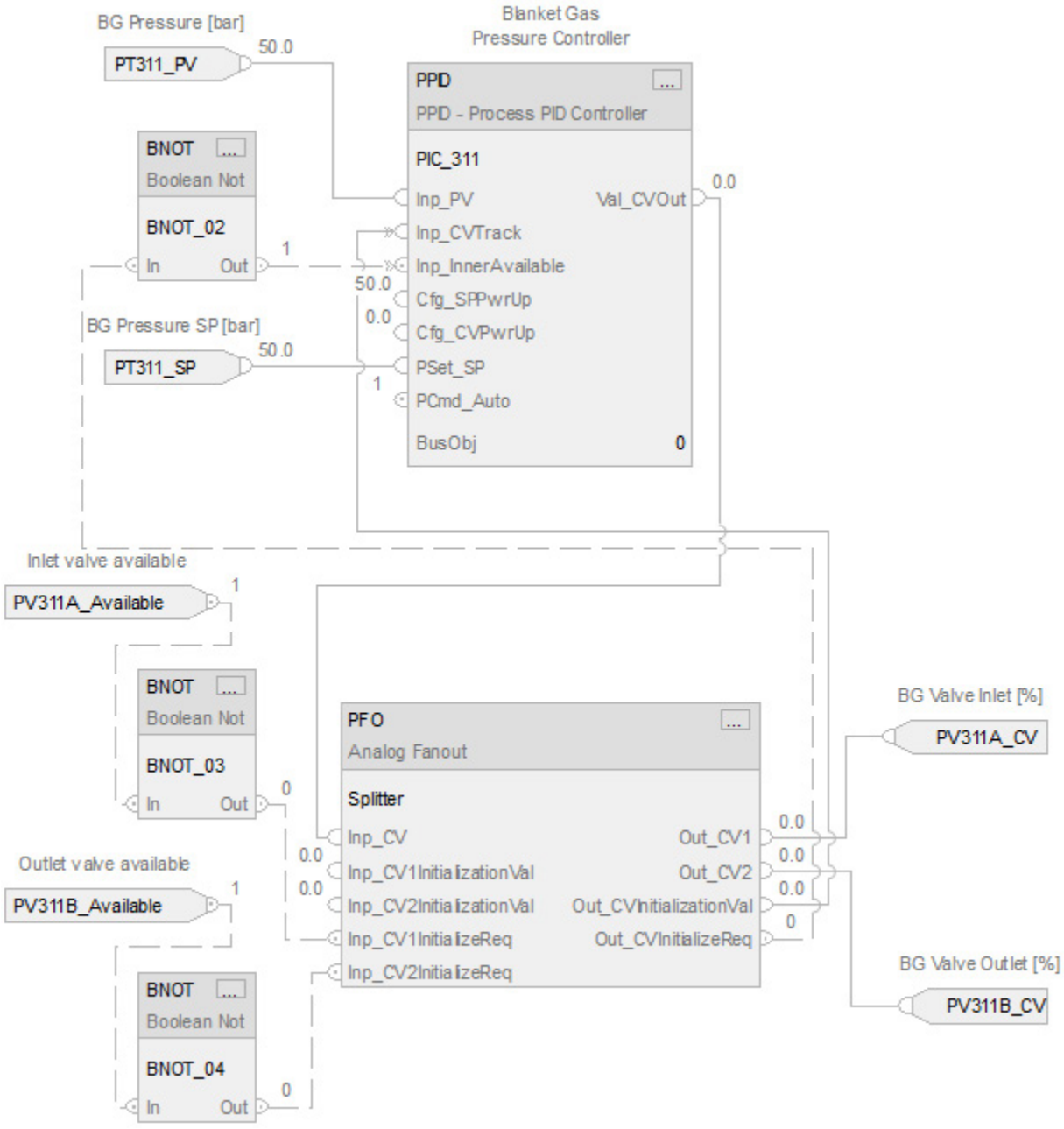
At CV=0 (Val_CVOut=0), both valves are closed. When the CV is positive, the pressurizing valve is open and the vent valve is kept closed. If the PPID instruction is configured with CV scaling limits Cfg_CVEUMax=100 and Cfg_CVEUMin=-100, then at CV=100 the pressurizing valve is wide open. When the CV is negative, the vent valve is open and the pressurizing valve is closed. At CV=-100, vent is wide open. CV splitting is done with the Process Analog Fanout (PFO) instruction.

Configure PPID with Cfg_CVEUMin=-100, Cfg_CVEUMax=100 and PFO with Cfg_CVEUMin=-100, Cfg_CVEUMax=100, Cfg_CV1Ratio=1, Cfg_CV1Offset=0, Cfg_CV1HiLim=100, Cfg_CV1LoLim=0, Cfg_CV2Ratio=-1, Cfg_CV2Offset=0, Cfg_CV2HiLim=100, Cfg_CV2LoLim=0.

The PPID instruction must receive an indication whether its downstream object can be controlled. If the downstream object is not ready for the PPID instruction, the instruction should track what the downstream block defines for the situation. Wire the negation of the initialization request (Out_CVInitializeReq) received by the PFO instruction from its downstream object to Inp_InnerAvailable of the PPID instruction. In addition, wire the PFO instruction's Out_CVInitializationVal to Inp_CVTrack of the PPID instruction. The PPID instruction will track this value when the downstream block is not ready for PPID control. The PPID instruction goes back to control without any bump if the downstream object becomes available again.

The example is shown in FBD.

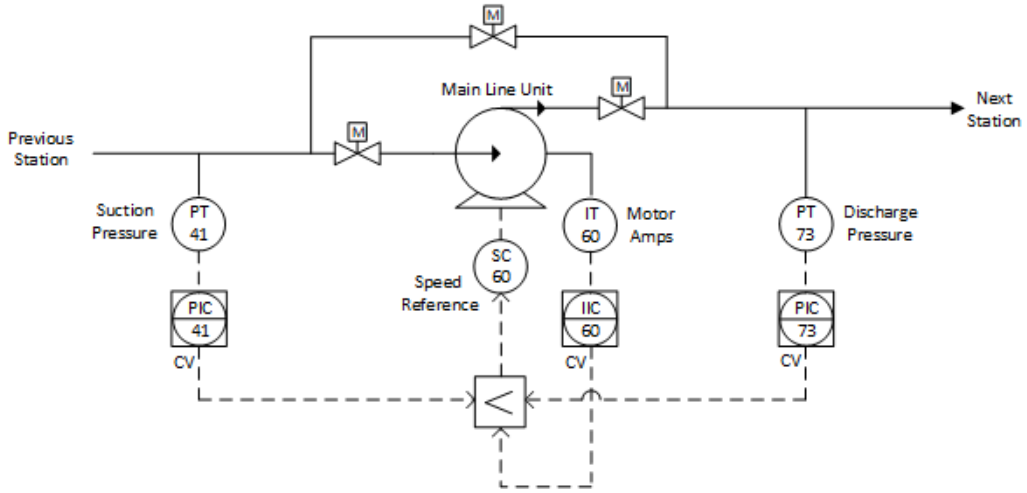
Function Block Diagram



Example 6: Override-select control

Consider the oil pipeline pump station in the figure below. In this example, the system measures suction pressure (41) and discharges pressure (73) and motor current (60). To control discharge pressure, use the variable-speed drive on the pump. The challenge is that if the suction pressure goes low, the pump cavitates. If the motor current goes too high, the drive trips and pressure upset is sent down the line. Both constraints act in the same direction. If the suction pressure goes low, the pump needs to slow down until

it recovers. If the motor current goes high, slowing down the pump reduces the power, and so reduces the motor current.



The Primary loop is the station discharge pressure. Suction pressure and motor current are Override loops. The low-select picks the lowest CV to send to the drive speed reference. Suction pressure and motor current loops' setpoints are set to the constraint threshold (where to start acting). When a constraint is approached, that loop's error gets small, its output drops, and it is selected. The selected CV is fed back to all three loops. The control scheme takes the advantage of using Inp_CVTrack for tracking final CV.

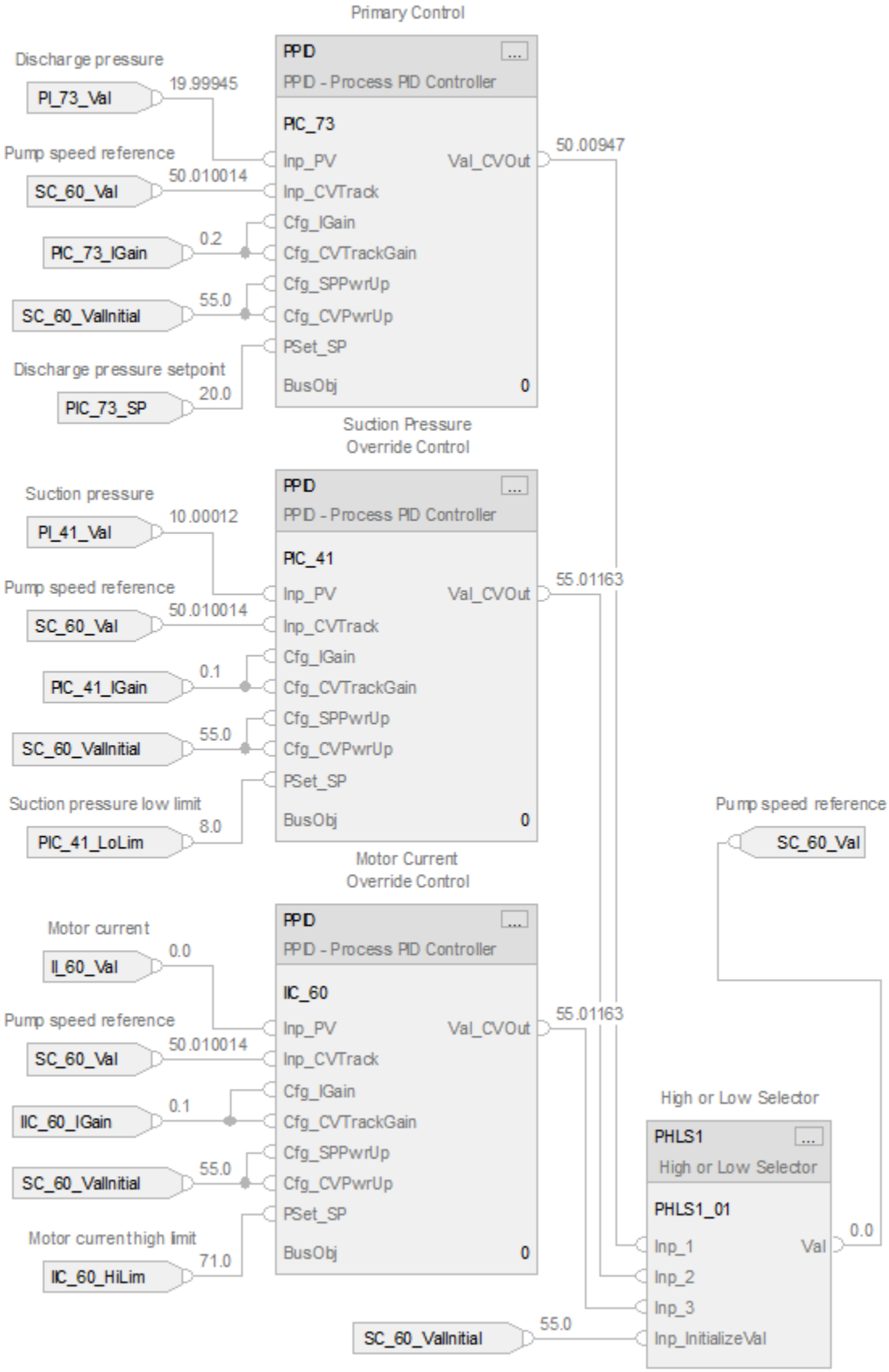
Recommended setting of the tracking parameter:

- Dependent gains (Cfg_Dependent = 1). Set Cfg_CVTrackGain = Cfg_IGain.
- Independent gains (Cfg_Dependent = 0). Set Cfg_CVTrackGain = Cfg_IGain/Cfg_PGain.

This setting leads to a steady state difference between selected (active) CV and unselected (tracking) CV equal to Cfg_PGain x Error which keeps proper leeway for the selected controller to control without unnecessarily frequent switching to another controller.

The example is shown in FBD.

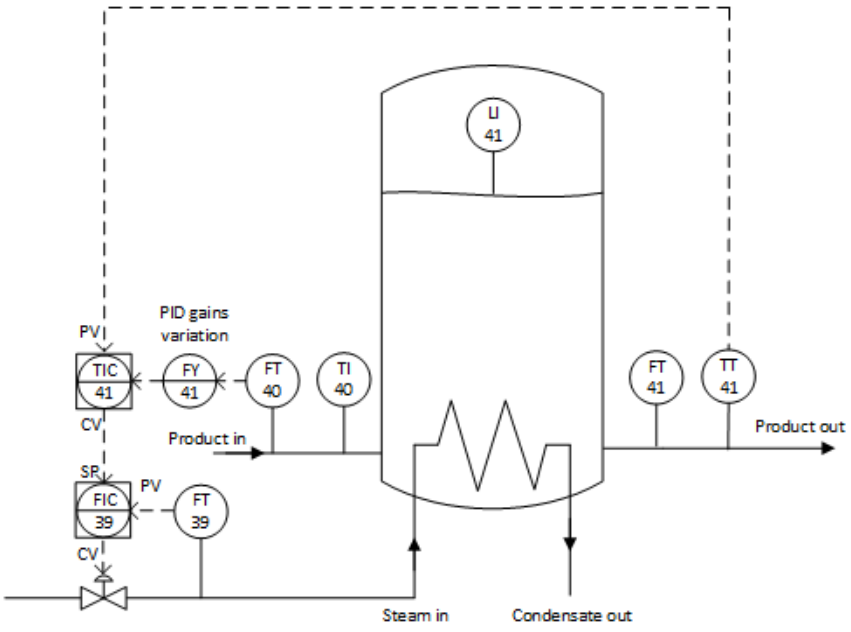
Function Block Diagram



Example 7: PID gain scheduling

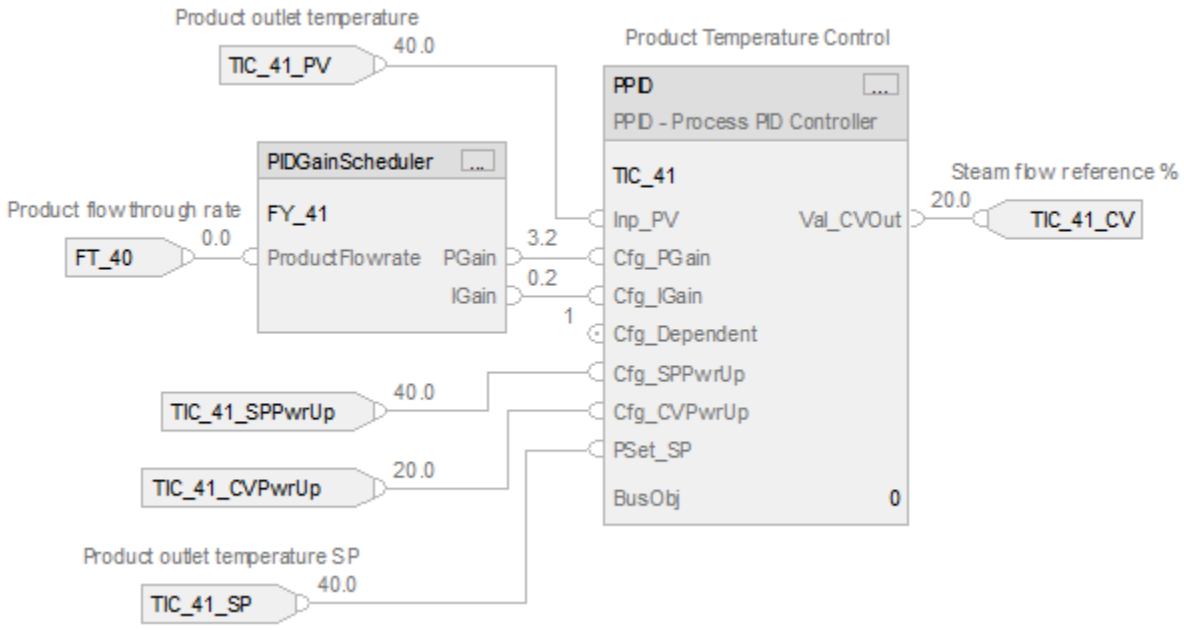
Consider the oil pipeline pump station in the figure below. In this example, the system measures suction pressure (41) and discharges pressure (73) and motor current (60). To control discharge pressure, use the variable-speed drive on the pump. The challenge is that if the suction pressure goes low, the pump cavitates. If the motor current goes too high, the drive trips and pressure upset is sent down the line. Both constraints act in the same direction. If the suction pressure goes low, the pump needs to slow down until it recovers. If the motor current goes high, slowing down the pump reduces the power, and so reduces the motor current.

In this example the gain scheduling technique is used to compensate for changes in process dynamics on-the-fly. Standard temperature controller TIC_41 reads outlet temperature TT_41 (PV) and calculates reference for steam flow control loop (CV) to keep product temperature at the setpoint. PPID is configured with dependent gains (TIC_41.Cfg_Dependent=1) and the overall gain TIC_41.Cfg_PGain changes with product flow FT_40. Formula for PGain calculation is application specific. In this example PGain is calculated in PIDGainScheduler AOI as a linear function of product flowrate with bias.



The example is shown in FBD.

Function Block Diagram



See also

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Data Conversions on page 1086

Process Pressure/Temperature Compensated Flow (PPTC)

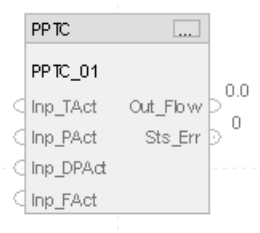
This information applies to the ControlLogix 5380P and 5580P controllers.

The Pressure/Temperature Compensated Flow (PPTC) instruction calculates a flow at standard temperature and pressure, essentially a mass flow rate, given a volumetric flow rate or differential pressure measurement. This instruction requires measurements of the actual temperature and pressure of the flowing gas.

Available Languages

Ladder Diagram

Function Block Diagram



Structured Text

PPTC(PPTC_01);

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PlantPAx Control	P_PRESS_TEMP_COMPENSATED	tag	Data structure required for proper operation of the instruction.

P_PRESS_TEMP_COMPENSATED Structure

Public members are standard, visible tag members that are programmatically accessible. Private, hidden members are used in HMI faceplates and are not programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	Description
EnableIn	BOOL	Enable input. Ladder Diagram: Corresponds to the rung-condition-in. Default is true.
Inp_InitializeReq	BOOL	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request when reinitialization is needed. The instruction clears this operand automatically. Default is true.
Inp_TAct	REAL	Actual (measured) temperature, can be absolute or common units. Valid = any float. Default is 0.0.
Inp_PAct	REAL	Actual (measured) pressure, can be absolute or common units. Valid = any float. Default is 0.0.

Public Input Members	Data Type	Description
Inp_DPAct	REAL	Actual (measured) differential pressure. Valid = any float. Default is 0.0.
Inp_FAct	REAL	Actual (measured) uncompensated flow in volumetric units. Valid = any float. Default is 0.0.
Cfg_LoFlowCutoff	REAL	If Out_Flow is less than this cutoff value, it is shown as 0.0. Valid = 0.0 to maximum positive float. Default is 0.0.
Cfg_TStd	REAL	Standard temperature in Inp_TAct units. Valid = any float. Default is 0.0.
Cfg_PStd	REAL	Standard pressure in Inp_PAct units. Valid = any float. Default is 0.0.
Cfg_TOffset	REAL	Zero input-units temperature in absolute units. Typically 273.15 Kelvins or 459.67 Rankine. Valid = 0.0 to maximum positive float. Default is 273.15.
Cfg_POffset	REAL	Zero input-units pressure in absolute units. Typically 14.696 PSIA. Valid = 0.0 to maximum positive float. Default is 14.696.
Cfg_DPRef	REAL	Reference (full-scale) differential pressure. Common value 100.0 inches WC. Valid = 0.0 to maximum positive float. Default is 100.0.
Cfg_FRef	REAL	Reference flow in volumetric units at reference dp. Valid = 0.0 to maximum positive float. Default is 1.0.
Cfg_UseDP	BOOL	1 = Use Inp_DPAct (square root curve) to calculate flow. 0 = use Inp_FAct (linear). Default is false.

Public Output Members	Data Type	Description
EnableOut	BOOL	Enable Output. This output state always reflects EnableIn input state.
Out_Flow	REAL	Compensated flow (at standard temperature and pressure: mass flow).
Sts_Initialized	BOOL	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.
Sts_Err	BOOL	1 = Error in configuration: See detail bits (Sts_Errxxx) for reason.
Sts_ErrTStd	BOOL	1 = Invalid standard temperature (Cfg_TStd + Cfg_TOffset must be > 0.0).
Sts_ErrPStd	BOOL	1 = Invalid standard pressure (Cfg_PStd + Cfg_PStd must be > 0.0).
Sts_ErrDPRef	BOOL	1 = Invalid reference differential pressure (must be > 0.0 if DP used).
Sts_ErrFRef	BOOL	1 = Invalid reference flow (at reference DP) (must be > 0.0).

Operation

The PPTC instruction is intended as a calculation function only, between other blocks. If a faceplate or alarms are needed, the calculated output from the instruction can be sent to a PAI (analog input) instruction for alarming and display.

The PPTC instruction:

- Takes as its primary input either a volumetric flow rate or a differential pressure across a flow element, such as an orifice plate or pitot tube. When a differential pressure is used, the PPTC instruction allows

configuration of the volumetric flow rate for a given differential pressure.
- Accepts a temperature in common units (Fahrenheit or Celsius degrees) or in absolute units (Rankine degrees or Kelvins).
- Accepts a pressure in common units (PSIG, kPa Gauge, or MPa Gauge) or in absolute units (PSIA, kPa Absolute, MPa Absolute).
- Has user-configurable standard conditions, such as 14.696 PSIA and 60 °F, or 101.325 kPa and 0 °C.
- Determines flow at the specified standard conditions by using the Ideal Gas Law (PV = nRT) to adjust from the given temperature and pressure to the standard temperature and pressure.

Virtualization

Virtualization is not applicable to the PPTC instruction.

Initialization

The instruction is normally initialized in the instruction first run. Re-initialization can be requested any time by setting Inp_InitializeReq = 1. For proper initialization, when adding the instruction while performing an online edit of the code, make sure that Inp_InitializeReq = 1, the default value.

Configuration of Strings for HMI

Configure strings for HMI faceplates (FT View) and for the Logix Designer configuration dialog box.

- Description – Description of PPTC tag
- Label for graphic symbol – Label metadata of PPTC tag
- Display Library for HMI Faceplate call-up – Library metadata of PPTC tag
- Instruction name – Instruction metadata of PPTC tag
- Area name – Area metadata of PPTC tag
- URL link – URL metadata of PPTC tag
- Actual differential pressure units – Engineering Unit metadata of .Inp_DPAct
- Actual uncompensated flow in volumetric units – Engineering Unit metadata of .Inp_FAct
- Actual pressure, can be abs or common units – Engineering Unit metadata of .Inp_PAct
- Actual temperature, can be abs. or common units – Engineering Unit metadata of .Inp_TAct

- Compensated flow (at standard temperature and pressure: mass Flow) units – Engineering Unit metadata of .Out_Flow

Monitor the PPTC Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Rung-condition-out is cleared to false.
Instruction first run	The instruction executes normally.
Rung-condition-in is false	Set rung-condition-out to rung-condition-in.
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	EnableOut is cleared to false.
Instruction first run	See Instruction first run in the Ladder Diagram table.
Instruction first scan	See Instruction first run in the Function Block Diagram table.
EnableIn is false	EnableOut is cleared to false.
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

In Structured Text, EnableIn is always true during normal scan. The instruction executes when it is in the control path activated by the logic.

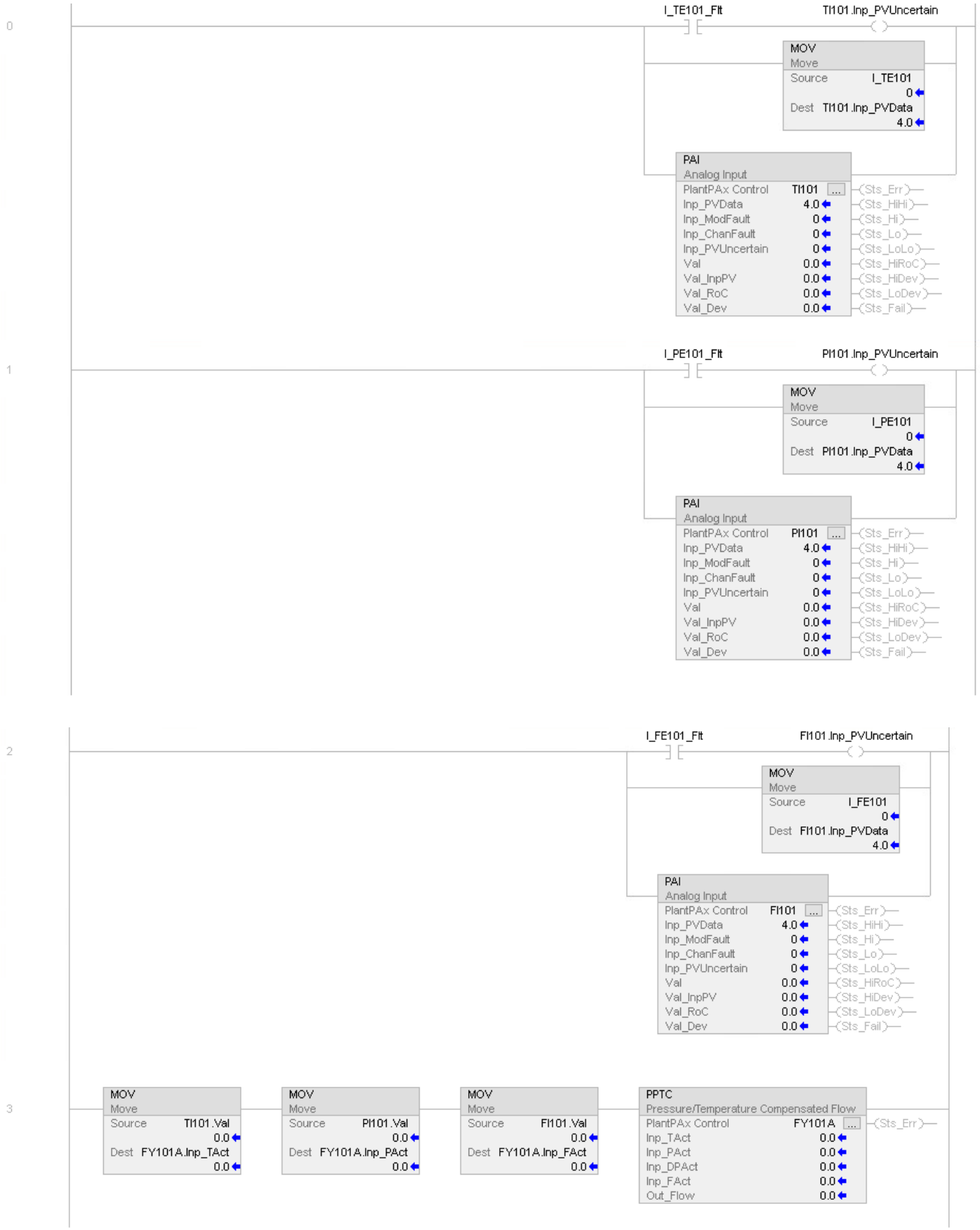
Condition/State	Action Taken
Prescan	See Prescan in the Function Block Diagram table.
Instruction first run	See Instruction first run in the Function Block Diagram table.
EnableIn is true	See EnableIn is true in the Function Block Diagram table.
Postscan	See Postscan in the Function Block Diagram table.

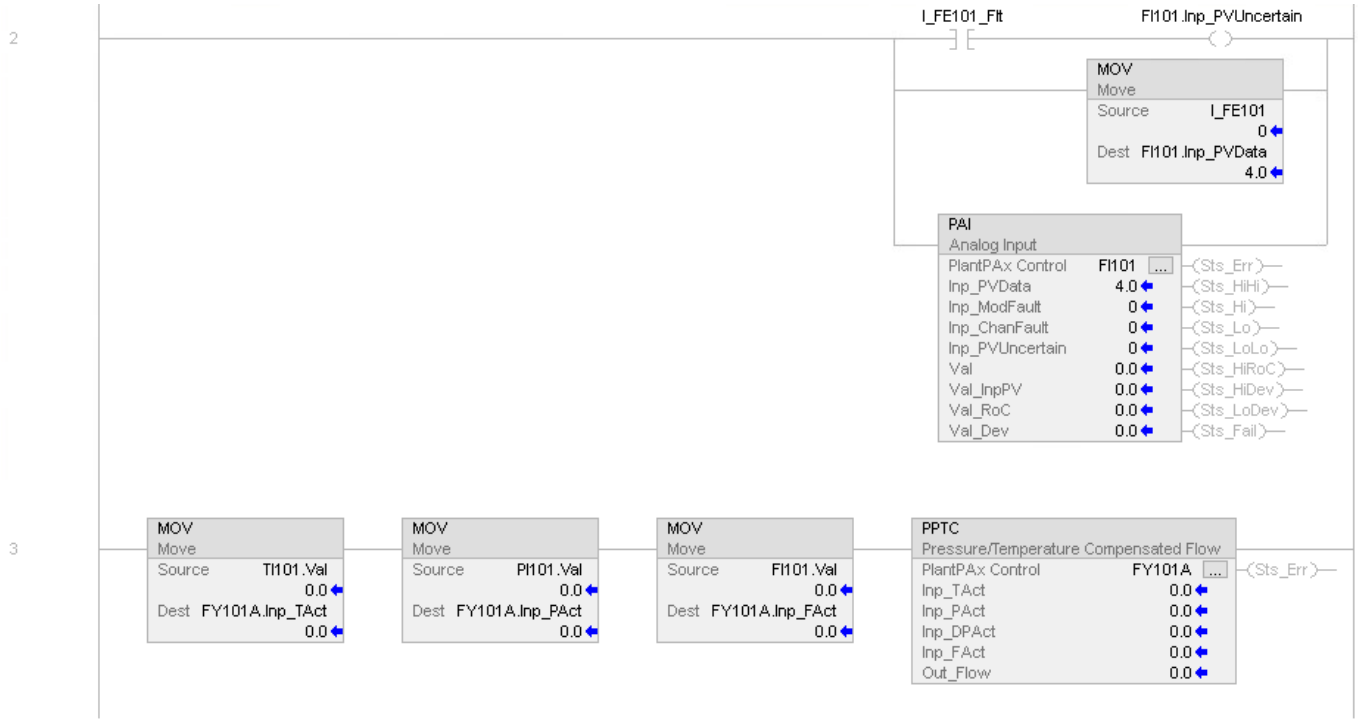
Example

In this example, the PPTC instruction to determine the flow rate of compressed nitrogen at a standard pressure and flow. This can provide a more accurate measurement for custody transfer or control calculations where there is variability in environmental conditions and the flow transmitter is not capable of performing the compensation.

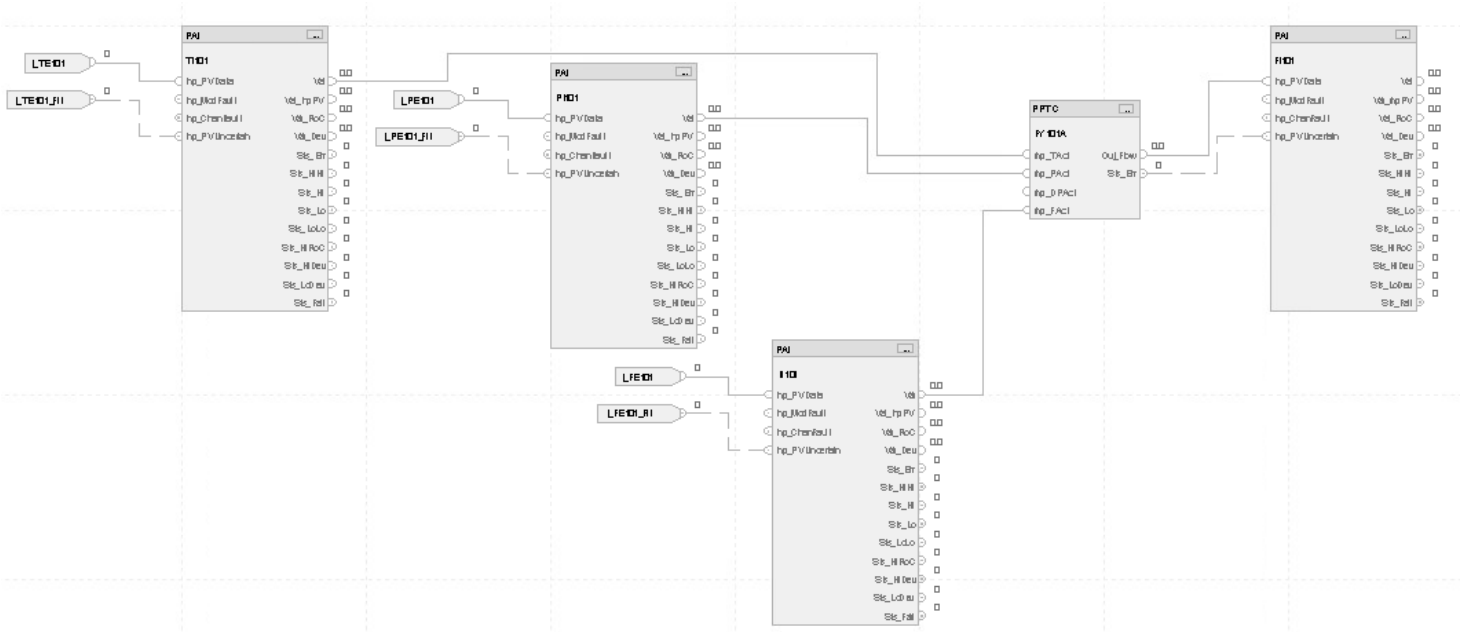
In this case, the PPTC instruction measures flow from a dp-transmitter. The transmitter provides the controller with a value that has been scaled to volumetric flow but not compensated for environmental temperature and pressure. We also have temperature and pressure measurements from where the flow is measured. In this example, the desired standard pressure and flow is 0 psig and 15 °C.

Ladder Diagram





Function Block Diagram



Structured Text

```
TI101.Inp_PVData:=I_TE101;

TI101.Inp_PVUncertain:=I_TE101_Flt;

PAI(TI101);
```

```
PI101.Inp_PVData:=I_PE101;
PI101.Inp_PVUncertain:=I_PE101_Flt;
PAI(PI101);
FI101.Inp_PVData:=I_FE101;
FI101.Inp_PVUncertain:=I_FE101_Flt;
PAI(FI101);
FY101A.Inp_Tact:=TI101.Val;
FY101A.Inp_Pact:=PI101.Val;
FY101A.Inp_Fact:=FI101.Val;
PPTC(FY101A);
FI101.Inp_PVData:=FY101A.Out_Flow;
FI101.Inp_PVUncertain:=FY101A.Sts_Err;
PAI(FI101);
```

See also

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Process Restart Inhibit (PRI)

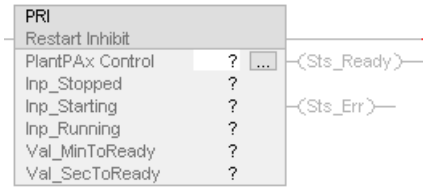
This information applies to the ControlLogix 5380P and 5580P controllers.

Use the Process Restart Inhibit instruction for Large Motor (PRI) instruction to prevent large motors from starting repeatedly. The high starting current for a large motor causes heating. Continual starts or start attempts in a short period overheat the motor windings and damage the motor.

The PRI instruction provides a rule-based state model for restarts. Do not use the instruction to model or monitor heating and replace sensor-based motor monitoring devices. Use the instruction to avoid overstressing a motor.

Available Languages

Ladder Diagram



Function Block Diagram

Structured Text

PRI (PRI tag);

Operands

IMPORTANT Unexpected operation may occur if one of the Cold or Hot timers are set to 0.

Configuration Operands

Operand	Type	Format	Description
PlantPAx Control	P_RESTART_INHIBIT	tag	PRI structure

P_RESTART_INHIBIT_INPUT Structure

Public members are standard (visible) Tag members that are programmatically accessible. Private (hidden) members are used in HMI faceplates and are not programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	Description
EnableIn	BOOL	Enable Input - System Defined Parameter Default is true.

Public Input Members	Data Type	Description
Inp_Stopped	BOOL	Equipment is confirmed Stopped Default is false.
Inp_Starting	BOOL	Equipment is Starting, indicating a start attempt Default is false.
Inp_Running	BOOL	Equipment is confirmed Running Default is true.
Cfg_ThreeColdStarts	REAL	Time within which three starts are allowed if cold (hr) Default is 0.0.
Cfg_FirstFailCold	REAL	Time for cold motor to wait after 1st start failure before ready to start (hr) Default is 0.0.
Cfg_SubseqFailCold	REAL	Time for cold motor to wait after 2nd and subsequent start failure before ready (hr) Default is 0.2.
Cfg_FirstFailHot	REAL	Time for hot motor to wait after 1st start failure before ready to start (hr) Default is 0.0.
Cfg_SubseqFailHot	REAL	Time for hot motor to wait after 2nd and subsequent start failure before ready (hr) Default is 0.2.
Cfg_HotRestartOK	REAL	Time for hot motor to run so it can immediately restart after stop (hr) Default is 0.0.
Cfg_RestartHot	REAL	Time for hot motor to wait after stop if stopped before Hot Restart OK time (hr) Default is 0.0.
Cfg_HotToCold	REAL	Time for a stopped hot motor to become cold (hr) Default is 0.7.
Inp_InitializeReq	BOOL	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request when reinitialization is needed. The instruction clears this operand automatically. Default is true.

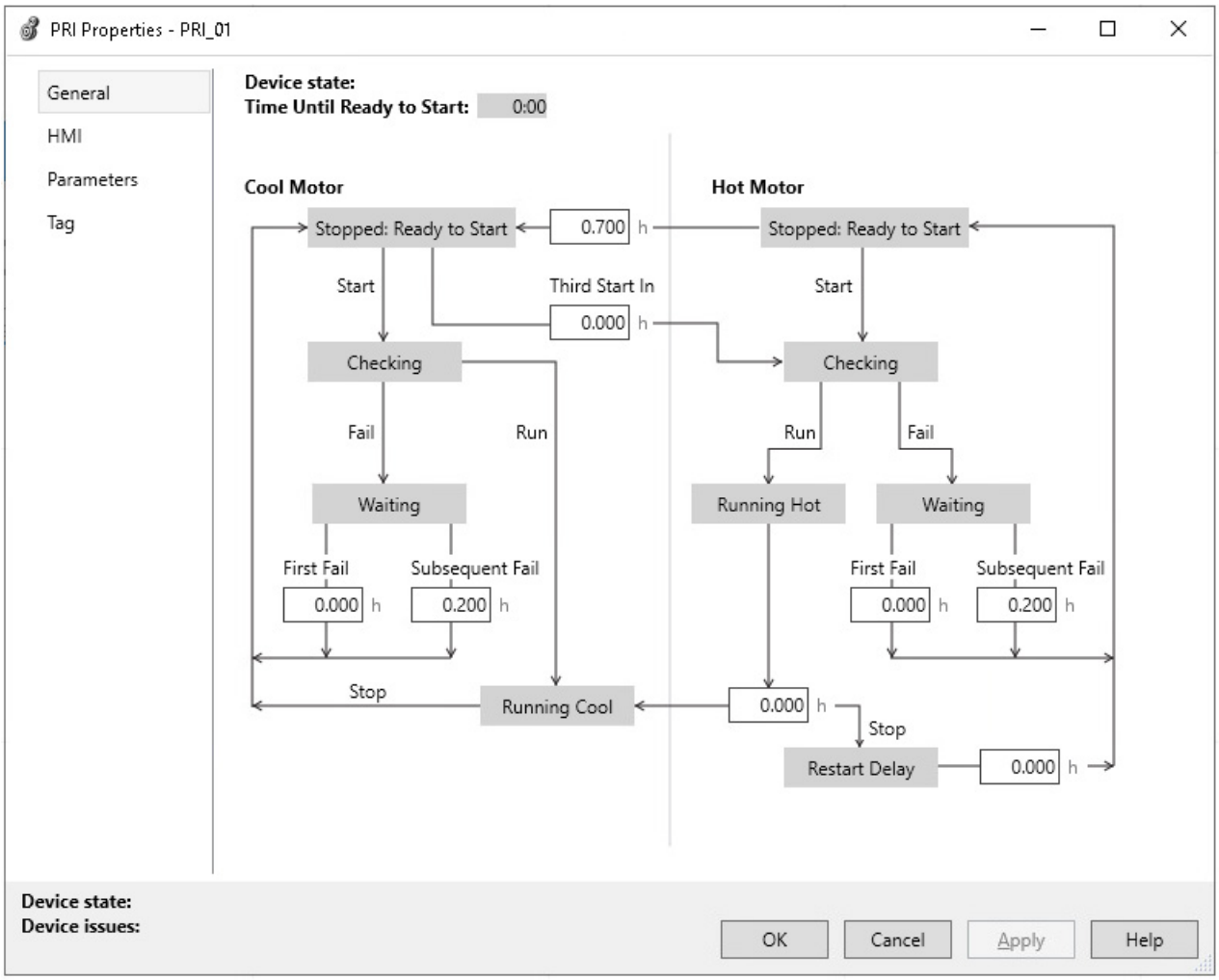
Public Output Members	Data Type	Description
EnableOut	BOOL	Enable Output - System Defined Parameter
Val_MinToReady	DINT	Minutes yet inhibited before ready to start (mmm:ss)
Val_SecToReady	DINT	Seconds yet inhibited before ready to start (mmm:ss)
Sts_bFdbk	SINT	Device Feedback 0 =None/Multiple, 1 = Stopped, 2 = Starting, 3 = Running
Sts_State	SINT	State Number (see State Diagram in docs) for HMI
Sts_Ready	BOOL	Permissive for unit to start 1 = ready, 0 = not ready
Sts_Err	BOOL	1 = Error in Config: Invalid Time (use 0.0 to 2147483)
Sts_Initialized	BOOL	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.

Operation

The PRI instruction provides these capabilities:

- Provides a rule-based state model for restarts and is not intended to model or monitor the motor heating.
- Display of the time before ready state; the time is displayed in minutes and seconds and is configurable. The ready state is also displayed independently of the time.
- Ready status is determent by the cold ready to start time (First start) and the hot ready to start time.

- The cold ready to start time will be determent by the cold first fail time and the cold subsequent fail time.
- The hot ready to start time will be determent by the hot first fail time, the cold subsequent fail time and the hot restart delay time.



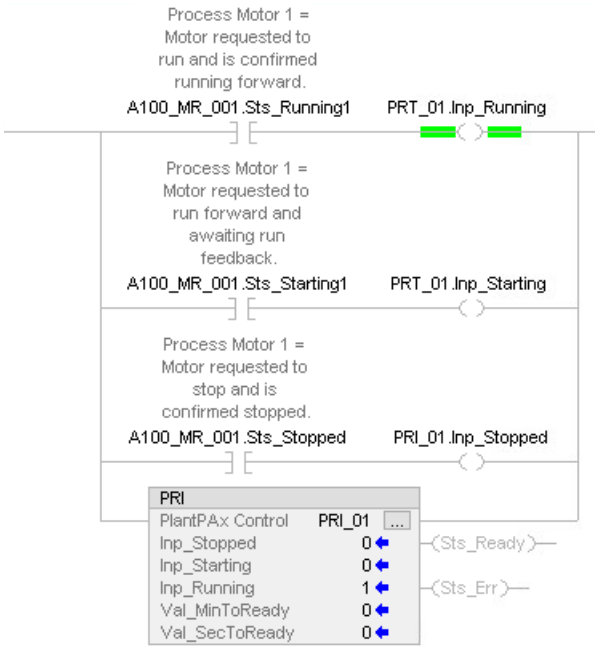
Configuration of Strings for HMI

Configure strings for HMI faceplates and for the Logix Designer configuration dialog box. The strings are set to extended properties of tag items. Configure the strings in the Logix Designer application only.

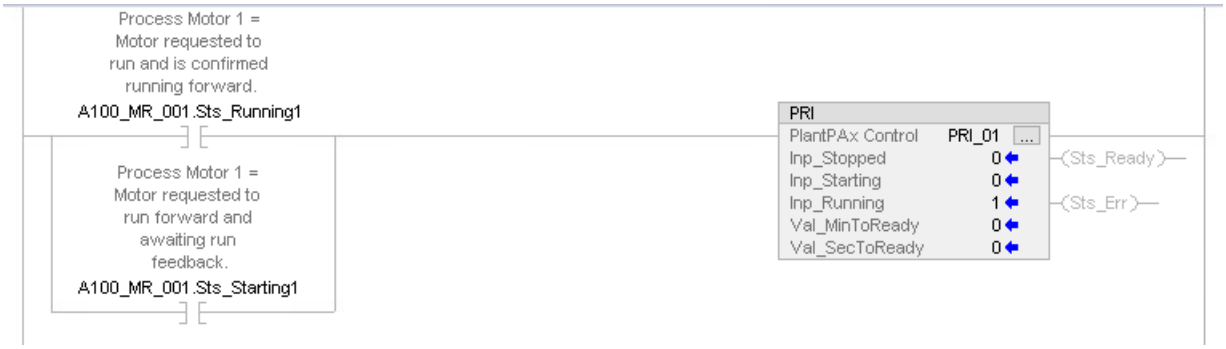
- Description
- Label for graphic symbol
- Display Library for HMI Faceplate call-up
- Instruction name
- Area name
- URL link

Implementation

This illustration shows normal implementation with the input condition mapped to Inp_Running, Inp_Starting, and Inp_Stopped on a separate branch.



This illustration shows the implementation with the input condition mapped to the PRI instruction using the rung-condition-in. When the rung-condition-in is false (EnableIn is false) the instruction executes normally. To use the rung-condition-in mapping method, set Inp_Running to 1, its default value.



Monitor the PRI Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Rung-condition-out is cleared to false.
Instruction first run	The instruction executes normally.
Rung-condition-in is false	Set rung-condition-out to rung-condition-in.
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	EnableOut is cleared to false.
Instruction first run	See Instruction first run in the Ladder Diagram table.
Instruction first scan	See Instruction first run in the Function Block Diagram table.
EnableIn is false	EnableOut is cleared to false.
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

In Structured Text, EnableIn is always true during normal scan. Therefore, if the instruction is in the control path activated by the logic it will execute.

Condition/State	Action Taken
Prescan	See Prescan in the Function Block Diagram table.
Instruction first run	See Instruction first run in the Function Block Diagram table.
EnableIn is true	See EnableIn is true in the Function Block Diagram table.
Postscan	See Postscan in the Function Block Diagram table.

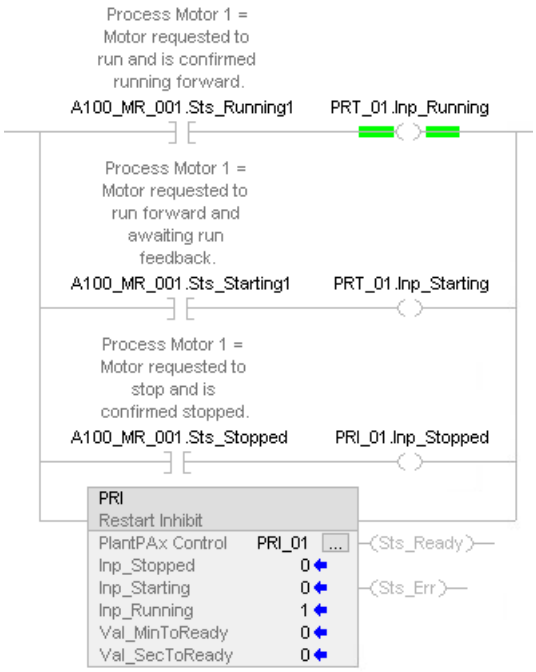
Example

In this example, tag A100_MR_001 is the motor value monitored by the PRI instruction.

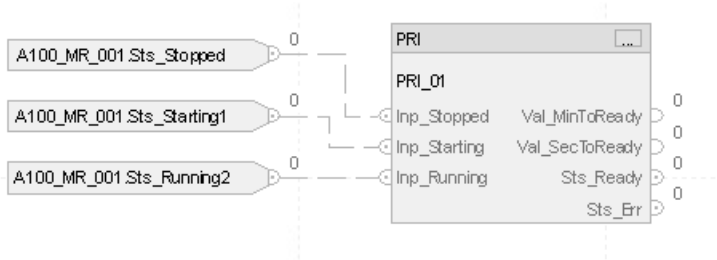
Inp_Running is connected to the Motor Running status tag (A100_MR_001.Sts_Running) that comes from the Sts_Running output of the P_Motor instruction instance for this motor (A100_MR_001). Inp_Starting is connected to the Motor Starting status tag (A100_MR_001.Sts_Starting) that comes from the Sts_Starting output of the P_Motor instruction instance for this motor (A100_MR_001). Inp_Stopped is connected to the Motor Stopped status tag (A100_MR_001.Sts_Stopped) that comes from the Sts_Stopped output of the P_Motor instruction instance for this motor (A100_MR_001).

Finally, PRI_01 is the output tag that will indicate the status of A100_MR_001 with appropriate delays and number of running and starts/attempts based on whether the motor is allowed to start again.

Ladder Diagram



Function Block Diagram



Structured Text

```
PRI_01.Inp_Stopped := A100_MR_001.Sts_Stopped;
PRI_01.Inp_Starting := A100_MR_001.Sts_Stopped;
PRI_01.Inp_Running := A100_MR_001.Sts_Running;
PRI(PRI_01);
```

See also

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

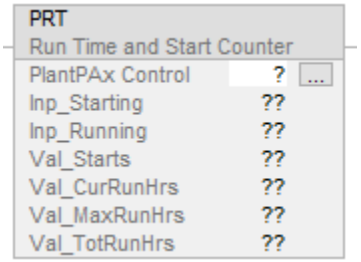
Process Run Time and Start Counter (PRT)

This information applies to the ControlLogix 5380P and 5580P controllers.

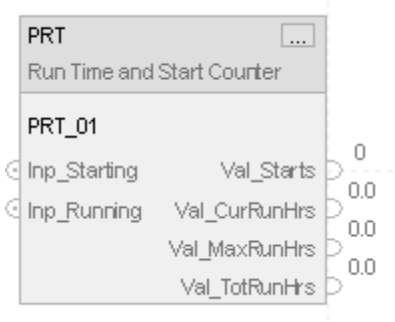
The Process Run Time and Start Counter (PRT) instruction records the total run time and number of instances the motor or other equipment starts. The PRT is a software implementation of the mechanical hour meter that displays the total motor runtime. Maintenance personnel use the run time and equipment start variables to create a maintenance schedule for the applicable equipment.

Available Languages

Ladder Diagram



Function Block Diagram



Structured Text

```
PRT(PRT_01);
```

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PRT	P_RUN_TIME	tag	PRT structure

P_RUN_TIME Structure

Public Input Members	Data Type	Description
EnableIn	BOOL	Enable Input - System Defined Parameter.
Inp_Starting	BOOL	1 = Equipment is starting. Default = false.
Inp_Running	BOOL	1 = Equipment is confirmed running. Default = true.
PCmd_ClearStarts	BOOL	Program Command to clear count of starts. The instruction clears this operand automatically. Default = false.
PCmd_ClearMaxHrs	BOOL	Program Command to clear the maximum, continuous runtime for any start. The instruction clears this operand automatically. Default = false.
PCmd_ClearTotHrs	BOOL	Program Command to clear the total run time. The instruction clears this operand automatically. Default = false.

Public Output Members	Data Type	Description
EnableOut	BOOL	Enable Output – System Defined Parameter.
Val_Starts	DINT	Total number of equipment starts or attempts.
Val_CurRunHrs	REAL	Current running time this start (hours).
Val_MaxRunHrs	REAL	Maximum continuous running time for a given start (hours).
Val_TotRunHrs	REAL	Total accumulated running time (hours).

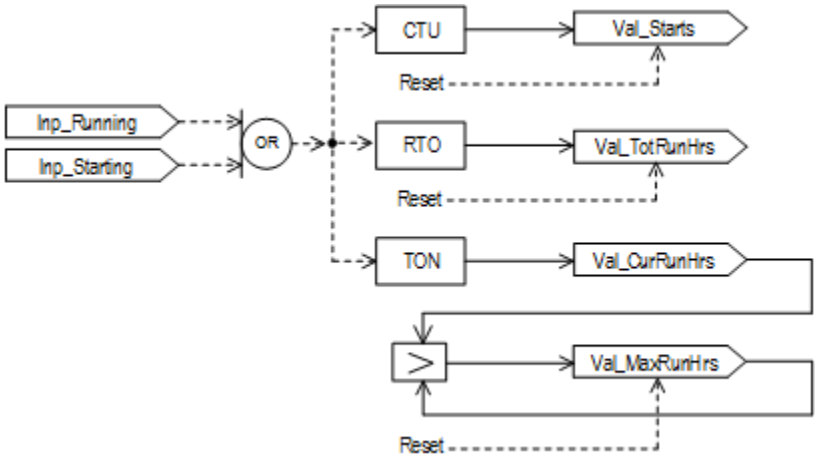
Private Input Members	Data Type	Description
MCmd_ClearStarts	BOOL	Maintenance Command to clear count of starts. The instruction clears this operand automatically. Default = false.
MCmd_ClearMaxHrs	BOOL	Maintenance Command to clear the maximum, continuous runtime for any start. The instruction clears this operand automatically. Default = false.
MCmd_ClearTotHrs	BOOL	Maintenance Command to clear the total run time. The instruction clears this operand automatically. Default = false.

Operation

The PRT instruction:

- Accumulates and displays the total running time for the associated equipment.
- Accumulates and displays the count of starts or start attempts for the associated equipment.
- Shows the amount of run time since the last start, or the length of the current run. This total is held after the equipment is stopped, until the next start, when it is reset to zero.
- Shows the maximum amount of time for any single run; this is the highest value achieved by the previous total.
- Allows maintenance personnel, but not operators, to clear individually the total run time, starts count, or maximum single run time. This lets the times be reset when the motor or other equipment is serviced, rebuilt or replaced.

This diagram illustrates the functionality of the PRT instruction:



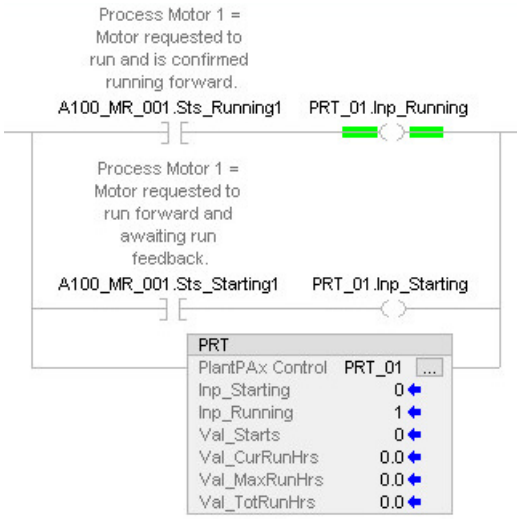
Configuration of Strings for HMI

Configure strings for HMI faceplates (FactoryTalk View) and for the Logix Designer configuration dialog box. The strings are set to extended properties of tag items. Configure the strings in the Logix Designer application only.

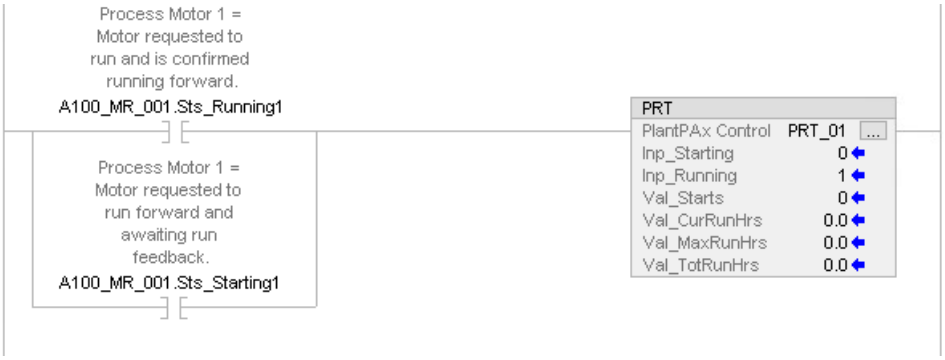
- Description
- Label for graphic symbol
- Display Library for HMI Faceplate call-up
- Instruction name
- Area name
- URL link

Implementation

This illustration shows normal implementation with the input condition mapped to Inp_Running and Inp_Starting on a separate branch.



This illustration shows the implementation with the input condition mapped to the PRT instruction by using the rung-condition-in.



When the rung-condition-in is false (EnableIn is false) the instruction executes normally. To use the rung-condition-in mapping method, set Inp_Running to 1, its default value.

Monitor the PRT instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Total run time, Maximum run time and starts count are retained through a power cycle. Current run time is reset (internally only, Val_CurHours is maintained for display). All Prog, Maint, commands that are automatically cleared each execution are cleared.

Instruction first run	Total run time, Maximum run time and starts count are retained through a power cycle. Current run time is reset (internally only, Val_CurHours is maintained for display). All Prog, Maint, commands that are automatically cleared each execution are cleared.
Rung-condition-in is false	EnableIn False is treated the same as Motor Stopped: Commands are still processed, total run time is held (RTO), the Starts counter is prepared for the next start (.CU clears on CTU with AFI), and the Current Run Time is cleared (TON false). Command Processing: • This object has no Command Source and • Commands are accepted regardless of Source. Commands should be restricted to authorized personnel (typically at a maintenance level). • PCmd_ClearTotHrs: Program Command to Clear Total Runtime Hours • MCmd_ClearTotHrs: Maintenance Command to Clear Total Runtime Hours • PCmd_ClearStarts: Program Command to Clear Count of Starts • MCmd_ClearStarts: Maintenance Command to Clear Count of Starts • PCmd_ClearMaxHrs: Program Command to Clear Maximum Single Run Hours • MCmd_ClearMaxHrs: Maintenance Command to Clear Maximum Single Run Hours • The number of starts is directly reported as an integer Value. • The current hours working register is cleared, but the Value is left in place for display until the next run begins. • The Total Running Time is reported as a REAL number of Hours. (This will always be in completed tenths of an hour, like a mechanical hours counter.)
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.

Function Block Diagram

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Instruction first run	See Instruction first run in the Ladder Diagram table.
Instruction first scan	See Instruction first run in the Ladder Diagram table.
EnableIn is false	See Rung-condition-in is false in the Ladder Diagram table.
EnableIn is true	See Rung-condition-in is true in the Ladder Diagram table.

Structured Text

In Structured Text, EnableIn is always true during normal scan. Therefore, if the instruction is in the control path activated by the logic, it will execute.

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Instruction first run	See Instruction first run in the Ladder Diagram table.
EnableIn is true	See Rung-condition-in is true in the Ladder Diagram table.
Postscan	EnableIn and EnableOut bits are cleared to false.

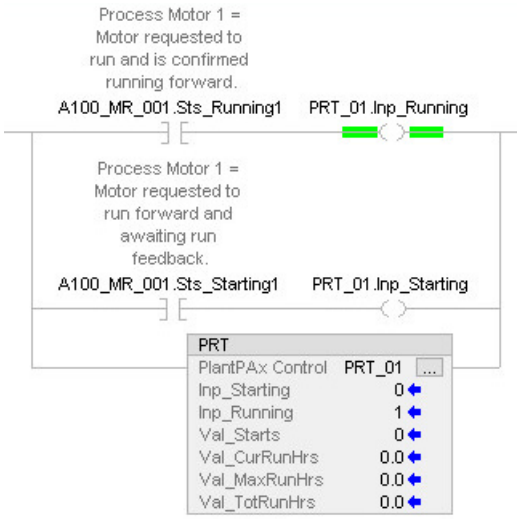
Example

In the following example, tag A100_MR_001 is the motor value monitored by the PRT instruction. This tag provides a Boolean indication of motor run time value.

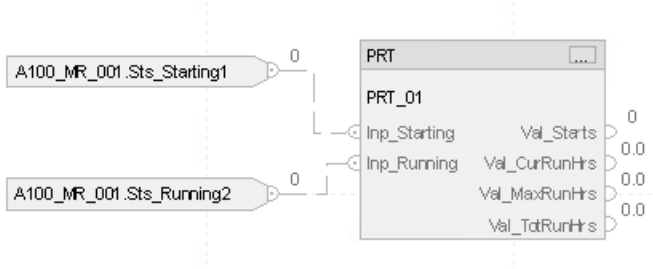
Inp_Runnung is connected to the Motor Running status tag (A100_MR_001.Sts_Running) that comes from the Sts_Running output of the P_Motor instruction instance for this motor (A100_MR_001). Inp_Starting is connected to the Motor Starting status tag (A100_MR_001.Sts_Starting) that comes from the Sts_Starting output of the P_Motor instruction instance for this motor (A100_MR_001).

Finally, A100_MR_001_RT.Val_TotRunHrs is the output tag that will indicate the total running hours of A100_MR_001. There is also current running hours (A100_MR_001_RT.Val_CurRunHrs) of the motor and maximum running hours (A100_MR_001_RT.Val_MaxRunHrs) of the motor available.

Ladder Diagram



Function Block Diagram



Structured Text

PRT_01.Inp_Starting:=A100_MR_001.Sts_Starting;

PRT_01.Inp_Running:=A100_MR_001.Sts_Running;

PRT(PRT_01);

See also

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Process Tank Strapping Table (PTST)

This information applies to the ControlLogix 5380P and 5580P controllers.

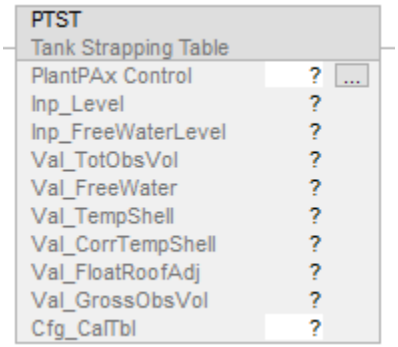
The Process Tank Strapping Table (PTST) instruction calculates the volume of product in an upright cylindrical tank, given the level of the product and the tank calibration table. The instruction can compensate for:

- Free water at the bottom of the tank, given a product/water interface level.
- Thermal expansion of the tank shell, given the coefficient of linear expansion of the shell material and product and ambient temperatures.
- A floating tank roof, given the product density is provided.

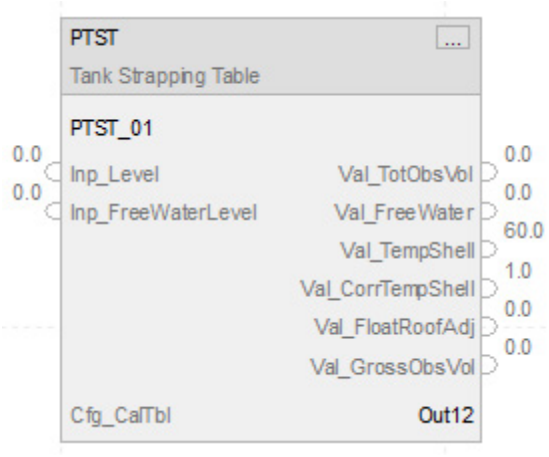
The Process Tank Strapping Table (PTST) instruction is a calculation function, between blocks.

Available Languages

Ladder Diagram



Function Block Diagram



Structured Text

PTST(PTST_tag, Cfg_CalTbl);

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PlantPAx Control	P_TANK_STRAPPING_TABLE	tag	Data structure required for proper operation of instruction.
Cfg_CalTbl	P_STRAPPING_TABLE_ROW	tag	Tank calibration table, level to volume.

P_TANK_STRAPPING_TABLE Structure

Public members are standard, visible tag members that are programmatically accessible. Private, hidden members are used in HMI faceplates and are not programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	Description
EnableIn	BOOL	Enable input. Ladder Diagram: Corresponds to the rung-condition-in. Default is true.

Public Input Members	Data Type	Description
Inp_InitializeReq	BOOL	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request when reinitialization is needed. The instruction clears this operand automatically. Default is true.
Inp_Level	REAL	Tank innage level, in feet or meters. Valid = any float. Default is 0.0.
Inp_FreeWaterLevel	REAL	Tank innage free water interface level, in feet or meters. Valid = any float. Default is 0.0.
Inp_ObsAPI	REAL	Observed density, or degrees API, at product temperature. This is used for floating roof compensation to calculate displacement based on weight of roof. Valid = any float. Default is 30.5.
Inp_AvgProdTemp	REAL	Average product temperature input in degrees Fahrenheit or Celsius. Valid = any float. Default is 60.0.
Inp_AmbTemp	REAL	Ambient temperature input in degrees Fahrenheit or Celsius. Valid = any float. Default is 60.0.
Cfg_MinorPerMajor	REAL	Table minor units, in inches, centimeters, millimeters, per major unit, in feet or meters. Type 0.0 if minor units not used. Valid = any float. Default is 12.0.
Cfg_HasCorrTempShell	BOOL	0 = No correction for temperature of tank shell. 1 = Include correction for temperature of tank shell. Default is false.
Cfg_HasFloatRoofAdj	BOOL	0 = Do not use floating roof adjustment. 1 = Include floating roof adjustment to account for displacement of fluid level. Default is false.
Cfg_HasMoreObj	BOOL	1 = Tells HMI an object with more info is available. Default is false.
Cfg_CalTemp	REAL	Temperature of tank calibration (typically 60 °F or 15 °C). Valid = any float. Default is 60.0.
Cfg_ShellCoefOfExp	REAL	Tank shell linear coefficient of thermal expansion (1 per degree Fahrenheit or 1 per Celsius). Valid = any float. Default is 0.0000062.
Cfg_K	REAL	Temperature weighting (type 0.0 for insulated tank). See API MPMS 2.2A Appendix D. Valid = any float. Default is 7.0.
Cfg_FloatRoofLevel	REAL	Lowest level at which to add or subtract floating roof compensation (feet). Valid = any float. Default is 0.0.
Cfg_FloatRoofCalAPI	REAL	Degrees API for which table includes floating roof data. Valid = any float. Default is 30.5.
Cfg_FloatRoofVolPerAPI	REAL	Adjustment to table values for API <> CalAPI (volume/degrees API, typically a negative number). Valid = any float. Default is -2.5.

Public Output Members	Data Type	Description
EnableOut	BOOL	Enable Output. This output state always reflects EnableIn input state.
Val_TotObsVol	REAL	Raw total observed volume from Calibration Table (barrels, gallons, liters).
Val_FreeWater	REAL	Free water volume (barrels, gallons, liters).
Val_TempShell	REAL	Calculated tank shell temperature in degrees Fahrenheit or Celsius.
Val_CorrTempShell	REAL	Correction for temperature of tank shell (multiplier).
Val_FloatRoofAdj	REAL	Floating roof adjustment volume (barrels, gallons, liters).

Public Output Members	Data Type	Description
Val_GrossObsVol	REAL	Primary value: Gross observed volume (see API MPMS 12.1.1).
Sts_Initialized	BOOL	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.
Sts_UnderMin	BOOL	Inp_Level is below lowest level in strapping table.
Sts_OverMax	BOOL	Inp_Level is above highest level in strapping table.

Public InOut Members	Data Type	Description
Cfg_CalTbl	P_STRAPPING_TABLE_ROW[2]	Tank calibration table (level to volume).

P_STRAPPING_TABLE_ROW Structure

Members	Data Type	Description
Major	REAL	Number of major units (feet, meters). Valid = 0.0 to maximum positive float.
Minor	REAL	Number of minor units (inches, centimeters, or millimeters). Valid = 0.0 to maximum positive float.
Volume	REAL	Tank volume (oil barrels, gallons, liters) at given level (feet, inches). Valid = 0.0 to maximum positive float.

Operation

Configuration of Strings for HMI

Configure strings for HMI faceplates (FT View) and for the Logix Designer configuration dialog box. The strings are set to extended properties of tag items. Configure the strings in Logix Designer only.

- Description
- Label for graphic symbol
- Display Library for HMI Faceplate call-up
- Instruction name
- Area name
- URL link
- More Information
- Gross volume units
- Free water volume units
- Raw volume units
- Calculated tank temperature units
- Correction for tank temperature units
- Floating roof adjustments volume units
- Temperature for tank calibration units
- Temperature for API units
- Level units
- Volume units

- Temperature for tank shell linear coefficient units

Monitor the PTST Instruction

Monitor from within Logix Designer. The PTST does not have an associated operator faceplate.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Rung-condition-out is cleared to false.
Instruction first run	All internal statuses and calculations are reset. The instruction executes normally.
Rung-condition-in is false	Rung-condition-out is cleared to false. Calculation values is not updated (holds last value).
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	EnableOut is cleared to false.
Instruction first run	All internal statuses and calculations are reset. The instruction executes normally.
Instruction first scan	See Instruction first run in the Function Block Diagram table.
EnableIn is false	EnableOut is cleared to false. Calculation values is not updated (holds last value).
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

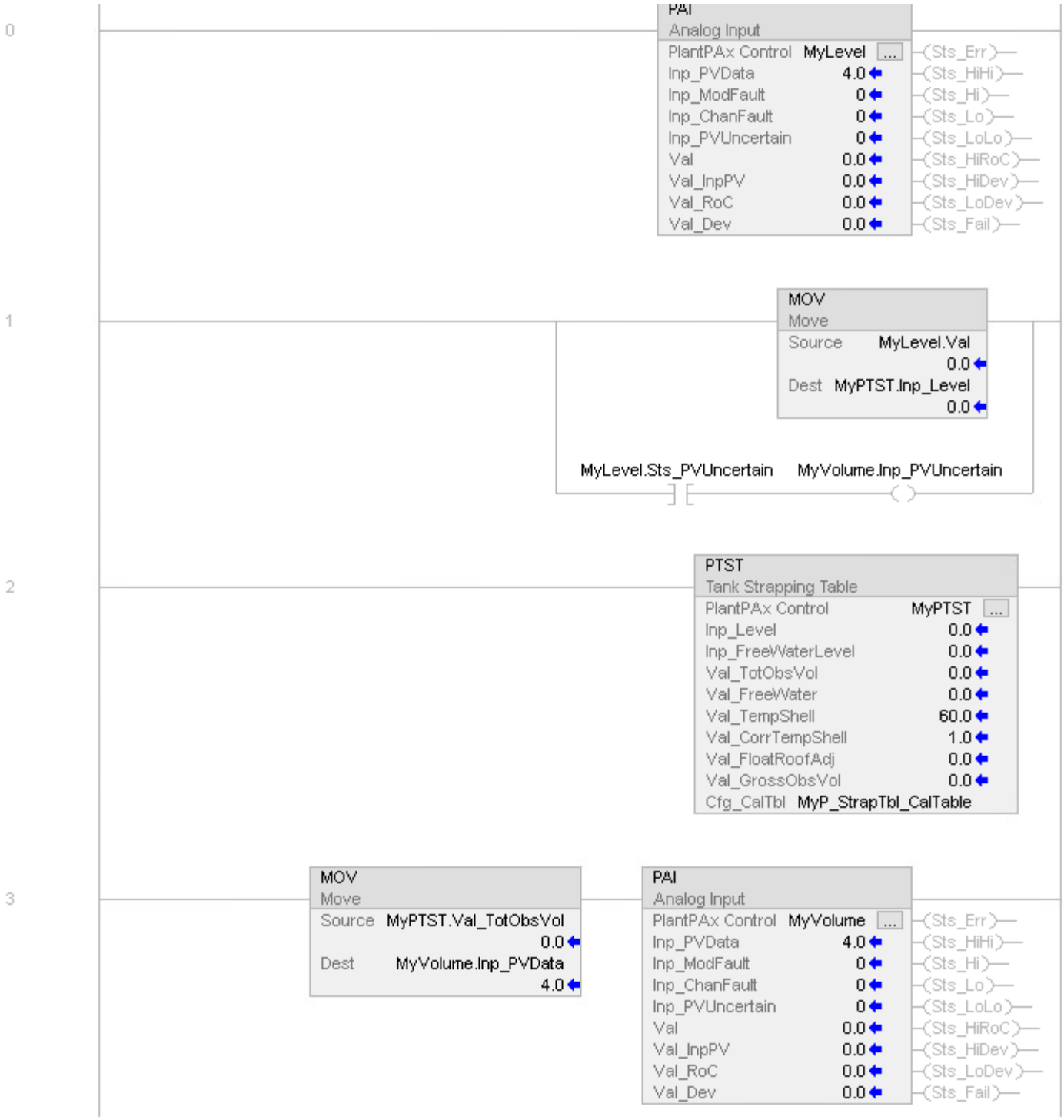
In Structured Text, EnableIn is always true during normal scan. The instruction executes when it is in the control path activated by the logic.

Condition/State	Action Taken
Prescan	See Prescan in the Function Block Diagram table.
Instruction first run	See Instruction first run in the Function Block Diagram table.
EnableIn is true	See EnableIn is true in the Function Block Diagram table.
Postscan	See Postscan in the Function Block Diagram table.

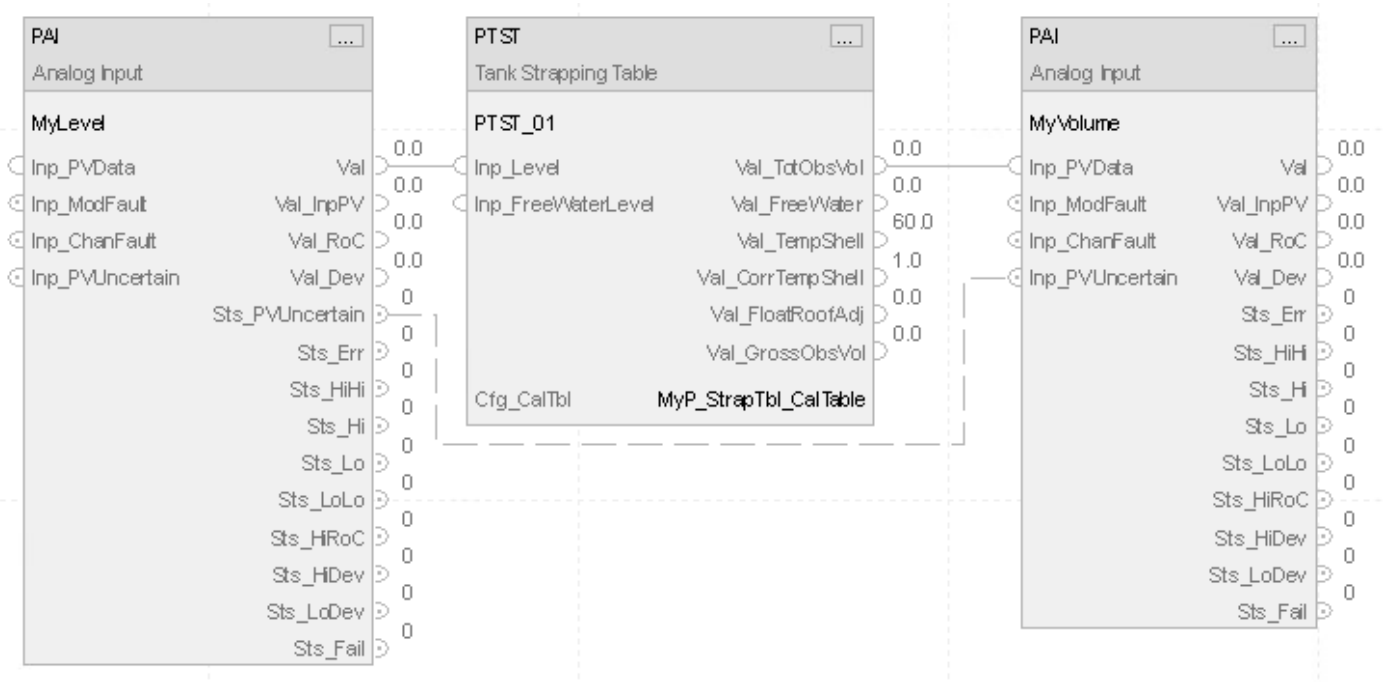
Example

In this example, the PTST instruction calculates the volume of product in a storage tank based on the measured storage tank level and storage tank strapping table information. There is no floating roof and no compensation for displacement. There are no adjustments based on temperature to account for thermal expansion of the tank.

Ladder Diagram



Function Block Diagram



The measured storage tank level connects to the PTST instruction by using the input Inp_Level. The level is in units of feet. The storage tank is four feet tall. In this example, data is in six-inch increments. The strapping table has nine rows:

Level (ft-in.)	Volume (barrels)
0-00	3.1
0-06	136.6
1-00	264.2
1-06	402.7
2-00	541.4
2-06	692.7
3-00	844.1
3-06	990.8
4-00	1137.5

To store the strapping table information in the controller, the tag MyP_StrapTbl_CalTable is created as type P_STRAPPING_TABLE_ROW [9], a nine-element array.

	.Major	.Minor	.Volume
MyP_StrapTbl_CalTable[0]	0	0	3.1
MyP_StrapTbl_CalTable[1]	0	6	136.6
MyP_StrapTbl_CalTable[2]	1	0	264.2
MyP_StrapTbl_CalTable[3]	1	6	402.7
MyP_StrapTbl_CalTable[4]	2	0	541.4
MyP_StrapTbl_CalTable[5]	2	6	692.7
MyP_StrapTbl_CalTable[6]	3	0	844.1
MyP_StrapTbl_CalTable[7]	3	6	990.8
MyP_StrapTbl_CalTable[8]	4	0	1137.5

The InOut tag Cfg_CalTbl of the PTST instruction is modified to point to the new array MyP_StrapTbl_CalTable to provide the instruction with the strapping table information. The output of PTST is then connected to another PAI instruction. The output is the calculated volume of the storage tank.

Structured Text

```
MyPTST.Inp_Level := MyLevel.Val;

MyVolume.Inp_PVUncertain := MyLevel.Sts_PVUncertain;

MyVolume.Inp_PVData := MyPTST.Val_TotObsVol;

PAI(MyLevel);

PTST(MyPTST,MyP_StrapTbl_CalTable);

PAI(MyVolume);
```

See also

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Process Valve (PVLV)

This information applies to the ControlLogix 5380P and 5580P controllers.

The Process Valve (PVLV) instruction operates a two-position, single-solenoid operated valve, a dual-solenoid operated valve, or a motor-operated valve in various modes, monitoring for fault conditions. It also monitors hand-operated two-position valves. It is a built-in analogy of the existing PlantPAx P_ValveSO, P_ValveMO, and P_ValveHO add-on instructions in the Rockwell Automation Library of Process Objects.

The PVLV instruction:

- Allows ownership of the valve through the PCMDSRC instruction.
- Provides for configuration of the de-energized state of the valve: Fail Position 2 (energize to Position 1), Fail Position 1 (energize to Position 2) or Fail Last Position.
- Allows a valve to be set to Position 2 or Position 1. If the valve is so equipped, monitor Position 2/Position 1 limit switch feedback to verify that the valve is Position 2 or Position 1. Whether the valve has each of the feedback limit switches can be configured at the engineer level.

Whether to use each of the feedback limit switches can be configured at the Maintenance level.

- Stops the motion of a Motor-operated Valve. Also provides a Stop Output, which is typically used to break the valve motor seal-in circuit and stop the actuating motor. If the option to allow stopping the valve is enabled, the instruction lets the operator reverse travel. For example, an operator can select Position 2 while closing, which stops the valve, then moves it in the opposite direction.
- Provides an alarm for Full Stall if the valve feedback indicates it did not move off the original position within a configured amount of time when commanded to the other position. Provides an alarm for Transit Stall if the valve feedback indicates the valve moved from the original position but did not reach the target position within a configured amount of time. The Transit Stall or Full Stall condition can optionally de-energize the output to the valve, requiring a reset.
- Provides a limit switch Failure indication if the limit switches indicate the valve is not Position 1, not Position 2, and not moving. Provides a configuration for the failure state: whether both switches are ON or both switches are OFF to indicate limit switch failure.
- Provide for Permissives (those that can be bypassed and those that cannot be bypassed) which are conditions that allow the valve to energize.
  - Permissive to energize (solenoid-operated valve)
  - Position 2 permissives (motor-operated valve)
  - Position 1 permissives (motor-operated valve)
- Provides for Interlocks (those that can be bypassed and those that cannot be bypassed) which are conditions that de-energize the valve and prevent energizing. Provides an alarm when an interlock de-energizes the valve. Allows maintenance personnel to bypass the Permissives and Interlocks.
- Allows maintenance personnel to disable, or force to remain de-energized, the solenoid valve.
- Monitors an I/O Fault input and alarm on an I/O Fault. The I/O Fault condition can de-energize the output to the valve, requiring a reset.
- In Override mode, provides an Override State input that determines whether the Override is to set the valve to Position 2 or Position 1 (default = Position 1).
- Provides a Simulation capability, where the output to the valve remains de-energized, but the instruction can be manipulated as if a working valve were present. The response delay is configurable between a command to Position 2 or Position 1 and the simulated Position 2 or Position 1 response. This same delay is used if the valve is configured with no Position 2/Position 1 feedback. This capability is often used for activities such as instruction testing and operator training.

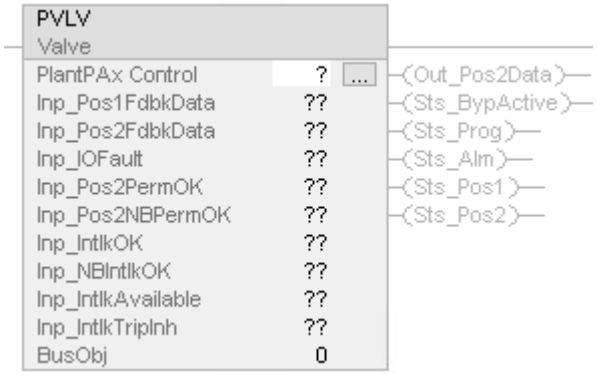
- Provides an output for holding the valve coil energized (to Position 2 or Position 1, based on the configured fail state).
- Provides an actuator fault input for use by valves that generate a fault contact, such as actuator motor overload trip. The actuator fault condition can de-energize the outputs to the valve, requiring a reset.
- Provides the ability to trip the valve (de-energize it or drive it to a default trip position). The program (through program commands) or the operator (through the HMI faceplate) can trip the valve any time.

  The trip function provides these capabilities:

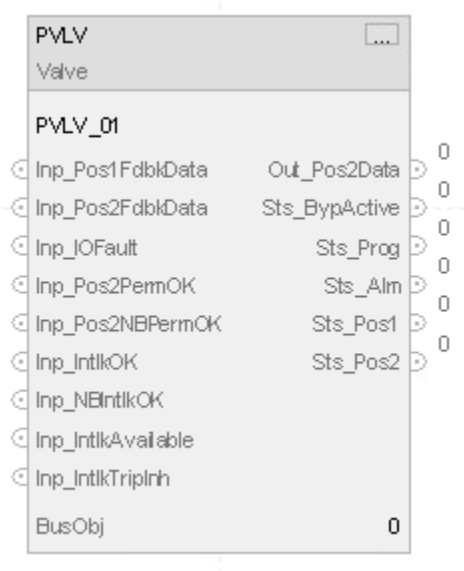
  - Detect failure to reach the configured trip position when tripped and generate an appropriate alarm.
  - Monitor interlock conditions to trip the valve and alarm when an interlock initiates moving the valve to its trip position.
  - Provide for simulation of a working valve while disabling the trip output, for use in off-process training, testing, or simulation.
  - Monitor I/O communication, and alarm and trip if the shed on I/O fault function is enabled on a communication fault.

Available Languages

Ladder Diagram



Function Block Diagram



Structured Text

PVLV(PVLVTag, BusObj);

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PlantPAx ControlPlantPAx Control	P_VALVE_DISCRETE	tag	Data structure required for proper operation of instruction.
BusObj	BUS_OBJ	tag	Bus component

P_VALVE_DISCRETE Structure

Public members are standard, visible tag members that are programmatically accessible. Private, hidden members are used in HMI faceplates and are not

programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
EnableIn	BOOL	Not Visible	Not Required	Input	Enable input. Ladder Diagram: Corresponds to the rung-condition-in. Default is true.
Inp_InitializeReq	BOOL	Not Visible	Not Required	Input	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. use this request when reinitialization is needed. The instruction clears this operand automatically. Default is true.
Inp_OvrdCmd	SINT	Not Visible	Not Required	Input	Override valve command: 0 = None, 1 = Position 1, 2 = Position 2, 3 = Stop, 4 = Pulse Position 1, 5 = Pulse Position 2, 6 = Pulse continuously. Default is 0.
Inp_Pos1FdbkData	BOOL	Visible	Not Required	Input	Feedback from Position 1 limit switch of the device. 1 = Device confirmed Position 1. Default is false.
Inp_Pos2FdbkData	BOOL	Visible	Not Required	Input	Feedback from Position 2 limit switch of the device. 1 = Device confirmed Position 2. Default is false.
Inp_ActuatorFault	BOOL	Not Visible	Not Required	Input	Valve actuator fault (overload, etc.), 0 = Ok, 1 = Fault). Default is false.
Inp_IOFault	BOOL	Visible	Not Required	Input	Indicates the IO data are inaccurate. 0 = The IO data are good, 1 = The IO data are bad, causing fault. If the Valve is not virtual, this input sets Sts_IOFault, which raises IOFault Alarm. Default is false.
Inp_Pos1PermOK	BOOL	Not Visible	Not Required	Input	1 = Position 1 Permissives OK, valve can move to Position 1. Default is true.
Inp_Pos1NBPermOK	BOOL	Not Visible	Not Required	Input	1 = Non-bypassable Position 1 Permissives OK, valve can move to Position 1. Default is true.
Inp_Pos2PermOK	BOOL	Visible	Not Required	Input	1 = Position 2 Permissives OK, valve can move to Position 2. Default is true.
Inp_Pos2NBPermOK	BOOL	Visible	Not Required	Input	1 = Non-bypassable Position 2 Permissives OK, valve can move to Position 2. Default is true.
Inp_IntlkOK	BOOL	Visible	Not Required	Input	1 = Interlocks ok, valve can energize outputs. Default is true.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Inp_NBIntlkOK	BOOL	Visible	Not Required	Input	1 = Non-bypassable interlocks ok, valve can energize outputs. Default is true.
Inp_IntlkAvailable	BOOL	Visible	Not Required	Input	1 = Interlock availibility ok. Default is true.
Inp_IntlkTripInh	BOOL	Visible	Not Required	Input	1 = Inhibit Interlock Trip Status. Default is false.
Inp_RdyReset	BOOL	Not Visible	Not Required	Input	1 = Related object, reset by this valve, is ready to be reset. Default is false.
Inp_Hand	BOOL	Not Visible	Not Required	Input	1 = Acquire hand (typically hardwired local), 0 = Release hand. Default is false.
Inp_Ovrd	BOOL	Not Visible	Not Required	Input	1 = Acquire Override (higher priority Program logic), 0 = Release Override. Default is false.
Inp_OwnerCmd	DINT	Not Visible	Not Required	Input	Owner device command. 0 = None, .10 = Operator Lock, .11 = Operator Unlock, .12 = Program Lock, .13 = Program Unlock, .14 = Acquire Maintenance, .15 = Release Maintenance, .16 = Acquire External, .17 = Release External, .29 = Echo. Default is 0.
Inp_ExtInh	BOOL	Not Visible	Not Required	Input	1 = Inhibit External acquisition, 0 = Allow External acquisition. Default is false.
Inp_HornInh	BOOL	Not Visible	Not Required	Input	1 = Inhibit audible alert, 0 = Alllow audible alert. Default is false.
Inp_Reset	BOOL	Not Visible	Not Required	Input	1 = Reset Shed Latches and Cleared Alarms. Default is false.
Inp_VirtualPos1HO	BOOL	Not Visible	Not Required	Input	1 = Sets virtualized valve HO state to Position 1, 0 = No change. Default is false.
Inp_VirtualPos2HO	BOOL	Not Visible	Not Required	Input	1 = Sets virtualized valve HO state to Position 2, 0 = No change. Default is false.
Cfg_AllowDisable	BOOL	Not Visible	Not Required	Input	1 = Allow Operator alarm disable, 0 = Disallow Operator alarm disable. Default is false.
Cfg_AllowShelve	BOOL	Not Visible	Not Required	Input	1 = Allow Operator alarm shelve, 0 = Disallow Operator alarm shelve. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_eObjType	SINT	Not Visible	Not Required	Input	Valve object type 0 = Solenoid Operated 1 = Motor Operated 2 = Hand Operated. Default is 0.
Cfg_HasPos1Fdbk	BOOL	Not Visible	Not Required	Input	1 = Device provides Position 1 feedback signal. Default is false.
Cfg_HasPos2Fdbk	BOOL	Not Visible	Not Required	Input	1 = Device provides Position 2 feedback signal. Default is false.
Cfg_UsePos1Fdbk	BOOL	Not Visible	Not Required	Input	1 = Use device Position 1 feedback for failure checking. Default is false.
Cfg_UsePos2Fdbk	BOOL	Not Visible	Not Required	Input	1 = Use device Position 2 feedback for failure checking. Default is false.
Cfg_HasPos1PermObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI a valve is connected to Inp_Pos1Perm inputs. Default is false.
Cfg_HasPos2PermObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI a valve is connected to Inp_Pos2Perm inputs. Default is false.
Cfg_HasIntlkObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI a valve is connected to Inp_Intlk inputs. Default is false.
Cfg_FailPos2	BOOL	Not Visible	Not Required	Input	1 = Fail Position 2 (energize to Position 1), 0 = Fail Position 1 (energize to Position 2). Default is false.
Cfg_FdbkFail	BOOL	Not Visible	Not Required	Input	1 = Both feedbacks Position 1/2 are ON is invalid, 0 = Both feedbacks Position 1/2 are OFF is invalid. Default is true.
Cfg_HasStop	BOOL	Not Visible	Not Required	Input	1 = Stop output can be used to break local seal-in and stop valve motion. Default is false.
Cfg_MntnOut	BOOL	Not Visible	Not Required	Input	1 = Maintain Output when requested state reached. Default is false.
Cfg_MntnOutAlm	BOOL	Not Visible	Not Required	Input	1 = Maintain Output when requested state reached is true and when alarm active Default is false.
Cfg_MntnStop	BOOL	Not Visible	Not Required	Input	1 = Maintain Stop Output when stopped state reached. Default is false.
Cfg_HasTrip	BOOL	Not Visible	Not Required	Input	1 = Trip output is connected to valve, 0 = Monitor only. Default is false.
Cfg_TripPos2	BOOL	Not Visible	Not Required	Input	1 = Trip moves valve to Position 2, 0 = Trip moves valve to Position 1. Default is false.
Cfg_HasPulse	BOOL	Not Visible	Not Required	Input	1 = Enable pulsing functions, 0 = Position 1/2 only. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_CompletePulse	BOOL	Not Visible	Not Required	Input	1 = Finish pulse in progress when Commanded to Position 2 or Position 1, 0 = Switch immediately to Position 2 or Position 1 state when Commanded to. Default is false.
Cfg_HasPulseToState	BOOL	Not Visible	Not Required	Input	1 = Enable pulsing functions to state, 0 = Enable pulsing functions to time. Default is false.
Cfg_HasMoreObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI a valve with more info is available. Default is false.
Cfg_HasStatsObj	BOOL	Not Visible	Not Required	Input	1 = Enable stats valve function, 0 = Position 1/2 only. Default is false.
Cfg_CoastToLS	BOOL	Not Visible	Not Required	Input	1 = Coasting into Limit Switch when stopped changes state, 0 = Stop is independent. Default is true.
Cfg_OperPos1Prio	BOOL	Not Visible	Not Required	Input	1 = OCmd_Pos1 has priority, accepted any time, 0 = OCmd_Pos1 openly in Operator and Maintenance command sources. Default is false.
Cfg_OCmdResets	BOOL	Not Visible	Not Required	Input	1 = New Operator state command resets fault, 0 = Reset required to clear fault. Default is false.
Cfg_XCmdResets	BOOL	Not Visible	Not Required	Input	1 = New valve XCmd resets shed latches and cleared alarms, 0 = XCmd_Reset req'd. Default is false.
Cfg_OvrdPermIntlk	BOOL	Not Visible	Not Required	Input	1 = Override ignores bypassable permissives/interlocks, 0 = Always use permissives/interlocks. Default is false.
Cfg_PCmdPos2AsLevel	BOOL	Not Visible	Not Required	Input	1 = PCmd_Pos2 used as a level (1 = Position 2, 0 = Position 1), 0 = The instruction clears PCmd_Pos2 operand automatically. Default is false.
Cfg_ShedOnActuatorFault	BOOL	Not Visible	Not Required	Input	1 = Stop valve and alarm on Actuator fault; 0 = alarm only on Actuator fault. Default is true.
Cfg_ShedOnIOFault	BOOL	Not Visible	Not Required	Input	1 = Stop Valve and Alarm on I/O Fault; 0 = Alarm only on I/O Fault. Default is true.
Cfg_ShedOnFailToTrip	BOOL	Not Visible	Not Required	Input	1 = Continue trip and alarm on Fail to Trip; 0 = Alarm only on Fail to Trip. Default is true.
Cfg_ShedOnFullStall	BOOL	Not Visible	Not Required	Input	1 = Stop valve and alarm on Full Stall; 0 = Alarm only on Full Stall. Default is true.
Cfg_ShedOnLossPos1	BOOL	Not Visible	Not Required	Input	1 = Stop valve and alarm on Loss Position 1; 0 = Alarm only on Loss Position 1 feedback. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_ShedOnLossPos2	BOOL	Not Visible	Not Required	Input	1 = Stop valve and alarm on Loss Position 2; 0 = Alarm only on Loss Position 2 feedback. Default is false.
Cfg_ShedOnTransitStall	BOOL	Not Visible	Not Required	Input	1 = Stop valve and alarm on Transit stall; 0 = Alarm only on Transit stall. Default is true.
Cfg_HasOper	BOOL	Not Visible	Not Required	Input	1 = Operator (unlocked) exists, can be selected. Default is true.
Cfg_HasOperLocked	BOOL	Not Visible	Not Required	Input	1 = Operator locked exists, can be selected. Default is true.
Cfg_HasProg	BOOL	Not Visible	Not Required	Input	1 = Program (unlocked) exists, can be selected. Default is true.
Cfg_HasProgLocked	BOOL	Not Visible	Not Required	Input	1 = Program locked exists, can be selected. Default is true.
Cfg_HasExt	BOOL	Not Visible	Not Required	Input	1 = External exists, can be selected. Default is false.
Cfg_HasMaint	BOOL	Not Visible	Not Required	Input	1 = Maintenance exists, can be selected. Default is true.
Cfg_OvrdOverLock	BOOL	Not Visible	Not Required	Input	1 = Override supersedes Program/Operator lock, 0 = Don't Override lock. Default is true.
Cfg_ExtOverLock	BOOL	Not Visible	Not Required	Input	1 = External supersedes Program/Operator lock, 0 = Don't Override lock. Default is false.
Cfg_ProgPwrUp	BOOL	Not Visible	Not Required	Input	1 = Power up to Program, 0 = Power up to Operator. Default is false.
Cfg_ProgNormal	BOOL	Not Visible	Not Required	Input	Normal source: 1 = Program if no requests; 0 = Operator if no requests. Default is false.
Cfg_PCmdPriority	BOOL	Not Visible	Not Required	Input	Command priority. 1 = Program commands win, 0 = Operator commands win. Default is false.
Cfg_PCmdProgAsLevel	BOOL	Not Visible	Not Required	Input	1 = PCmd_Prog used as a level. Default is false.
Cfg_PCmdLockAsLevel	BOOL	Not Visible	Not Required	Input	1 = PCmd_Lock used as a level (1 = Lock, 0 = Unlock). Default is false.
Cfg_ExtAcqAsLevel	BOOL	Not Visible	Not Required	Input	1 = XCmd_Acq used as level (1 = Acquire, 0 = Release). Default is false.
Cfg_Pos1Dly	REAL	Not Visible	Not Required	Input	Delay before initially activating output Position 1 Valid = 0.0 to 2147483.0 seconds. Default is 2.0.
Cfg_Pos2Dly	REAL	Not Visible	Not Required	Input	Delay before initially activating output Position 2. Valid = 0.0 to 2147483.0 seconds. Default is 2.0.
Cfg_Pos1PulseTime	REAL	Not Visible	Not Required	Input	Output Position 1 time for pulse Position 1 or pulse continuous. Valid = 0.0 to 2147483.0 seconds. Default is 5.0.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_Pos2PulseTime	REAL	Not Visible	Not Required	Input	Output Position 2 time for pulse Position 2 or pulse continuous. Valid = 0.0 to 2147483.0 seconds. Default is 5.0.
Cfg_OutPulseTime	REAL	Not Visible	Not Required	Input	Time to pulse valve outputs (0 = Outputs maintained on). Valid = 0.0 to 2147483.0 seconds. Default is 5.0.
Cfg_StartHornTime	REAL	Not Visible	Not Required	Input	Time to sound audible after command request. (0 = Disabled). Valid = 0.0 to 1000.0 seconds. Default is 0.0.
Cfg_FullStallTime	REAL	Not Visible	Not Required	Input	Time after output Position 1 to get Position 1 feedback before fault. Valid = 0.0 to 2147483.0 seconds. Default is 15.0.
Cfg_TransitStallTime	REAL	Not Visible	Not Required	Input	Time after output Position 1/2 to get Position 1/2 feedback before fault. Valid = 0.0 to 2147483.0 seconds. Default is 60.0.
Cfg_TripFailTime	REAL	Not Visible	Not Required	Input	After tripped, time to reach trip position before alarm. Valid = 0.0 to 2147483.0 seconds. Default is 10.0.
Cfg_VirtualFdbkTime	REAL	Not Visible	Not Required	Input	Delay to echo back of Position 1/2 status when the valve is treated as virtual (seconds). Default is 3.0.
Cfg_CnfrmReqd	SINT	Not Visible	Not Required	Input	Operator Command Confirmation Required. Represents the type of command confirmation required. 0 = None, 1 = Command confirmation required, 2 = Performer e-signature required, 3 = Performer and approver e-signature required. Default is 0.
PCmd_Pos1	BOOL	Not Visible	Not Required	Input	Program command to move valve to Position 1. The Position 1 command is ignored when Cfg_PCmdPosition 2 as level is 1. The instruction clears this operand automatically. Default is false.
PCmd_Pos2	BOOL	Not Visible	Not Required	Input	Program command to move valve to Position 2. The Position 2 command is ignored when Cfg_PCmdPosition 2 as level is 1. The instruction clears this operand automatically. Default is false.
PCmd_Pos1Pulse	BOOL	Not Visible	Not Required	Input	Program command to pulse valve that is in Position 2, to Position 1 once. The instruction clears this operand automatically. Default is false.
PCmd_Pos2Pulse	BOOL	Not Visible	Not Required	Input	Program command to pulse valve that is in Position 1, to Position 2 once. The instruction clears this operand automatically. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
PCmd_ContPulse	BOOL	Not Visible	Not Required	Input	Program command to pulse valve continuously. The instruction clears this operand automatically. Default is false.
PCmd_Trip	BOOL	Not Visible	Not Required	Input	Program command to trip valve. Default is false.
PCmd_Reset	BOOL	Not Visible	Not Required	Input	Program command to reset all alarms requiring reset. The instruction clears this operand automatically. Default is false.
PCmd_Stop	BOOL	Not Visible	Not Required	Input	Program command to stop valve. The instruction clears this operand automatically. Default is false.
PSet_Owner	DINT	Not Visible	Not Required	Input	Program owner request ID (non-zero) or release (zero). Default is 0.
PCmd_Lock	BOOL	Not Visible	Not Required	Input	Program command to lock Program (disallow Operator). The instruction clears this operand automatically if Cfg_PCmdLockAsLevel = 0. Default is false.
PCmd_Normal	BOOL	Not Visible	Not Required	Input	Program command to select normal command source (Operator or Program). The instruction clears this operand automatically. Default is false.
PCmd_Oper	BOOL	Not Visible	Not Required	Input	Program command to select Operator (Program to Operator). The instruction clears this operand automatically. Default is false.
PCmd_Prog	BOOL	Not Visible	Not Required	Input	Program command to select Program (Operator to Program). The instruction clears this operand automatically. Default is false.
PCmd_Unlock	BOOL	Not Visible	Not Required	Input	Program command to unlock Program (allow Operator to acquire). The instruction clears this operand automatically. Default is false.
PCmd_Virtual	BOOL	Not Visible	Not Required	Input	Program Command to select Virtual (simulated) device operation Default is false.
PCmd_Physical	BOOL	Not Visible	Not Required	Input	Program Command to select Physical device operation (not simulated) Default is false.
XCmd_Pos1	BOOL	Not Visible	Not Required	Input	External command to turn object Position 1. The instruction clears this operand automatically. Default is false.
XCmd_Pos2	BOOL	Not Visible	Not Required	Input	External command to turn object Position 2. The instruction clears this operand automatically. Default is false.
XCmd_Pos1Pulse	BOOL	Not Visible	Not Required	Input	External command to pulse valve that is in Position 2, to Position 1 once. The instruction clears this operand automatically. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
XCmd_Pos2Pulse	BOOL	Not Visible	Not Required	Input	External command to pulse valve that is in Position 1, to Position 2 once. The instruction clears this operand automatically. Default is false.
XCmd_ContPulse	BOOL	Not Visible	Not Required	Input	External command to pulse object continuously (blink). The instruction clears this operand automatically. Default is false.
XCmd_Trip	BOOL	Not Visible	Not Required	Input	External command to trip valve. Default is false.
XCmd_Reset	BOOL	Not Visible	Not Required	Input	External command to clear shed latches and cleared alarms. Default is false.
XCmd_Stop	BOOL	Not Visible	Not Required	Input	External command to stop valve motion.the instruction clears this operand automatically. Default is false.
XCmd_ResetAckAll	BOOL	Not Visible	Not Required	Input	External command to acknowledge and reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.
XCmd_Acq	BOOL	Not Visible	Not Required	Input	External command to acquire ownership (Operator/Program/Override/Maintenance to External). The instruction clears this operand automatically if Cfg_ExtAcqAsLevel = 0. Default is false.
XCmd_Rel	BOOL	Not Visible	Not Required	Input	External command to release ownership if Cfg_ExtAcqAsLevel = 0 (External to Operator/Program/Override/Maintenance). The instruction clears this operand automatically. Default is false.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
EnableOut	BOOL	Not Visible	Not Required	Output	Enable output. This output state always reflects EnableIn input state.
Out_Pos1Data	BOOL	Not Visible	Not Required	Output	1 = Activate to move valve to Position 1.
Out_Pos2Data	BOOL	Visible	Not Required	Output	1 = Activate to move valve to Position 2.
Out_StopData	BOOL	Not Visible	Not Required	Output	1 = Break seal-in circuit in actuator to stop valve motion.
Out_TripData	BOOL	Not Visible	Not Required	Output	1 = Trip valve to safe/fail state.
Out_HornData	BOOL	Not Visible	Not Required	Output	1 = Sound audible prior to commanded valve start.
Out_Reset	BOOL	Not Visible	Not Required	Output	1 = Reset command has been received and accepted.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Out_OwnerSts	DINT	Not Visible	Not Required	Output	Status of command source, owner command handshake and ready status. 0 = None, .10 = Operator Lock, .11 = Operator Unlock, .12 = Program Lock, .13 = Program Unlock, .14 = Acquire Maintenance, .15 = Release Maintenance, .16 = Acquire External, .17 = Release External, .18 = Has Maintenance, .19 = External Override Lock, .20 = Has External, .21 = Has Operator, .22 = Has Operator Locked, .23 = Has Program, .24 = Has Program Locked, .29 = Echo, .30 = Not Ready.
Sts_Initialized	BOOL	Not Visible	Not Required	Output	1 = Instruction is initialized. use Inp_initializereq to reinitialize.
Sts_Available	BOOL	Not Visible	Not Required	Output	1 = Valve available for control by automation (Program).
Sts_IntlkAvailable	BOOL	Not Visible	Not Required	Output	1 = Valve can be acquired by Program and is available for start/stop control when interlocks are OK.
Sts_Bypass	BOOL	Not Visible	Not Required	Output	1 = Bypassable interlocks are bypassed.
Sts_BypActive	BOOL	Visible	Not Required	Output	1 = Interlock bypassing active (bypassed or Maintenance).
Sts_MaintByp	BOOL	Not Visible	Not Required	Output	1 = Valve has a Maintenance bypass functiPosition 2 active.
Sts_NotRdy	BOOL	Not Visible	Not Required	Output	1 = Valve is not ready, for HMI use hidden detail bits (Sts_nrdyxxx) for reason.
Sts_NrdyCfgErr	BOOL	Not Visible	Not Required	Output	1 = Valve is not ready: configuration error.
Sts_NrdyIntlk	BOOL	Not Visible	Not Required	Output	1 = Valve is not ready: interlock not ok.
Sts_NrdyOoS	BOOL	Not Visible	Not Required	Output	1 = Valve is not ready: object disabled by Maintenance.
Sts_NrdyPos1Perm	BOOL	Not Visible	Not Required	Output	1 = Valve is not ready: Position 1 permissive not ok.
Sts_NrdyPos2Perm	BOOL	Not Visible	Not Required	Output	1 = Valve is not ready: Position 2 permissive not ok.
Sts_NrdyPerm	BOOL	Not Visible	Not Required	Output	1 = Valve is not ready: permissive not ok.
Sts_NrdyStopPerm	BOOL	Not Visible	Not Required	Output	1 = Valve is not ready: stop permissive not ok.
Sts_Err	BOOL	Not Visible	Not Required	Output	1 = Error in configuration: See detail bits (Sts_Errxxx) for reason.
Sts_ErrAlm	BOOL	Not Visible	Not Required	Output	1 = Error in logix tag-based alarm settings.
Sts_ErrFullStallTime	BOOL	Not Visible	Not Required	Output	1 = Invalid Position 1 fail timer preset (use 0.0 to 2147483.0).
Sts_ErrHas	BOOL	Not Visible	Not Required	Output	1 = Error in Config: must have at least one Limit Switch

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_ErrOutPulseTime	BOOL	Not Visible	Not Required	Output	1 = Invalid outpulse timer preset (use 0.0 to 2147483.0).
Sts_ErrPos1Dly	BOOL	Not Visible	Not Required	Output	1 = Invalid Position 1 delay timer preset (use 0 to 2147483.0).
Sts_ErrPos1PulseTime	BOOL	Not Visible	Not Required	Output	1 = Invalid Position 1 pulse timer preset (use 0.0 to 2147483.0).
Sts_ErrPos2Dly	BOOL	Not Visible	Not Required	Output	1 = Invalid Position 2 delay timer preset (use 0 to 2147483.0).
Sts_ErrPos2PulseTime	BOOL	Not Visible	Not Required	Output	1 = Invalid Position 2 pulse timer preset (use 0.0 to 2147483.0).
Sts_ErrTransitStallTime	BOOL	Not Visible	Not Required	Output	1 = Invalid Position 2 fail timer preset (use 0.0 to 2147483.0).
Sts_ErrTripFailTime	BOOL	Not Visible	Not Required	Output	1 = Invalid virtual feedback timer (use 0.0 to 2147483.0).
Sts_Hand	BOOL	Not Visible	Not Required	Output	1 = Hand is selected (supersedes OoS, Maintenance, Override, External, Program, Operator).
Sts_OoS	BOOL	Not Visible	Not Required	Output	1 = Out of service is selected (supersedes Maintenance, Override, External, Program, Operator).
Sts_Maint	BOOL	Not Visible	Not Required	Output	1 = Maintenance is selected (supersedes Override, External, Program, Operator).
Sts_Ovrd	BOOL	Not Visible	Not Required	Output	1 = Override is selected (supersedes External, Program, Operator).
Sts_Ext	BOOL	Not Visible	Not Required	Output	1 = External is selected (supersedes Program and Operator).
Sts_Prog	BOOL	Visible	Not Required	Output	1 = Program is selected.
Sts_ProgLocked	BOOL	Not Visible	Not Required	Output	1 = Program is selected and locked.
Sts_Oper	BOOL	Not Visible	Not Required	Output	1 = Operator is selected.
Sts_OperLocked	BOOL	Not Visible	Not Required	Output	1 = Operator is selected and locked.
Sts_ProgOperSel	BOOL	Not Visible	Not Required	Output	Program/Operator selection (latch) state: 1 = Program, 0 = Operator.
Sts_ProgOperLock	BOOL	Not Visible	Not Required	Output	Program/Operator lock (latch) state, 1 = Locked, 0 = Unlocked.
Sts_Normal	BOOL	Not Visible	Not Required	Output	1 = Selection equals the normal (Program or Operator).
Sts_ExtReqInh	BOOL	Not Visible	Not Required	Output	1 = External request inhibited, cannot get to External from current state.
Sts_ProgReqInh	BOOL	Not Visible	Not Required	Output	1 = Program request inhibited, cannot get to Program from current state.
Sts_MAcqRcvd	BOOL	Not Visible	Not Required	Output	1 = Maintenance acquire command received this scan.
Sts_CmdConflict	BOOL	Not Visible	Not Required	Output	1 = Conflicting commands received this scan.
Sts_Alm	BOOL	Visible	Not Required	Output	1 = An alarm is active.
Sts_AlmInh	BOOL	Not Visible	Not Required	Output	1 = One or more alarms shelved, disabled or suppressed.
Sts_ActuatorFault	BOOL	Not Visible	Not Required	Output	1 = Actuator fault detected (input).
Sts_CmdToPos1	BOOL	Not Visible	Not Required	Output	1 = Valve commanded to Position 1, has not yet moved off Position 1 Limit Switch.
Sts_CmdToPos2	BOOL	Not Visible	Not Required	Output	1 = Valve commanded to Position 2, has not yet moved off Position 2 Limit Switch.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_ErrUse	BOOL	Not Visible	Not Required	Output	1 = Error in Config: must use at least one Limit Switch
Sts_ErrVirtualFdbkTime	BOOL	Not Visible	Not Required	Output	1 = Invalid virtual feedback timer (use 0.0 to 2147483.0).
Sts_FdbkPos1	BOOL	Not Visible	Not Required	Output	1 = Valve feedback shows valve in Position 1 state.
Sts_FdbkPos2	BOOL	Not Visible	Not Required	Output	1 = Valve feedback shows valve in Position 2 state.
Sts_FdbkFail	BOOL	Not Visible	Not Required	Output	1 = Feedbacks are in an invalid state (not Position 1, Position 2, or transition).
Sts_FullStall	BOOL	Not Visible	Not Required	Output	1 = Valve full stall (failed to move at all) (one-shot).
Sts_Horn	BOOL	Not Visible	Not Required	Output	1 = Valve horn active.
Sts_LossPos1	BOOL	Not Visible	Not Required	Output	1 = Loss of Position 1 position alarm.
Sts_LossPos2	BOOL	Not Visible	Not Required	Output	1 = Loss of Position 2 position alarm.
Sts_IOFault	BOOL	Not Visible	Not Required	Output	I/O comm fault status (0 = Ok, 1 = Bad).
Sts_Moving	BOOL	Not Visible	Not Required	Output	1 = Valve not requested to trip and is not confirmed Position 1 or Position 2.
Sts_MovingToPos1	BOOL	Not Visible	Not Required	Output	1 = Valve requested to Position 1 and awaiting Position 1 feedback.
Sts_MovingToPos2	BOOL	Not Visible	Not Required	Output	1 = Valve requested to Position 2 and awaiting Position 2 feedback.
Sts_Pos1	BOOL	Visible	Not Required	Output	1 = Valve requested to Position 1 and is confirmed Position 1.
Sts_Pos2	BOOL	Visible	Not Required	Output	1 = Valve requested to Position 2 and is confirmed Position 2.
Sts_Pulsing	BOOL	Not Visible	Not Required	Output	1 = Output is in a pulsing sequence.
Sts_Stopped	BOOL	Not Visible	Not Required	Output	1 = Valve requested to stop and is not at either end of travel.
Sts_TransitStall	BOOL	Not Visible	Not Required	Output	1 = Valve transit stall (did not reach target position) (one-shot).
Sts_TripFail	BOOL	Not Visible	Not Required	Output	1 = Valve failed to trip (did not reach trip position).
Sts_Tripping	BOOL	Not Visible	Not Required	Output	1 = Valve requested to trip and has not reached trip position.
Sts_UnackAlmCount	DINT	Not Visible	Not Required	Output	Count of unacknowledged alarms.
Sts_NrdyTrip	BOOL	Not Visible	Not Required	Output	1 = Valve not ready: tripped (at object or by command).
Sts_NrdyIOFault	BOOL	Not Visible	Not Required	Output	1 = Valve is not ready: io fault (shed requires reset).
Sts_NrdyActuatorFault	BOOL	Not Visible	Not Required	Output	1 = Valve not ready:actuator fault (fault or shed requires reset).
Sts_IntlkTrip	BOOL	Not Visible	Not Required	Output	1 = Valve tripped by an interlock not ok.
Sts_NrdyFail	BOOL	Not Visible	Not Required	Output	1 = Valve is not ready: object failure (shed requires reset).
Sts_RdyAck	BOOL	Not Visible	Not Required	Output	1 = An alarm is ready to be acknowledged.
Sts_RdyReset	BOOL	Not Visible	Not Required	Output	1 = A latched alarm or shed condition is ready to be reset.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_Virtual	BOOL	Not Visible	Not Required	Output	1 = The instruction treats the object as virtual. The instruction acts as normal but the output is kept de-energized (Out_ = 0). 0 = the instruction operates the object normally.
Sts_bSrc	INT	Not Visible	Not Required	Output	Active selection bitmap (for HMI totem pole with command source request selection): Sts_bSrc.0: hand, Sts_bSrc.1: Programmed out of service (rung false), Sts_bSrc.2: Maintenance out of service, Sts_bSrc.3: Maintenance, Sts_bSrc.4: Override, Sts_bSrc.5: External, Sts_bSrc.6: Program locked, Sts_bSrc.7: Program, Sts_bSrc.8: Operator locked, Sts_bSrc.9: Operator.
Sts_eSrc	INT	Not Visible	Not Required	Output	The current command source enumerated values: 0 = Logic not in use, 4 = Hand, 8 = Maintenance, 16 = Override, 32 = Program, 33 = Program locked, 34 = Program by default (Normal), 64 = Operator, 65 = Operator locked, 66 = Operator by default (Normal), 128 = Maintenance Out of Service, 129 = Programmed Out of Service (rung false), 256 = External.
Sts_eCmd	SINT	Not Visible	Not Required	Output	Valve command: 0=None, 1=Position 1, 2=Position 2, 4=Stop, 8=Pulse Position 1, 16=Pulse Position 2, 32=Pulse continuously, 64=Trip
Sts_eFdbk	SINT	Not Visible	Not Required	Output	Valve feedback: 0 = transition, 1 = Position 1, 2 = Position 2, 3 = invalid.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_eFault	SINT	Not Visible	Not Required	Output	Valve fault status: 0 =None, 2 = feedback fault, 7 = transitstall, 8 = fullstall, 11 = lossPosition 1, 12 = lossPosition 2, 13 = Actualtor fault, 14 = trip fail, 15 = Interlock trip, 16 - Not ready trip, 32 = I/O Fault, 34 = Config Error.
Sts_eNotify	SINT	Not Visible	Not Required	Output	All alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyAll	SINT	Not Visible	Not Required	Output	All alarm status enumerated values including related valves: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_eState	DINT	Not Visible	Not Required	Output	Internal logic state 1 = Position 1 2 = Position 2 9 = Position 2 command ON-DELAY 10 = Position 1 command OFF-DELAY 17 = PULSE Position 2 command ON-DELAY 18 = PULSE Position 1 command OFF-DELAY 33 = PULSE Position 2 command ON PULSE time 34 = PULSE Position 1 command OFF PULSE time 65 = PULSE CONTINUOUS command ON-DELAY 66 = PULSE CONTINUOUS command OFF-DELAY 129 = PULSE CONTINUOUS command OFF PULSE time 130 = PULSE CONTINUOUS command ON PULSE time 257 = Position 1 PULSE completion time
Sts_eSts	SINT	Not Visible	Not Required	Output	Valve status: 0 = ?, 1 = Position 1, 2 = Position 2, 3 = cmd to Position 1, 4 = cmd to Position 2, 5 = moving to Position 1, 6 = moving to Position 2, 7 = stopped, 8 = tripping, 9 = pulse Position 1, 10 = pulse Position 2, 11 = pulse continuously, 12 = moving, 14 = Horn, 15 = out of service.
Sts_eNotifyActuatorFault	SINT	Not Visible	Not Required	Output	Fail alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_eNotifyFullStall	SINT	Not Visible	Not Required	Output	Full Stall alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyIntlkTrip	SINT	Not Visible	Not Required	Output	Interlock Trip alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyIOFault	SINT	Not Visible	Not Required	Output	IO Fault alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyLossPos1	SINT	Not Visible	Not Required	Output	Loss Position 1 alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_eNotifyLossPos2	SINT	Not Visible	Not Required	Output	Loss Position 2 alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyTransitStall	SINT	Not Visible	Not Required	Output	Transit Stall alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyTripFail	SINT	Not Visible	Not Required	Output	Trip Fail alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
SrcQ	SINT	Not Visible	Not Required	Output	Source and quality of primary Val/Sts (enumeration).
SrcQ_IO	SINT	Not Visible	Not Required	Output	Source and quality of primary I/O (enumeration).
Val_Owner	DINT	Not Visible	Not Required	Output	Current object owner ID (0 = Not owned).
XRdy_Acq	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Acq, enable HMI button.
XRdy_Rel	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Rel, enable HMI button.
XRdy_Reset	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Reset, enable HMI button.
XRdy_ResetAckAll	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Resetackall, enable HMI button.

Private Input Members	Data Type	Description
HMI_BusObjIndex	DINT	HMI bus object index Default is 0.

Private Input Members	Data Type	Description
MCmd_Acq	BOOL	Maintenance command to acquire ownership (Operator/Program/External/Override to Maintenance). The instruction clears this operand automatically. Default is false.
MCmd_Bypass	BOOL	Maintenance command to bypass all bypassable interlocks. The instruction clears this operand automatically. Default is false.
MCmd_Check	BOOL	Maintenance command to check (not bypass) all interlocks. The instruction clears this operand automatically. Default is false.
MCmd_IS	BOOL	Maintenance command to select in service. The instruction clears this operand automatically. Default is false.
MCmd_OoS	BOOL	Maintenance command to select out of service. The instruction clears this operand automatically. Default is false.
MCmd_Physical	BOOL	Maintenance Command to select Physical device operation (not simulated) Default is false.
MCmd_Rel	BOOL	Maintenance command to release ownership (Maintenance to Operator/Program/External/Override). The instruction clears this operand automatically. Default is false.
MCmd_Virtual	BOOL	Maintenance Command to select Virtual (simulated) device operation Default is false.
OCmd_ContPulse	BOOL	Operator command to pulse valve continuously. The instruction clears this operand automatically. Default is false.
OCmd_Lock	BOOL	Operator command to lock Operator (disallow Program). The instruction clears this operand automatically. Default is false.
OCmd_Normal	BOOL	Operator command to select normal (Operator or Program). The instruction clears this operand automatically. Default is false.
OCmd_Oper	BOOL	Operator command to select Operator (Program to Operator). The instruction clears this operand automatically. Default is false.
OCmd_Pos1	BOOL	Operator command to move valve to Position 1. The instruction clears this operand automatically. Default is false.
OCmd_Pos1Pulse	BOOL	Operator command to pulse valve that is in Position 2, to Position 1 once. The instruction clears this operand automatically. Default is false.
OCmd_Pos2	BOOL	Operator command to move valve to Position 2. The instruction clears this operand automatically. Default is false.
OCmd_Pos2Pulse	BOOL	Operator command to pulse valve that is in Position 1, to Position 2 once. The instruction clears this operand automatically. Default is false.
OCmd_Prog	BOOL	Operator command to select Program (Operator to Program). The instruction clears this operand automatically. Default is false.

Private Input Members	Data Type	Description
OCmd_Reset	BOOL	Operator command to reset all alarms requiring reset. The instruction clears this operand automatically. Default is false.
OCmd_ResetAckAll	BOOL	Operator command to reset all alarms and latched shed conditions. The use of OCmd_Resetackall is restricted to HMI. The instruction clears this operand automatically. Default is false.
OCmd_Stop	BOOL	Operator command to stop valve motion. The instruction clears this operand automatically. Default is false.
OCmd_Trip	BOOL	Operator command to trip valve. Default is false.
OCmd_Unlock	BOOL	Operator command to unlock / release (allow Program to acquire) ownership. The instruction clears this operand automatically. Default is false.

Private Output Members	Data Type	Description
MRdy_Acq	BOOL	1 = Ready for MCmd_Acq, enable HMI button.
MRdy_Bypass	BOOL	1 = Ready to receive OCmd_Bypass, enable HMI button.
MRdy_Check	BOOL	1 = Ready to receive OCmd_Check, enable HMI button.
MRdy_IS	BOOL	1 = Ready for MCmd_IS, enable HMI button.
MRdy_OoS	BOOL	1 = Ready for MCmd_OoS, enable HMI button.
MRdy_Physical	BOOL	1 = Ready for MCmd_Physical, enable HMI button.
MRdy_Rel	BOOL	1 = Ready for MCmd_Rel, enable HMI button.
MRdy_Virtual	BOOL	1 = Ready for MCmd_Virtual, enable HMI button.
ORdy_ContPulse	BOOL	1 = Ready to receive OCmd_Contpulse, enable HMI button.
ORdy_Lock	BOOL	1 = Ready for OCmd_Lock, enable HMI button.
ORdy_Normal	BOOL	1 = Ready for OCmd_Normal, enable HMI button.
ORdy_Oper	BOOL	1 = Ready for OCmd_Oper, enable HMI button.
ORdy_Pos1	BOOL	1 = Ready to receive OCmd_Pos1, enable HMI button.
ORdy_Pos1Pulse	BOOL	1 = Ready to receive OCmd_Pos1pulse, enable HMI button.
ORdy_Pos2	BOOL	1 = Ready to receive OCmd_Pos2, enable HMI button.
ORdy_Pos2Pulse	BOOL	1 = Ready to receive OCmd_Pos2pulse, enable HMI button.
ORdy_Prog	BOOL	1 = Ready for OCmd_Prog, enable HMI button.
ORdy_Reset	BOOL	1 = At least one alarm or shed condition requires reset.
ORdy_ResetAckAll	BOOL	1 = At least one alarm or latched shed condition requires reset or acknowledgement.
ORdy_Stop	BOOL	1 = Ready for OCmd_Stop (enables HMI button).
ORdy_Trip	BOOL	1 = Ready for OCmd_trip (enables HMI button).
ORdy_Unlock	BOOL	1 = Ready for OCmd_Unlock, enable HMI button.

Public InOut Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
BusObj	BUS_OBJ	Visible	Required	InOut	Bus component

BUS_OBJ Structure

The BUS_OBJ structure is used to link the valve to other devices and instructions in a complex control strategy, typically into a hierarchy. A Bus

Object rolls up status and alarm information from lower level devices to higher level control and fans out commands from higher level control to lower level devices. Items link to the bus by referencing a single member of the BUS_OBJ array associated with the bus.

This parameter links the instruction to an external tag that contains necessary data for the instruction to operate. The external tag must be of the data type shown, or may be NULL. If NULL, the Bus functions of this instruction are not available.

Members	Data Type	Description
Inp_Cmd	DINT	Input to assert commands
Out_Cmd	DINT	Resultant commands
Inp_CmdLLH	DINT	Input for level normally high commands
Out_CmdLLH	DINT	Resultant line level high commands
Inp_Sts	DINT	Input to assert status
Out_Sts	DINT	Resultant status
Inp_CmdAck	DINT	Input to assert a command acknowledgement
Out_CmdAck	DINT	Resultant command acknowledgements
Inp_SeverityMax	DINT	Input: maximum alarm severity
Out_SeverityMax	DINT	Resultant of maximum alarm severity
Cfg_CmdMask	DINT	Propagation mask for commands
Cfg_CmdLLHMask	DINT	Propagation mask for line level high commands
Cfg_StsMask	DINT	Propagation mask for status
Ref_Index	DINT	Bus array index

Alarms

Discrete Logix tag-based alarms are defined for these members:

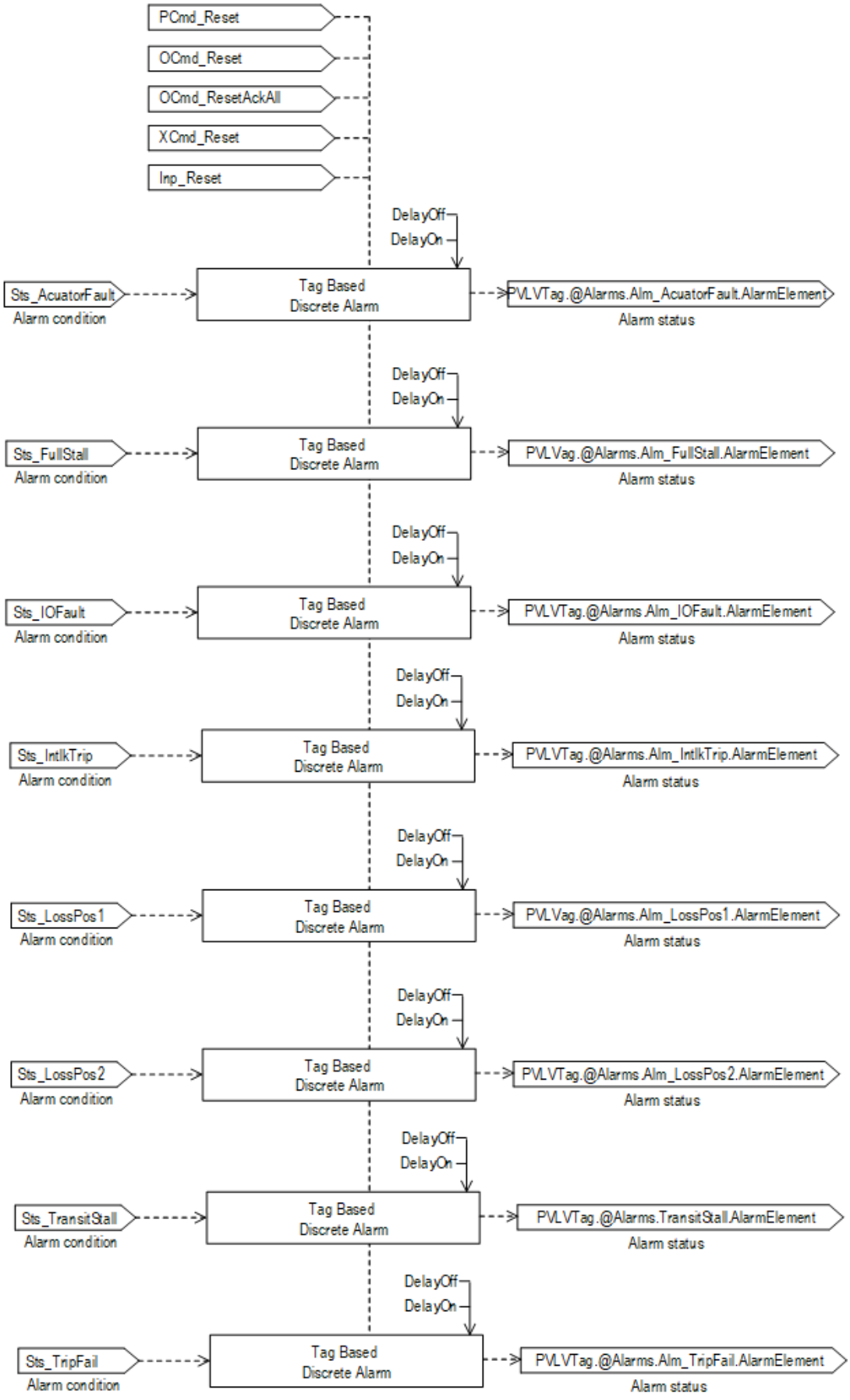
Member	Alarm Name	Description
Sts_AcuatorFault	Alm_AcuatorFault	Raised if the Inp_ActuatorFault input is true. This alarm is provided for use by valves that generate a fault contact, such as actuator motor overload trip.
Sts_FullStall	Alm_FullStall	Raised when the valve has and is using Position 2 and/or Position 1 limit switches, an attempt is made to Position 2 or Position 1 the valve, and the limit switches indicate that the valve did not move from its original position at all within the configured time.
Sts_IOFault	Alm_IOFault	Raised when the Inp_IOFault input is true. This input is usually used to indicate to the instruction that a communication failure has occurred for its I/O. If the I/O Fault is configured as a shed fault, the valve is commanded to Stop motion and cannot be commanded to either position until reset.
Sts_IntlkTrip	Alm_IntlkTrip	Raised when the valve is energized and an interlock Not-OK condition causes the valve to be de-energized. If interlocks are not bypassed, a bypassable interlock or a non-bypassable interlock Not-OK condition initiates an interlock trip. If interlocks are bypassed, only a non-bypassable interlock Not-OK condition initiates an interlock trip.
Sts_LossPos1	Alm_LossPos1	Raised when the valve has reached the confirmed Position 1 state and the Position 1 confirmation feedback is lost.
Sts_LossPos2	Alm_LossPos2	Raised when the valve has reached the confirmed Position 2 state and the Position 2 confirmation feedback is lost.

Member	Alarm Name	Description
Sts_TransitStall	Alm_TransitStall	Raised when the valve has and is using both Position 2 and Position 1 position feedback, an attempt is made to Position 2 or Position 1 the valve, and the position feedback indicates that the valve moved off the original position but did not reach the target position within the configured transit stall time.
Sts_TripFail	Alm_TripFail	Raised when the valve has and is using the optional trip feature, an attempt is made to trip the valve, and the limit switch feedbacks show that the valve did not reach the configured tripped position (Position 2 or Position 1) within the configured fail to trip time.

Mark the alarm as used or unused and set standard configuration members of the discrete Logix tag-based alarm. Use this format to access alarm elements:

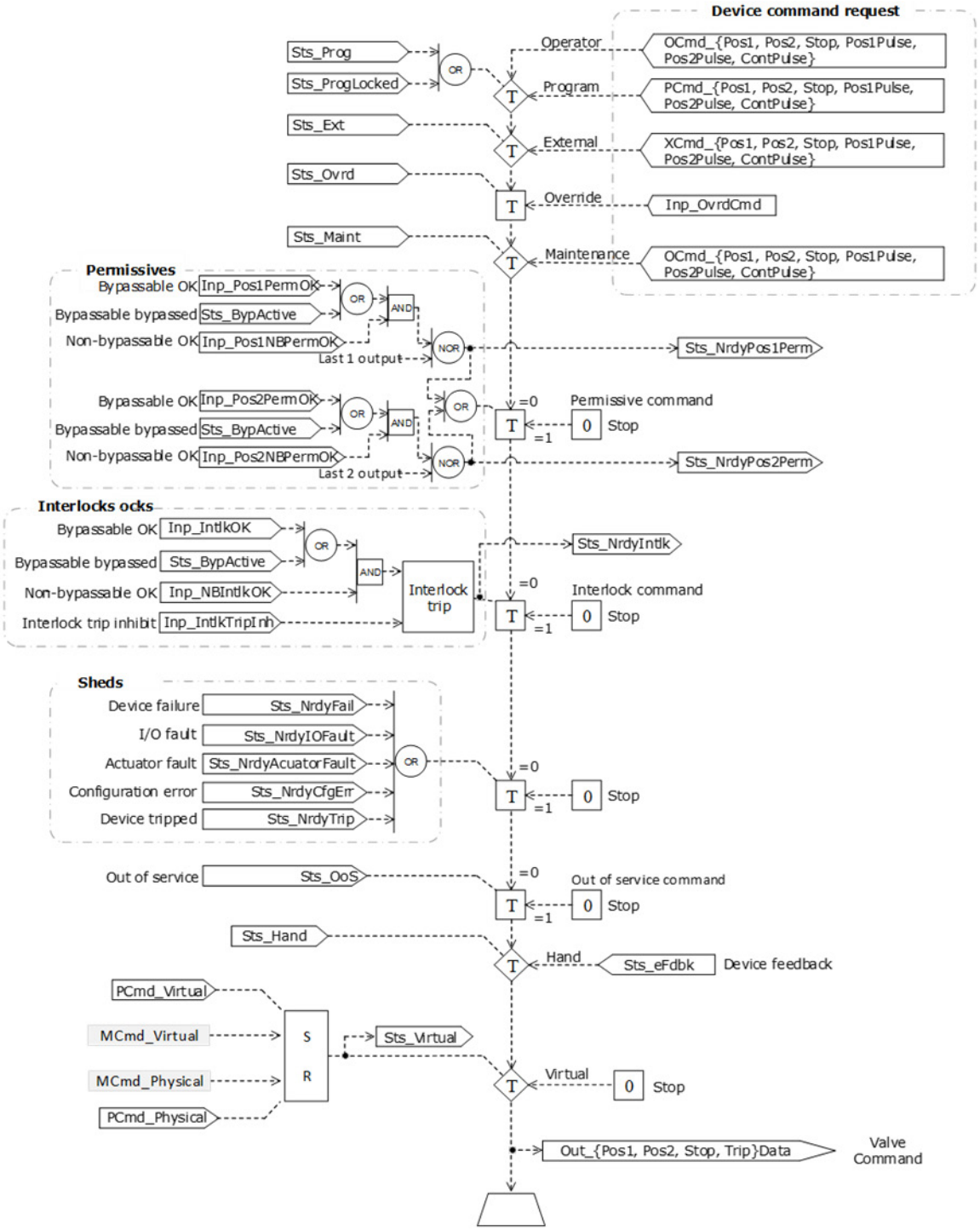
Tag.@Alarms.AlarmName.AlarmElement

Program, Operator, and External commands reset latched alarms, and reset and acknowledge all alarms of the instruction (Alarm Set) at the same time. This diagram shows how the commands interact with the PVLV instruction.


PCmd_Reset
OCmd_Reset
OCmd_ResetAckAll
XCmd_Reset
Inp_Reset
DelayOff
DelayOn
Sts_AcuatorFault
Alarm condition
Tag Based
Discrete Alarm
PVLVTag.@Alarms.Alm_AcuatorFault.AlarmElement
Alarm status
DelayOff
DelayOn
Sts_FullStall
Alarm condition
Tag Based
Discrete Alarm
PVLVag.@Alarms.Alm_FullStall.AlarmElement
Alarm status
DelayOff
DelayOn
Sts_IOFault
Alarm condition
Tag Based
Discrete Alarm
PVLVTag.@Alarms.Alm_IOFault.AlarmElement
Alarm status
DelayOff
DelayOn
Sts_IntlkTrip
Alarm condition
Tag Based
Discrete Alarm
PVLVTag.@Alarms.Alm_IntlkTrip.AlarmElement
Alarm status
DelayOff
DelayOn
Sts_LossPos1
Alarm condition
Tag Based
Discrete Alarm
PVLVag.@Alarms.Alm_LossPos1.AlarmElement
Alarm status
DelayOff
DelayOn
Sts_LossPos2
Alarm condition
Tag Based
Discrete Alarm
PVLVTag.@Alarms.Alm_LossPos2.AlarmElement
Alarm status
DelayOff
DelayOn
Sts_TransitStall
Alarm condition
Tag Based
Discrete Alarm
PVLVTag.@Alarms.TransitStall.AlarmElement
Alarm status
DelayOff
DelayOn
Sts_TripFail
Alarm condition
Tag Based
Discrete Alarm
PVLVTag.@Alarms.Alm_TripFail.AlarmElement
Alarm status

Operation

This diagram illustrates the functionality of the PVLV instruction:



Operator command request confirmation

The PVLV instruction enables these operator commands and settings:

- OCmd_ContPulse

- OCmd_Pos1
- OCmd_Pos1Pulse
- OCmd_Pos2
- OCmd_Pos2Pulse
- OCmd_Stop
- OCmd_Trip

Enforced security might require the request to be confirmed or canceled before the selected command executes or setting is accepted. The instruction checks the security rules inspecting Cfg_CnfrmReqd. If Cfg_CnfrmReqd=0 no confirmation is required and the request executes immediately. If Cfg_CnfrmReqd=1 the instruction waits for confirmation OCmd_CmdCnfrm=1 and/or cancellation. For Cfg_CnfrmReqd=2 or 3 eSignature is needed before the confirmation and cancellation is enabled.

Virtualization

Use virtualization for instruction testing and operator training. Use PCmd_Virtual or MCmd_Virtual to enable virtualization. After finishing virtualization, use PCmd_Physical or MCmd_Physical to return to normal (physical device) operation.

When Virtualization is active, the outputs of the PVLV instruction hold at 0, virtual feedback of a working device is provided and I/O faults are ignored. Manipulate the instruction to operate as if a working valve is present.

Initialization

The instruction is normally initialized in the instruction first run. Request re-initialization by setting Inp_InitializeReq = 1. For proper initialization, when adding the instruction while performing an online edit of the code, make sure that Inp_InitializeReq = 1, the default value.

Configuration of Strings for HMI

Configure strings for HMI faceplates (FT View) and for the Logix Designer configuration dialog box. The strings are set to extended properties of tag items. Configure the strings in Logix Designer only.

- Description
- Label for graphic symbol
- Display Library for HMI Faceplate call-up
- Instruction name
- Area name
- URL link
- More Information

- Position 2 Command button
- Position 1 Command button
- Stop Command button
- Pulse Position 2 Command button
- Pulse Position 1 button
- Pulse button
- Position 2 Target state
- Position 1 Target state
- Stop Target state
- Pulse Position 2 Target state
- Pulse Position 1 Target state
- Position 2 Transition state
- Position 1 Transition state
- Stop Transition state
- Pulse Position 2 Transition state
- Pulse Position 1 Transition state
- Actuator Alarm
- IO Fault Alarm
- Full Stall Alarm
- Transit Stall Alarm
- Interlock Trip Alarm
- Loss of Position 1 Alarm
- Loss of Position 2 Alarm
- Trip Fail Alarm

Monitor the PVLV Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Any commands received before first scan are discarded. The valve is de-energized and treated as if it were commanded to safe position.
Instruction first run	Any commands received before first scan are discarded. The valve state is evaluated and the instruction aligns with the current state of the valve, as if the Hand command source were selected.
Rung-condition-in is false	Handled the same as if the valve is taken Out of Service by command. The valve outputs are de-energized, and the valve Command Source is shown as Program Out of Service on the HMI. All alarms are cleared. The rung-condition-out continues as false.
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	Any commands received before first scan are discarded. The valve is de-energized and treated as if it were commanded to safe position.
Instruction first run	Any commands received before first scan are discarded. The valve state is evaluated and the instruction aligns with the current state of the valve, as if the Hand command source were selected.
Instruction first scan	See instruction first run in the function block diagram table.
EnableIn is false	Handled the same as if the valve is taken Out of Service by command. The valve outputs are de-energized, and the valve Command Source is shown as Program Out of Service on the HMI. All alarms are cleared. EnableOut is set to false.
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

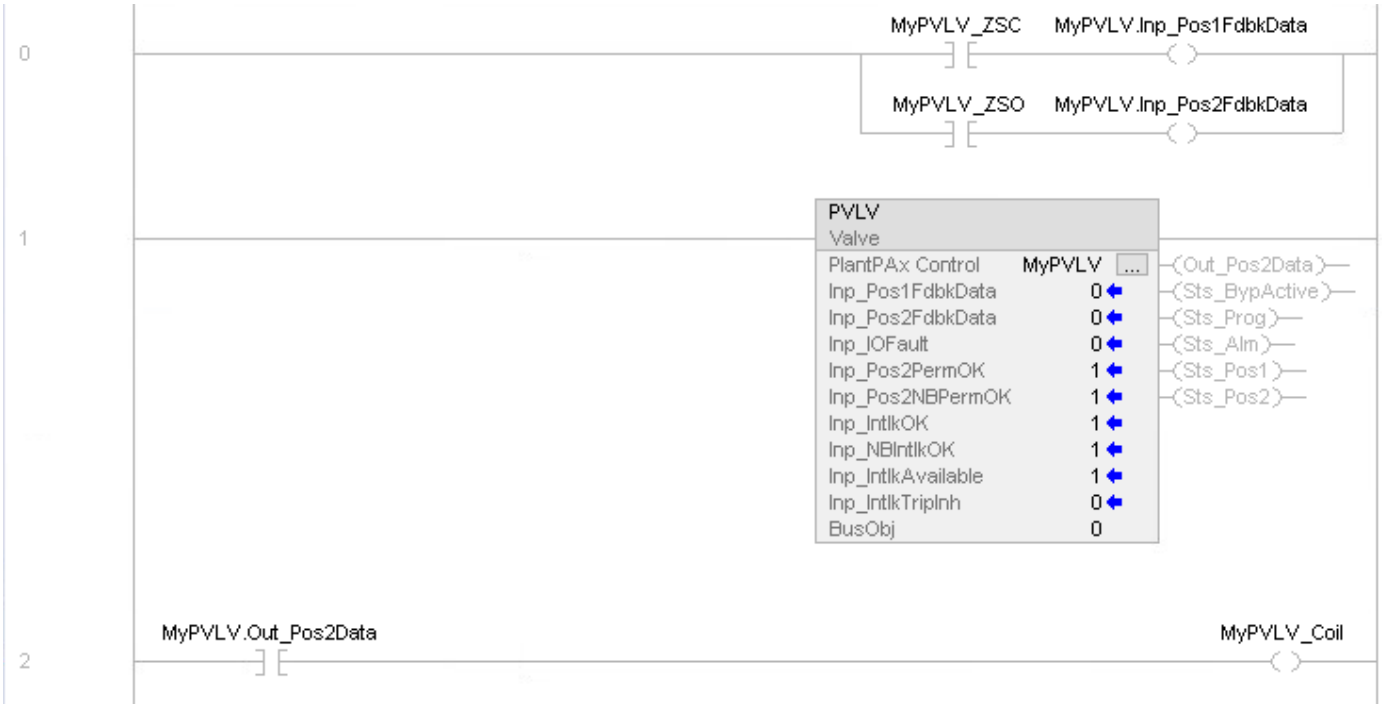
Structured Text

In Structured Text, EnableIn is always true during normal scan. Therefore, if the instruction is in the control path activated by the logic it will execute.

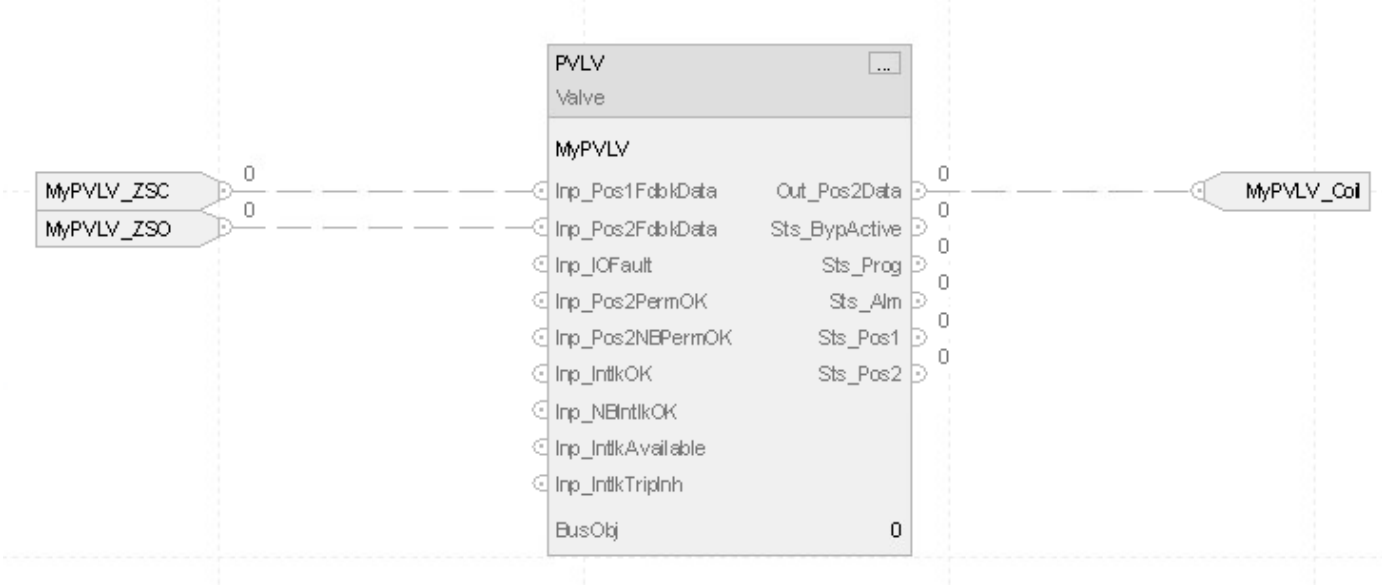
Condition/State	Action Taken
Prescan	See Prescan in the Function Block Diagram table.
Instruction first run	See Instruction first run in the Function Block Diagram table.
EnableIn is true	See EnableIn is true in the Function Block Diagram table.
Postscan	See Postscan in the Function Block Diagram table.

Example

Ladder Diagram



Function Block Diagram



Structured Text

```
MyPVLV.Inp_Pos1FdbkData:=MyPVLV_ZSC;
```

```
MyPVLV.Inp_Pos2FdbkData:=MyPVLV_ZSO;

PVLV(MyPVLV, 0);

MyPVLV_Coil:=MyPVLV.Out_Pos2Data;
```

See also

Process Valve (PVLV) Command Source on page 767

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Data Conversions on page 1086

Process Valve (PVLV) Command Source

The Process Valve (PVLV) instruction uses these command sources. The command sources are prioritized in order from highest to lowest in this table.

Command Source	Description
Hand	Hardwired logic or other logic outside the instruction controls the device. The instruction tracks the state of the device for bumpless transfer back to one of the other command sources. This is the highest priority command source.
Out-of-Service	The instruction is disabled. Drive commands and settings from any source are not accepted.
Maintenance	Maintenance controls the device and supersedes Operator, Program, External and Override control. Operator commands and settings from the HMI are accepted.
Override	Priority logic controls the device and supersedes Operator, Program and External control. Override Input (Inp_Ovrd) is accepted.
External	External logic (for example, field pilot control or upstream SCADA) controls the device. External commands (XCmd_) are accepted.
Program locked	Program logic controls the device. Program commands (PCmd_) are accepted. Operator cannot take control from the Program. Override cannot take control from theProgram unless Cfg_OvrdOverLock = 1.
Program	Program logic controls the device. Program commands (PCmd_) are accepted.
Operator locked	The Operator controls the device. Operator commands (OCmd_) from the HMI are accepted. Program cannot take control from Operator. Override cannot take control from Operator unless Cfg_OvrdOverLock = 1.
Operator	The Operator controls the device. Operator commands (OCmd_) from the HMI are accepted. This is the lowest priority command source.

The instruction enables or disables these operations:

- Maintenance Out-of-Service exists
- Maintenance exists
- External exists
- Program (unlocked) exists
- Program locked exists
- Operator (unlocked) exists
- Operator locked exists
- PCmd_Lock used as a Level (1 = Lock, 0 = Unlock)

The instruction checks for invalid configurations of control and forces the nearest valid configuration.

The core control model arbitrates the source of the commands and parameters that the receiving function accepts. The core control model determines if the source is:

- A programmatic entity which resides entirely within the processing environment, or
- An external interface entity which issues commands and parameters external and asynchronously to the processing environment.

Locking a control source prevents the other control source from acquiring privilege.

Core Command Source Model

The core control model consists of these control sources:

- Oper
- OperLocked
- Prog
- ProgLocked

The control model defaults to this configuration. Other control sources may be present in the model but act as overriding control sources, acting independent of the base Operator/Program state machine.

Enable control sources as Configuration

The user can enable and disable individual control sources. The default configuration uses the entire base model; upon power-up of the processing environment the control source will be the designated default. Some combinations of enabled control sources are disallowed as they are either unnecessary or could create unintended changes.

Prog Power Up

Configuration allows the user to specify whether Operator or Program is the power-up default.

Prog Priority

Configuration allows the user to specify whether Operator or Program commands take priority when simultaneously asserted.

Automatic reset of commands

All commands are treated as one-shot-latched. Commands are automatically cleared when the instruction executes and processes them.

Change Destination States

Under certain configurations the destination command source for some commands may change. This is in keeping with the intent of the command. For example, if the Program state is disabled, the destination of the OCmd_Prog command becomes the Program Locked state instead of the Program state. This maintains the intent of the OCmd_Prog command: the operator entity wishes to place the function in control of the program. If the command was eliminated then there would be no way to accomplish this. This is only done in configurations where it would cause no conflict or race condition, but serves to preserve as much user functionality as is practical.

Higher Priority Command Sources

These Higher priority command sources operate independently within the model:

- External
- Override
- Maintenance
- Out-of-Service
- In-Service
- Hand

See also

Process Valve (PVLV)

Process Valve Statistics (PVLVS)

This information applies to the ControlLogix 5380P and 5580P controllers.

The Process Valve Statistics Object (PVLVS) instruction monitors a two-state (open and close) valve and records statistics for stroke times and stroke counts to aid in planning maintenance or diagnosing valve and actuator problems. The PVLVS instruction is designed to work with the PVLV (solenoid, motor, and hand operated) valve instruction.

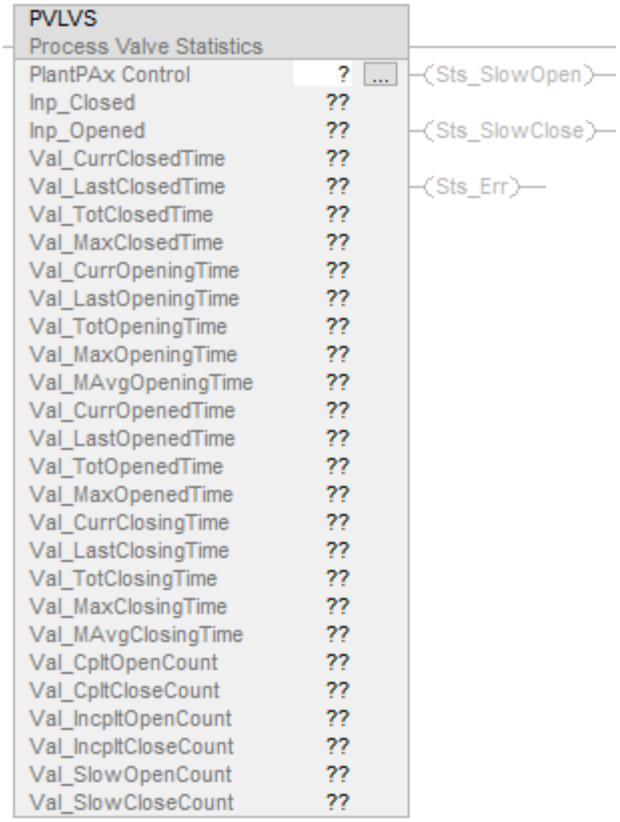
The PVLVS instruction records these statistics:

- Amount of time in the current state (closed, opening, opened, closing, stopped/other)
- Amount of time the valve was in each state the last time it was in that state (closed, opening, opened, closing, stopped/other)

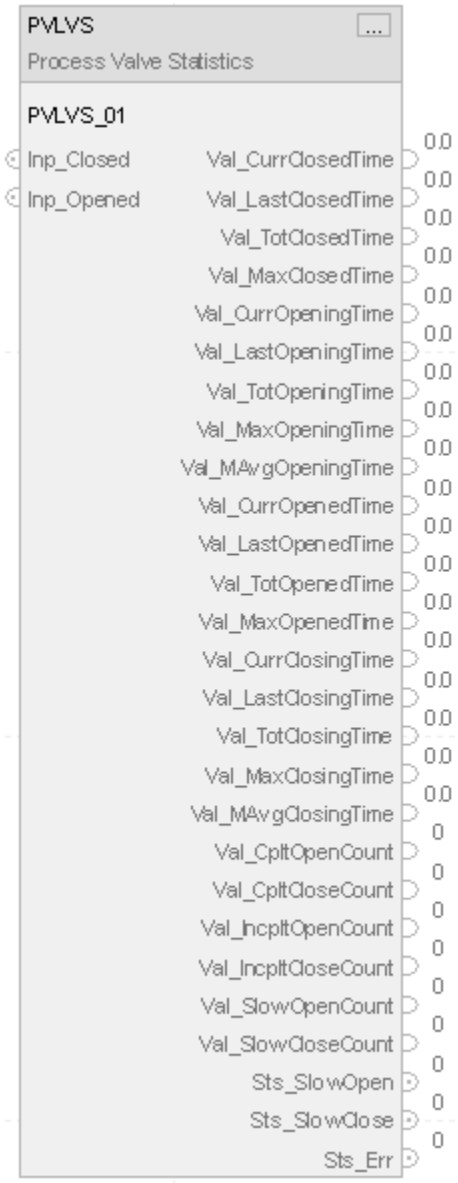
- Maximum amount of time spent in each state (closed, opening, opened, closing, and stopped/other); these are the highest values achieved by the previous statistics
- Total amount of time spent in each state (closed, opening, opened, closing, stopped/other)
- Moving average of the last 10 closing (close stroke) times
- Moving average of the last 10 opening (open stroke) times
- Number of completed open strokes (from closed to opened)
- Number of completed close strokes (from opened to closed)
- Number of incomplete open strokes (from closed to opening and back to closed)
- Number of incomplete close strokes (from opened to closing and back to opened)
- Number of times the valve was in the stopped/other state
- Number of open strokes that took longer than the configured Slow Open Time
- Number of close strokes that took longer than the configured Slow Close Time

Available Languages

Ladder Diagram



Function Block Diagram



Structured Text

PVLVS (PVLVS_01);

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PlantPAx Control	P_VALVE_STATISTICS	tag	Data structure required for proper operation of the instruction.

P_VALVE_STATISTICS Structure

Public members are standard, visible tag members that are programmatically accessible. Private, hidden members are used in HMI faceplates and are not programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	Description
EnableIn	BOOL	Enable Input - System Defined Parameter Default is true.
Inp_InitializeReq	BOOL	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request when reinitialization is needed. The instruction clears this operand automatically.
Inp_Closed	BOOL	1 = Device is confirmed Closed. Default is false.
Inp_Opened	BOOL	1 = Device is confirmed Opened. Default is false.
Inp_StopOther	BOOL	1 = Device is confirmed Stopped / Other. Default is false.
Cfg_HasStopOther	BOOL	1 = Device has Stopped or other state(s) to be monitored. Default is false.
Cfg_SlowOpenTime	REAL	Maximum time Opening (seconds) before raising Sts_SlowOpen. Valid = 0.0 to 2147483.0 seconds. Default is 10.0.
Cfg_SlowCloseTime	REAL	Maximum time Closing (seconds) before raising Sts_SlowClose. Valid = 0.0 to 2147483.0 seconds. Default is 10.0.
PCmd_ClearTotTimes	BOOL	Program command to clear total time statistics. Default is false.
PCmd_ClearMaxTimes	BOOL	Program command to clear maximum time statistics. Default is false.
PCmd_ClearStrokeCounts	BOOL	Program command to clear stroke count statistics. Default is false.
PCmd_ClearSlowCounts	BOOL	Program command to clear device slow count statistics. Default is false.
PCmd_ClearMAvgs	BOOL	Program command to clear moving average stroke times. Default is false.

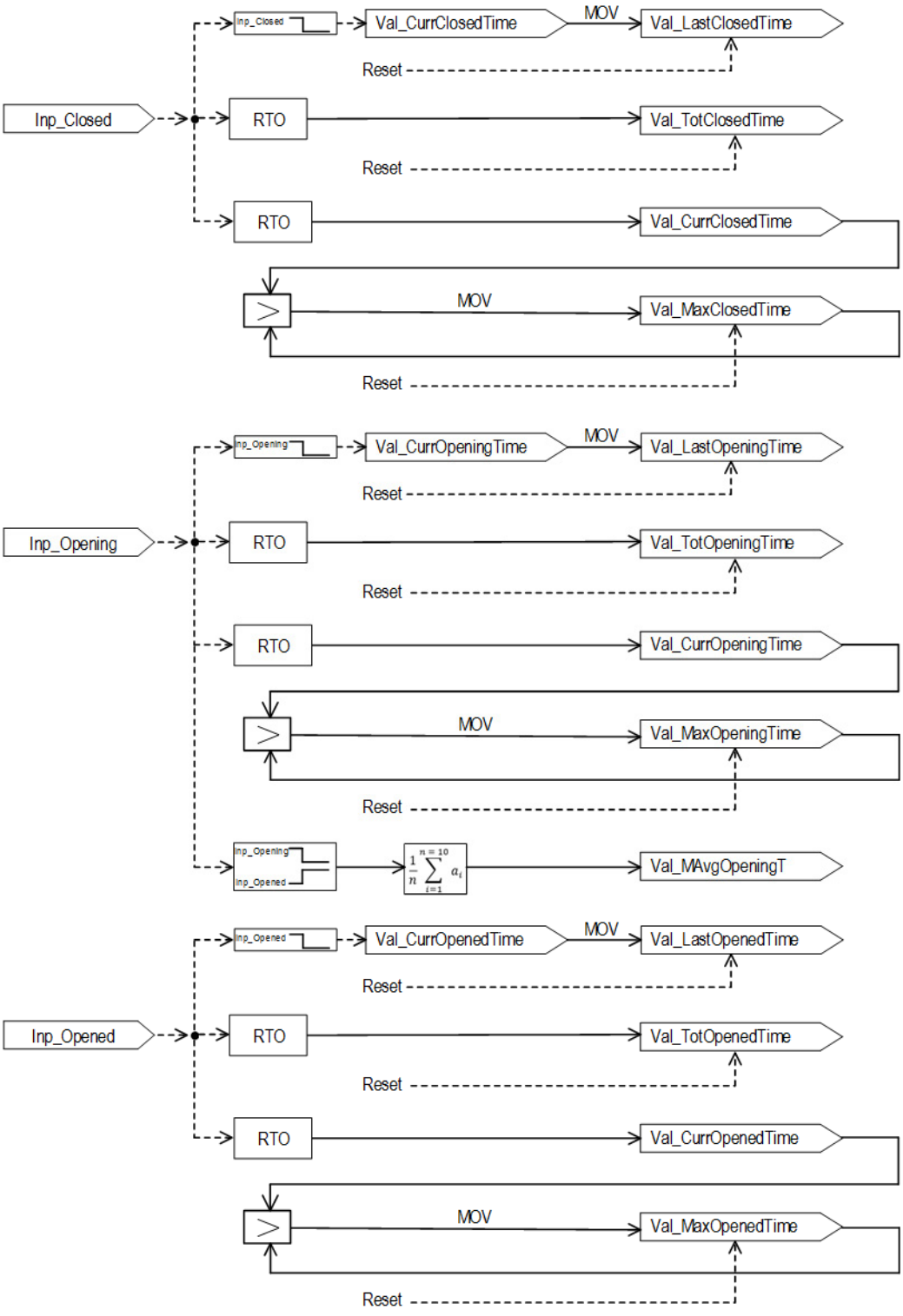
Public Output Members	Data Type	Description
EnableOut	BOOL	Enable Output - System Defined Parameter
Val_CurrClosedTime	REAL	Current time in Closed state (seconds).
Val_LastClosedTime	REAL	Time in Closed state (seconds) last time device was closed.
Val_TotClosedTime	REAL	Accumulated time in Closed state (hours).
Val_MaxClosedTime	REAL	Maximum time in Closed state (hours) of any occurrence.

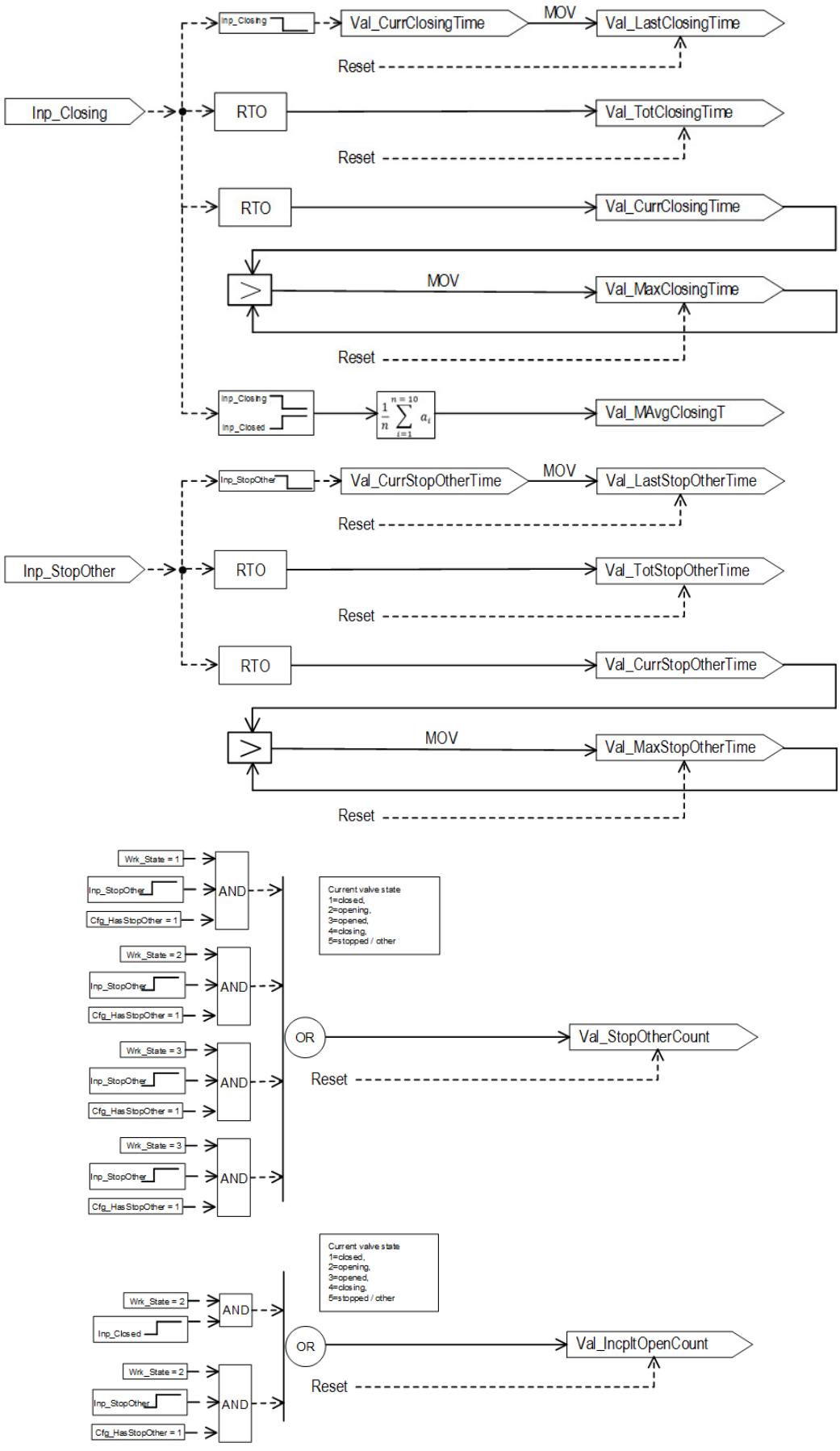
Public Output Members	Data Type	Description
Val_CurrOpeningTime	REAL	Current time in Opening state (seconds).
Val_LastOpeningTime	REAL	Time in Opening state (seconds) last time device was opening.
Val_TotOpeningTime	REAL	Accumulated time in Opening state (hours).
Val_MaxOpeningTime	REAL	Maximum time in Opening state (hours) of any occurrence.
Val_MAvgOpeningTime	REAL	Moving average Open stroke time, last 10 complete open strokes (seconds).
Val_CurrOpenedTime	REAL	Current time in Opened state (seconds).
Val_LastOpenedTime	REAL	Time in Opened state (seconds) last time device was opened.
Val_TotOpenedTime	REAL	Accumulated time in Opened state (hours).
Val_MaxOpenedTime	REAL	Maximum time in Opened state (hours) of any occurrence.
Val_CurrClosingTime	REAL	Current time in Closing state (seconds).
Val_LastClosingTime	REAL	Time in Closing state (seconds) last time device was closing.
Val_TotClosingTime	REAL	Accumulated time in Closing state (hours).
Val_MaxClosingTime	REAL	Maximum time in Closing state (hours) of any occurrence.
Val_MAvgClosingTime	REAL	Maximum time in Closing state (seconds) of any occurrence.
Val_CurrStopOtherTime	REAL	Current time in Stopped / Other state (seconds).
Val_LastStopOtherTime	REAL	Time in Stopped / Other state (seconds) last time device was stopped (or other).
Val_TotStopOtherTime	REAL	Accumulated time in Stopped / Other state (hours).
Val_MaxStopOtherTime	REAL	Maximum time in Stopped / Other state (hours) of any occurrence.
Val_CpltOpenCount	DINT	Count of complete device Open strokes (Closed to Opened).
Val_CpltCloseCount	DINT	Count of complete device Close strokes (Opened to Closed).
Val_IncpltOpenCount	DINT	Count of incomplete device Open strokes (Closed - Moving - Closed).
Val_IncpltCloseCount	DINT	Count of incomplete device Close strokes (Opened - Moving - Opened).
Val_StopOtherCount	DINT	Count of device Stopped / Other occurrences.
Val_SlowOpenCount	DINT	Count of device Slow to Open occurrences.
Val_SlowCloseCount	DINT	Count of device Slow to Close occurrences.
Sts_SlowOpen	BOOL	1 = Last Closed to Opened stroke exceeded configured time threshold.
Sts_SlowClose	BOOL	1 = Last Opened to Closed stroke exceeded configured time threshold.
Sts_Err	BOOL	1 = Configuration Error: invalid Slow Open Time or Slow Close Time.
Sts_ErrSlowCloseTime	BOOL	1 = Error in configuration: Invalid SlowCloseTime timer preset (use 0.0 to 2147483.0).
Sts_ErrSlowOpenTime	BOOL	1 = Error in configuration: Invalid SlowOpenTime timer preset (use 0.0 to 2147483.0).

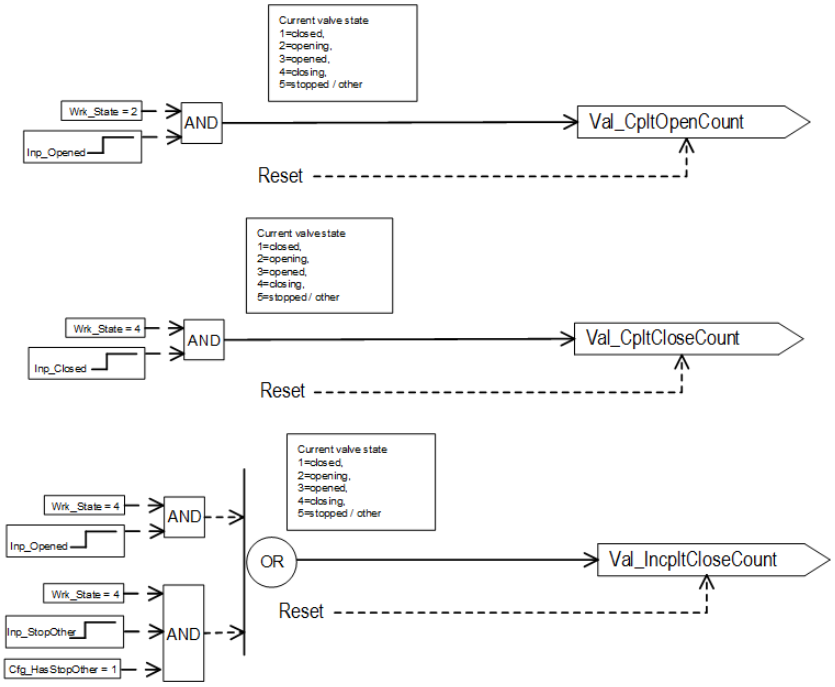
Private Input Members	Data Type	Description
MCmd_ClearMAvgs	BOOL	Maintenance command to clear moving average stroke times. Default is false.
MCmd_ClearMaxTimes	BOOL	Maintenance command to clear maximum time statistics. Default is false.
MCmd_ClearSlowCounts	BOOL	Maintenance command to clear device slow count statistics. Default is false.
MCmd_ClearStrokeCounts	BOOL	Maintenance command to clear stroke count statistics. Default is false.
MCmd_ClearTotTimes	BOOL	Maintenance command to clear total time statistics. Default is false.

Operation

These diagrams illustrate the functionality of the PVLVS instruction:


Inp_Closed
Val_CurrClosedTime
MOV
Val_LastClosedTime
Reset
Inp_Closed
RTO
Val_TotClosedTime
Reset
RTO
Val_CurrClosedTime
MOV
Val_MaxClosedTime
Reset
Inp_Opening
Val_CurrOpeningTime
MOV
Val_LastOpeningTime
Reset
Inp_Opening
RTO
Val_TotOpeningTime
Reset
RTO
Val_CurrOpeningTime
MOV
Val_MaxOpeningTime
Reset
Inp_Opening
Inp_Opened
Val_MAvgOpeningT
Inp_Opened
Val_CurrOpenedTime
MOV
Val_LastOpenedTime
Reset
Inp_Opened
RTO
Val_TotOpenedTime
Reset
RTO
Val_CurrOpenedTime
MOV
Val_MaxOpenedTime
Reset


Inp_Closing
Val_CurrClosingTime
MOV
Val_LastClosingTime
Reset
Inp_Closing
RTO
Val_TotClosingTime
Reset
RTO
Val_CurrClosingTime
MOV
Val_MaxClosingTime
Reset
Inp_Closing
Inp_Closed
$\frac{1}{n}\sum_{i=1}^{n=10} a_i$
Val_MAvgClosingT
Inp_StopOther
Val_CurrStopOtherTime
MOV
Val_LastStopOtherTime
Reset
Inp_StopOther
RTO
Val_TotStopOtherTime
Reset
RTO
Val_CurrStopOtherTime
MOV
Val_MaxStopOtherTime
Reset
Wrk_State = 1
Inp_StopOther
Cfg_HasStopOther = 1
AND
Wrk_State = 2
Inp_StopOther
Cfg_HasStopOther = 1
AND
Wrk_State = 3
Inp_StopOther
Cfg_HasStopOther = 1
AND
Wrk_State = 3
Inp_StopOther
Cfg_HasStopOther = 1
AND
Current valve state
1=closed,
2=opening,
3=opened,
4=closing,
5=stopped / other
OR
Val_StopOtherCount
Reset
Wrk_State = 2
Inp_Closed
AND
Wrk_State = 2
Inp_StopOther
Cfg_HasStopOther = 1
AND
Current valve state
1=closed,
2=opening,
3=opened,
4=closing,
5=stopped / other
OR
Val_IncpltOpenCount
Reset



Configuration of Strings for HMI

Configure strings for HMI faceplates (FT View) and for the Logix Designer configuration dialog box. The strings are set to extended properties of tag items. Configure the strings in Logix Designer only.

- Valve is confirmed Stopped / Other (in Logix Designer dialog box) – Description of Inp_StopOther item.
- Description
- Label for graphic symbol
- Display Library for HMI Faceplate call-up
- Instruction name
- Area name
- URL link

Monitor the PVLVS instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Rung-condition-out is cleared to false. Total times, stroke counts and slow counts are maintained through a power cycle. Current times are cleared. The internal state is set to **unknown**. When an input shows the valve in a known condition, the main logic transitions to the corresponding state.
Instruction first run	All commands that are automatically cleared each execution are cleared and ignored. The instruction executes normally.
Rung-condition-in is false	Set rung-condition-out to rung-condition-in. If this instruction is on a false rung, or if EnableIn is false in FBD, Total times, stroke counts, and slow counts are maintained, but Current times are cleared. The internal state is set to **unknown**. Copies the current time of the previous state to its last time and resets the current time (and accumulated 10ths of hours, if applicable). The internal state is set to **unknown** (disabled). When normal execution (Logic routine) is resumed, when an input shows the valve in a known condition, a transition to the corresponding state occurs. Should scan all the retentive timers FALSE to stop them.
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Instruction first run	See Instruction first run in the Ladder Diagram table.
Instruction first scan	See Instruction first run in the Ladder Diagram table.
EnableIn is false	See Rung-condition-in is false in the Ladder Diagram table.
EnableIn is true	See Rung-condition-in is true in the Ladder Diagram table.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

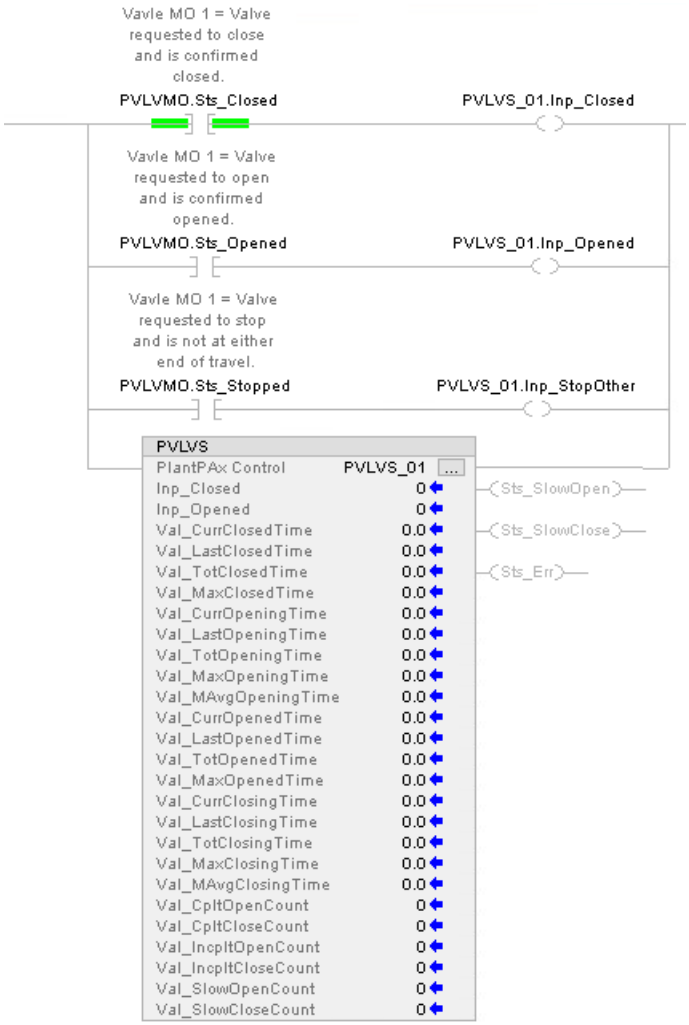
In Structured Text, EnableIn is always true during normal scan. Therefore, if the instruction is in the control path activated by the logic it will execute.

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Instruction first run	See Instruction first run in the Ladder Diagram table.
EnableIn is true	See Rung-condition-in is true in the Ladder Diagram table.
Postscan	EnableIn and EnableOut bits are cleared to false.

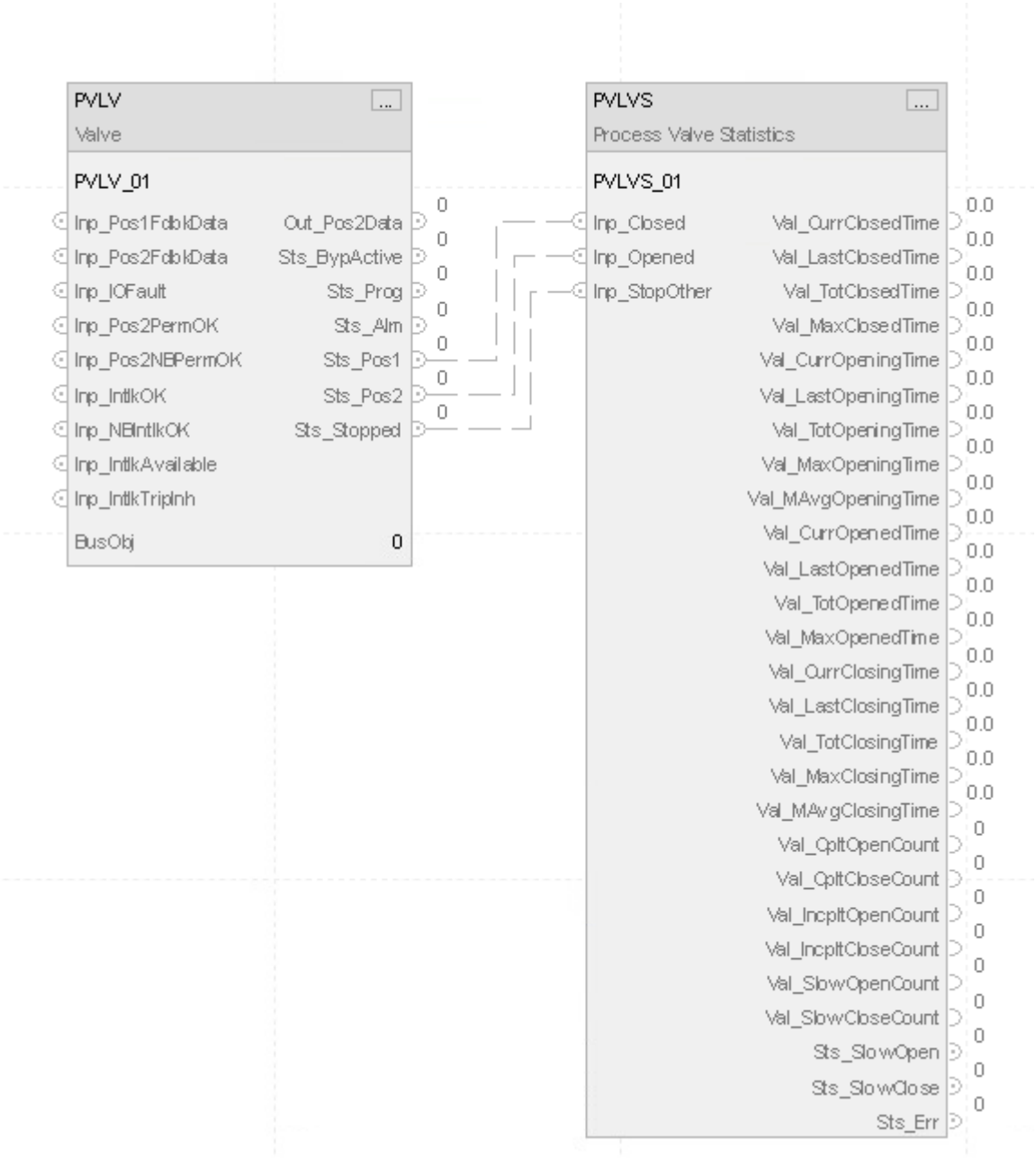
Example

This section shows how the PVLVS instruction collects statistics on a motor-operated valve. In this example, the motor-operated valve is controlled by using the PVLVS instruction. By naming the PVLVS instance tag the same as the PVLV tag plus ValveStats, the PVLVS instance tag is automatically linked at the HMI to the valve instance. In this example, the motor-operated valve is opened, closed, or the motor could stop moving while in travel before reaching either position. Statistics for all of these three states can be tracked by using the PVLVS instruction. In this example, the parameters Inp_Closed, Inp_Opened, and Inp_StopOther are connected to the parameters Sts_Closed, Sts_Opened, and Sts_Stopped of the PVLV instruction. The PVLVS instruction keeps track of completed strokes, plus open and close strokes that are slower than expected. The parameters Cfg_SlowOpenTime and Cfg_SlowCloseTime are set to 10, to indicate that any transition longer than 10 seconds is considered slow.

Ladder Diagram



Function Block Diagram



Structured Text

```
PVLVMO_ValveStat.Inp_Closed := PVLVMO.Sts_Closed;
PVLVMO_ValveStat.Inp_Opened := PVLVMO.Sts_Opened;
PVLVMO_ValveStat.Inp_StopOther := PVLVMO.Sts_Stopped;
PVLVS (PVLVMO_ValveStat);
```

See also

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Process Variable Speed Drive (PVSD)

This information applies to the ControlLogix 5380P and 5580P controllers.

The Process Variable Speed Drive (PVSD) instruction monitors and controls a variable speed motor using an AC (variable frequency) or DC drive. Use the instruction to run or jog the motor, forward or reverse. The drive interface can be through a Device Object Interface or through individual pins. The object is a built-in version of the existing P_VSD add-on instruction in the Rockwell Automation Library of Process Objects.

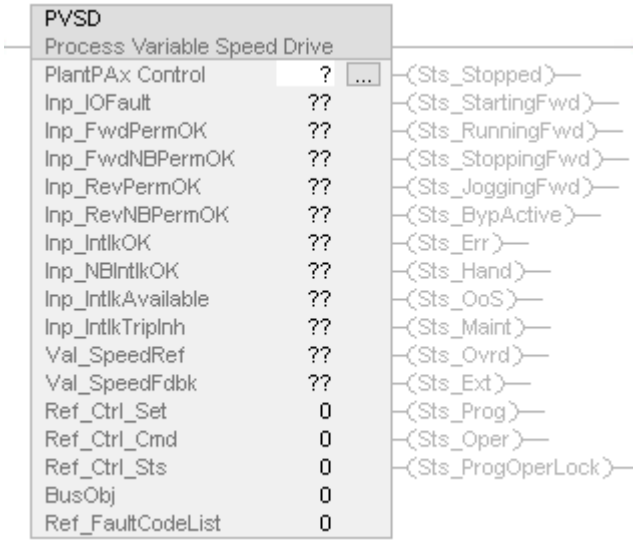
Use the PVSD instruction to:

- Control and monitor a variable speed motor using an AC or DC drive. This instruction is used with drives controlling velocity, not position, and it does not use any motion axes.
- Select Operator, Program, External, Override, Maintenance, Out of Service, or Hand as the source of drive commands and settings.
- Use the selected command source to enter a speed reference (setpoint).
- Use the selected command source to start the drive forward.
- Use the selected command source to start the drive reverse, if configured for reversing.
- Use the selected command source to jog the drive forward, if configured for jogging forward. Only Operator, External and Maintenance command sources are permitted to jog the drive forward.
- Use the selected command source to jog the drive reverse, if configured for jogging reverse. Only Operator, External and Maintenance command sources are permitted to jog the drive reverse.
- Monitor actual drive status, including:
  - Speed feedback
  - Drive ready
  - Drive active (run feedback)
  - Commanded direction
  - Actual direction
  - Accelerating
  - Decelerating
  - At speed
  - Drive warning
  - Drive faulted (with fault code and description)

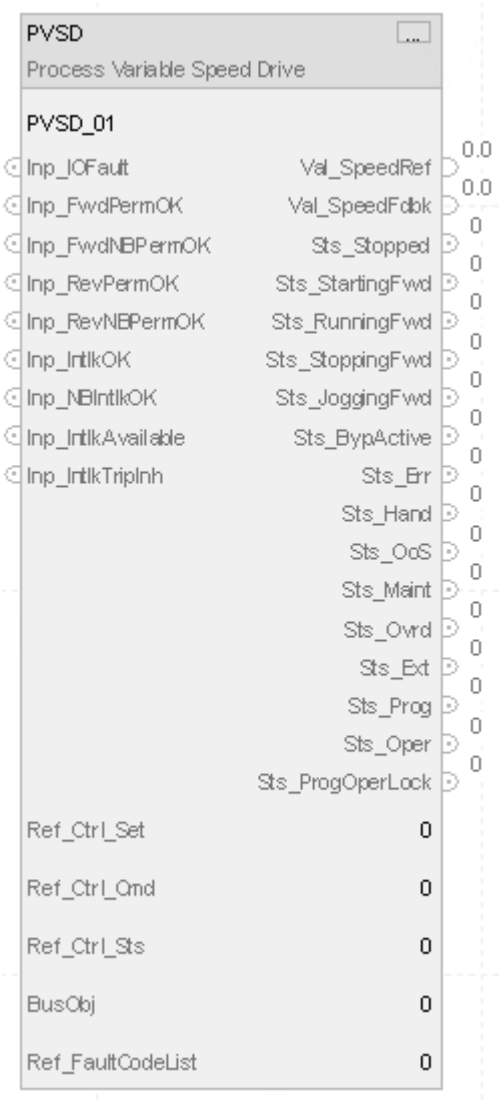
- Interface to a drive Device Object using a set of Power Velocity interface tags. If the interface tags are not linked (optional InOut parameters), a set of input and output parameters are used to interface to the drive signal-by-signal.
- Search a linked Fault Code Lookup Table to provide textual drive fault information, or use text provided through the Power Velocity interface fault record.
- Participate in a control strategy bus (BUS_OBJ) with other devices and process instructions.
- Configure an output to provide a pre-start audible warning (horn). The time the alert sounds before starting or jogging is configurable.
- Configure virtualization, providing simulated feedback of a working drive while disabling outputs to the physical device.
- Configure scaling of the speed reference from application engineering units to drive interface units.
- Configure scaling of the speed feedback from drive interface units to application engineering units.
- Configure limiting (clamping) of the speed reference.
- Monitor run feedback and provide status and alarms for failure to start in the configured time and failure to stop in the configured time.
- Monitor permissive conditions to allow starting or jogging the motor forward.
- Monitor permissive conditions to allow starting or jogging the motor reverse.
- Monitor interlock conditions to stop and prevent starting or jogging the motor. Trigger an alarm if interlock conditions cause the motor to stop.
- Monitor I/O communication faults.
- Automatically clear latched alarms and drive faults when an Operator Command (Start, Stop, Jog) is received.
- Automatically clear latched alarms and drive faults when an External Command (Start, Stop, Jog) is received.
- Use HMI breadcrumbs for Alarm Inhibited, Bad Configuration, Not Ready, and Maintenance Bypass Active.
- Use Available status for use by automation logic to indicate whether the motor can be controlled by other objects.
- Use Alarms for Fail to Start, Fail to Stop, Interlock Trip, I/O Fault and Drive Fault conditions.

Available Languages

Ladder Diagram



Function Block Diagram



Structured Text

PVSD(*PlantPAx Control*, Ref_Ctrl_Set, Ref_Ctrl_Cmd, Ref_Ctrl_Sts, BusObj, Ref_FaultCodeList)

Operands

IMPORTANT Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Configuration Operands

Operand	Type	Format	Description
PlantPAx Control	P_VARIABLE_SPEED_DRIVE	tag	Data structure required for proper operation of instruction.
Ref_Ctrl_Set	RAC_ITF_DVC_PWRVELOCITY_SET	tag	Velocity Automation Device Object Settings Interface.
Ref_Ctrl_Cmd	RAC_ITF_DVC_PWRVELOCITY_CMD	tag	Velocity Automation Device Object Command Interface.
Ref_Ctrl_Sts	RAC_ITF_DVC_PWRVELOCITY_STS	tag	Velocity Automation Device Object Status Interface.
BusObj	BUS_OBJ	tag	Bus component.
Ref_FaultCodeList	RAC_CODE_DESCRIPTION[x]	tag	Fault Code to Fault Description lookup table for intelligent motor controller.

P_VARIABLE_SPEED_DRIVE Structure

Public members are standard, visible tag members that are programmatically accessible. Private, hidden members are used in HMI faceplates and are not programmatically accessible. Private members are listed in separate tables after public members.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
EnableIn	BOOL	Not Visible	Not Required	Input	Enable Input - System Defined Parameter Default is true.
Inp_InitializeReq	BOOL	Not Visible	Not Required	Input	1 = Request to initialize the instruction. The instruction is normally initialized in instruction first run. Use this request when reinitialization is needed. The instruction clears this operand automatically. Default is true.
Inp_OwnerCmd	DINT	Not Visible	Not Required	Input	Owner device command. 0 = None, .10 = Operator Lock, .11 = Operator Unlock, .12 = Program Lock, .13 = Program Unlock, .14 = Acquire Maintenance, .15 = Release Maintenance, .16 = Acquire External, .17 = Release External, .29 = Echo. Default is 0.
Inp_SpeedFdbkData	REAL	Not Visible	Not Required	Input	Speed feedback in drive (raw) units (example: 0 to 32767 in drive units represents 0 to max frequency). Default is 0.0.
Inp_DatalinkData	REAL	Not Visible	Not Required	Input	Auxiliary signal (datalink) input in drive (raw) units. Default is 0.0.
Inp_LastFaultCodeData	DINT	Not Visible	Not Required	Input	Most recent drive fault code (enumeration). Default is 0.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Inp_ReadyData	BOOL	Not Visible	Not Required	Input	1 = Drive is ready to run. Default is true.
Inp_RunningData	BOOL	Not Visible	Not Required	Input	1 = Drive is running (active). Default is false.
Inp_CommandDirData	BOOL	Not Visible	Not Required	Input	1 = Drive is commanded forward, 0 = drive is commanded reverse. Default is true.
Inp_ActualDirData	BOOL	Not Visible	Not Required	Input	1 = Drive is running forward, 0 = drive is running reverse. Default is true.
Inp_AcceleratingData	BOOL	Not Visible	Not Required	Input	1 = Drive is accelerating. Default is false.
Inp_DeceleratingData	BOOL	Not Visible	Not Required	Input	1 = Drive is decelerating. Default is false.
Inp_AtSpeedData	BOOL	Not Visible	Not Required	Input	1 = Drive is at commanded speed. Default is false.
Inp_AlarmData	BOOL	Not Visible	Not Required	Input	1 = Drive has an alarm (warning) condition. See drive display or manual for detail. Default is false.
Inp_FaultedData	BOOL	Not Visible	Not Required	Input	1 = Drive has faulted. See drive display or manual for detail. Default is false.
Inp_DvcNotify	SINT	Not Visible	Not Required	Input	Related device object alarm priority and acknowledgement status. 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Inp_IOFault	BOOL	Visible	Not Required	Input	Indicates the IO data are inaccurate. 0 = The IO data are good, 1 = The IO data are bad, causing fault. If the drive is not virtual, this input sets Sts_IOFault, which raises IOFault Alarm. Default is false.
Inp_FwdPermOK	BOOL	Visible	Not Required	Input	1 = Permissives OK, drive can start or jog forward. Default is true.
Inp_FwdNBPermOK	BOOL	Visible	Not Required	Input	1 = Non-Bypassable Permissives OK, drive can start or jog forward. Default is true.
Inp_RevPermOK	BOOL	Visible	Not Required	Input	1 = Permissives OK, drive can start or jog reverse. Default is true.
Inp_RevNBPermOK	BOOL	Visible	Not Required	Input	1 = Non-Bypassable Permissives OK, drive can start or jog reverse. Default is true.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Inp_IntlkOK	BOOL	Visible	Not Required	Input	1 = Interlocks OK, drive can start or jog and keep running. Default is true.
Inp_NBIntlkOK	BOOL	Visible	Not Required	Input	1 = Non-bypassable interlocks OK, drive can start or jog and keep running. Default is true.
Inp_IntlkAvailable	BOOL	Visible	Not Required	Input	1 = Interlock Availibility OK. Default is false.
Inp_IntlkTripInh	BOOL	Visible	Not Required	Input	1 = Inhibit Interlock Trip Status. Default is false.
Inp_RdyReset	BOOL	Not Visible	Not Required	Input	1 = Related object, reset by this object, is ready to be reset. Default is false.
Inp_Hand	BOOL	Not Visible	Not Required	Input	1 = Acquire Hand (typically hardwired local), 0 = Release Hand. Default is false.
Inp_Ovrd	BOOL	Not Visible	Not Required	Input	1 = Acquire Override (higher priority program logic), 0 = Release Override. Default is false.
Inp_OvrdCmd	SINT	Not Visible	Not Required	Input	Override Command: 0 = None, 1 = Stop, 2 = Start Forward, 3 = Start Reverse. Default is 0.
Inp_OvrdSpeed	REAL	Not Visible	Not Required	Input	Value to set Speed Reference in Override, in speed reference engineering units. Default is 0.0.
Inp_OvrdOutDatalink	REAL	Not Visible	Not Required	Input	Value to set Output Datalink in Override, in output datalink engineering units. Default is 0.0.
Inp_ExtInh	BOOL	Not Visible	Not Required	Input	1 = Inhibit External acquisition, 0 = Allow External acquisition. Default is false.
Inp_HornInh	BOOL	Not Visible	Not Required	Input	1 = Inhibit audible alert, 0 = Alllow audible alert. Default is false.
Inp_Reset	BOOL	Not Visible	Not Required	Input	1 = Reset Shed Latches and Cleared Alarms. Default is false.
Cfg_AllowDisable	BOOL	Not Visible	Not Required	Input	1 = Allow Maintenance to disable alarms. Default is true.
Cfg_AllowShelve	BOOL	Not Visible	Not Required	Input	1 = Allow Operator to shelve alarms. Default is true.
Cfg_HasReverse	BOOL	Not Visible	Not Required	Input	1 = Drive can run or jog reverse, 0 = only allow forward operation. Default is false.
Cfg_HasJog	BOOL	Not Visible	Not Required	Input	1 = Drive jog command enabled and visible, 0 = drive jog command not allowed. Default is false.
Cfg_AllowLocal	BOOL	Not Visible	Not Required	Input	1 = Allow Local Start and Stop without alarm, 0 = Start or Stop by command only. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_HasRunFdbk	BOOL	Not Visible	Not Required	Input	1 = Drive provides feedback signal when running. Default is false.
Cfg_UseRunFdbk	BOOL	Not Visible	Not Required	Input	1 = Drive run feedback should be used for failure checking. Default is false.
Cfg_HasSpeedFdbk	BOOL	Not Visible	Not Required	Input	1 = Drive provides speed feedback. Default is false.
Cfg_UseSpeedFdbk	BOOL	Not Visible	Not Required	Input	1 = Drive speed determines running state, 0 = state determined by drive active feedback. Default is false.
Cfg_HasInpDatalink	BOOL	Not Visible	Not Required	Input	1 = A signal is connected to Inp_DatalinkData. Default is false.
Cfg_HasOutDatalink	BOOL	Not Visible	Not Required	Input	1 = A signal is connected to Out_DatalinkData. Default is false.
Cfg_HasDvcObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI a device (e.g., drive) object is connected. Default is false.
Cfg_HasFwdPermObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI an object is connected to forward permissive inputs. Default is false.
Cfg_HasRevPermObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI an object is connected to reverse permissive inputs. Default is false.
Cfg_HasIntlkObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI an object is connected to interlock inputs. Default is false.
Cfg_HasResInhObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI a Restart Inhibit object is connected. Default is false.
Cfg_HasRunTimeObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI a Run Time / Starts object is connected. Default is false.
Cfg_HasMoreObj	BOOL	Not Visible	Not Required	Input	1 = Tells HMI an object with more information is available. Default is false.
Cfg_SetTrack	BOOL	Not Visible	Not Required	Input	1 = Settings track for unselected sources, 0 = no tracking of settings. Default is true.
Cfg_SetTrackOvrdHand	BOOL	Not Visible	Not Required	Input	1 = Program, Operator and External settings track when Override or Hand is selected. Default is false.
Cfg_OperStopPrio	BOOL	Not Visible	Not Required	Input	1 = OCmd_Stop accepted any time; 0 = OCmd_Stop accepted only when Oper is selected. Default is false.
Cfg_ExtStopPrio	BOOL	Not Visible	Not Required	Input	1 = XCmd_Stop accepted any time; 0 = XCmd_Stop accepted only when Ext is selected. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_OCmdResets	BOOL	Not Visible	Not Required	Input	1 = Any drive OCmd resets shed latches and cleared alarms; 0 = OCmdReset is required. Default is false.
Cfg_XCmdResets	BOOL	Not Visible	Not Required	Input	1 = Any drive XCmd resets shed latches and cleared alarms; 0 = XCmdReset is required. Default is false.
Cfg_OvrdPermIntlk	BOOL	Not Visible	Not Required	Input	1 = Override ignores Bypassable Perm/ Intlk; 0 = Override uses all Perm/Intlk. Default is false.
Cfg_ShedOnFailToStart	BOOL	Not Visible	Not Required	Input	1 = Stop Motor and Alarm on Fail to Start; 0 = Alarm only on Fail to Start. Default is true.
Cfg_ShedOnIOFault	BOOL	Not Visible	Not Required	Input	1 = Stop Motor and Alarm on I/O Fault; 0 = Alarm only on I/O Fault. Default is true.
Cfg_HasOper	BOOL	Not Visible	Not Required	Input	1 = Operator (unlocked) exists, can be selected. Default is true.
Cfg_HasOperLocked	BOOL	Not Visible	Not Required	Input	1 = Operator Locked exists, can be selected. Default is true.
Cfg_HasProg	BOOL	Not Visible	Not Required	Input	1 = Program (unlocked) exists, can be selected. Default is true.
Cfg_HasProgLocked	BOOL	Not Visible	Not Required	Input	1 = Program Locked exists, can be selected. Default is true.
Cfg_HasExt	BOOL	Not Visible	Not Required	Input	1 = External exists, can be selected. Default is false.
Cfg_HasMaint	BOOL	Not Visible	Not Required	Input	1 = Maintenance exists, can be selected. Default is true.
Cfg_HasMaintOoS	BOOL	Not Visible	Not Required	Input	1 = Maintenance Out of Service exists, can be selected. Default is true.
Cfg_OvrdOverLock	BOOL	Not Visible	Not Required	Input	1 = Override supersedes Program/Operator Lock, 0 = Don't override Lock. Default is true.
Cfg_ExtOverLock	BOOL	Not Visible	Not Required	Input	1 = External supersedes Program/Operator Lock, 0 = Don't override Lock. Default is false.
Cfg_ProgPwrUp	BOOL	Not Visible	Not Required	Input	1 = Power up to Program, 0 = Power up to Operator. Default is false.
Cfg_ProgNormal	BOOL	Not Visible	Not Required	Input	Normal Source: 1 = Program if no requests; 0 = Operator if no requests. Default is false.
Cfg_PCmdPriority	BOOL	Not Visible	Not Required	Input	Command priority. 1 = Program commands win, 0 = Operator commands win. Default is false.
Cfg_PCmdProgAsLevel	BOOL	Not Visible	Not Required	Input	1 = PCmd_Prog used as a Level (1 = Prog, 0 = Oper). Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_PCmdLockAsLevel	BOOL	Not Visible	Not Required	Input	1 = PCmd_Lock used as a Level (1 = Lock, 0 = Unlock). Default is false.
Cfg_ExtAcqAsLevel	BOOL	Not Visible	Not Required	Input	1 = XCmd_Acq used as Level (1 = Acquire, 0 = Release). Default is false.
Cfg_DecPlcs	SINT	Not Visible	Not Required	Input	Number of decimal places for speed ref/fdbk display (0 to 6). Default is 2.
Cfg_InpDatalinkDecPlcs	SINT	Not Visible	Not Required	Input	Number of decimal places for Input Datalink display (0 to 6). Default is 2.
Cfg_OutDatalinkDecPlcs	SINT	Not Visible	Not Required	Input	Number of decimal places for Output Datalink display (0 to 6). Default is 2.
Cfg_MinSpeedRef	REAL	Not Visible	Not Required	Input	Minimum Speed Reference in EU (for limiting). Valid = any float less than or equal to Max. Default is 0.0.
Cfg_MaxSpeedRef	REAL	Not Visible	Not Required	Input	Maximum Speed Reference in EU (for limiting). Valid = any float greater than or equal to Min. Default is 60.0.
Cfg_JogSpeedRef	REAL	Not Visible	Not Required	Input	Speed Reference to use when Jogging (EU). Valid = any float (will be clamped). Default is 10.0.
Cfg_SpeedRefRawMin	REAL	Not Visible	Not Required	Input	Speed Reference Minimum in Drive (raw) Units (for scaling). Valid = any float not equal to Max. Default is 0.0.
Cfg_SpeedRefRawMax	REAL	Not Visible	Not Required	Input	Speed Reference Maximum in Drive (raw) Units (for scaling). Valid = any float not equal to Min. Default is 60.0.
Cfg_SpeedRefEUMin	REAL	Not Visible	Not Required	Input	Speed Reference Minimum in Engineering Units (for scaling). Valid = any float less than Max. Default is 0.0.
Cfg_SpeedRefEUMax	REAL	Not Visible	Not Required	Input	Speed Reference Maximum in Engineering Units (for scaling). Valid = any float greater than Min. Default is 60.0.
Cfg_SpeedFdbkRawMin	REAL	Not Visible	Not Required	Input	Speed Feedback Minimum in Drive (raw) Units (for scaling). Valid = any float not equal to Max. Default is 0.0.
Cfg_SpeedFdbkRawMax	REAL	Not Visible	Not Required	Input	Speed Feedback Maximum in Drive (raw) Units (for scaling). Valid = any float not equal to Min. Default is 60.0.
Cfg_SpeedFdbkEUMin	REAL	Not Visible	Not Required	Input	Speed Feedback Minimum in Engineering Units (for scaling). Valid = any float less than Max. Default is 0.0.
Cfg_SpeedFdbkEUMax	REAL	Not Visible	Not Required	Input	Speed Feedback Maximum in Engineering Units (for scaling). Valid = any float greater than Min. Default is 60.0.
Cfg_InpDatalinkRawMin	REAL	Not Visible	Not Required	Input	Input Datalink Minimum in Drive (raw) Units (for scaling). Valid = any float not equal to Max. Default is 0.0.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_InpDatalinkRawMax	REAL	Not Visible	Not Required	Input	Input Datalink Maximum in Drive (raw) Units (for scaling). Valid = any float not equal to Min. Default is 100.0.
Cfg_InpDatalinkEUMin	REAL	Not Visible	Not Required	Input	Input Datalink Minimum in Engineering Units (for scaling). Valid = any float less than Max. Default is 0.0.
Cfg_InpDatalinkEUMax	REAL	Not Visible	Not Required	Input	Input Datalink Maximum in Engineering Units (for scaling). Valid = any float greater than Min. Default is 100.0.
Cfg_OutDatalinkMin	REAL	Not Visible	Not Required	Input	Minimum Output Datalink in EU (for limiting). Valid = any float less than or equal to Max. Default is 0.0.
Cfg_OutDatalinkMax	REAL	Not Visible	Not Required	Input	Maximum Output Datalink in EU (for limiting). Valid = any float greater than or equal to Min. Default is 100.0.
Cfg_OutDatalinkRawMin	REAL	Not Visible	Not Required	Input	Output Datalink Minimum in Drive (raw) Units (for scaling). Valid = any float not equal to Max. Default is 0.0.
Cfg_OutDatalinkRawMax	REAL	Not Visible	Not Required	Input	Output Datalink Maximum in Drive (raw) Units (for scaling). Valid = any float not equal to Min. Default is 100.0.
Cfg_OutDatalinkEUMin	REAL	Not Visible	Not Required	Input	Output Datalink Minimum in Engineering Units (for scaling). Valid = any float less than Max. Default is 0.0.
Cfg_OutDatalinkEUMax	REAL	Not Visible	Not Required	Input	Output Datalink Maximum in Engineering Units (for scaling). Valid = any float greater than Min. Default is 100.0.
Cfg_StartHornTime	REAL	Not Visible	Not Required	Input	Time in seconds to sound audible on commanded start. Valid = 0.0 to 1000.0 seconds, 0.0 = disabled. Default is 0.0.
Cfg_VirtualRampTime	REAL	Not Visible	Not Required	Input	Time in seconds to ramp speed feedback when Virtualized. Valid = 0.0 to max float. Default is 10.0.
Cfg_FailToStartTime	REAL	Not Visible	Not Required	Input	Time in seconds after Start to receive Run Feedback before Fault. Valid = 0.0 to 2147483.0 seconds. Default is 15.0.
Cfg_FailToStopTime	REAL	Not Visible	Not Required	Input	Time in seconds after Stop to drop Run Feedback before Fault. Valid = 0.0 to 2147483,0 seconds. Default is 15.0.
Cfg_ResetPulseTime	REAL	Not Visible	Not Required	Input	Time in seconds to pulse Out_Reset to clear Motor fault. Valid = 0.0 to 2147483.0 seconds. Default is 2.0.
Cfg_MaxJogTime	REAL	Not Visible	Not Required	Input	Maximum jog time in seconds. Valid = 0.0 to 2147483.0 seconds, 0.0 = unlimited). Default is 0.0.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Cfg_eKeepRef	SINT	Not Visible	Not Required	Input	Ownership of Speed Reference (enumeration): 0 = follows CmdSrc, 1 = Operator, 2 = Program, 3 = External. Default is 0.
Cfg_eKeepStart	SINT	Not Visible	Not Required	Input	Ownership of Start commands (enumeration): 0 = follows CmdSrc, 1 = Operator, 2 = Program, 3 = External. Default is 0.
Cfg_eKeepJog	SINT	Not Visible	Not Required	Input	Ownership of Jog commands (enumeration): 0 = follows CmdSrc, 1 = Operator, 3 = External. Default is 0.
Cfg_eKeepOutDatalink	SINT	Not Visible	Not Required	Input	Ownership of Output Datalink (enumeration): 0 = follows CmdSrc, 1 = Operator, 2 = Program, 3 = External. Default is 0.
Cfg_CnfrmReqd	SINT	Not Visible	Not Required	Input	Operator Command Confirmation Required. Represents the type of command confirmation required. 0 = None, 1 = Command confirmation required, 2 = Performer e-signature required, 3 = Performer and approver e-signature required. Default is 0.
Cfg_HasHistTrend	SINT	Not Visible	Not Required	Input	Has Historical Trend. This enables navigation to the Device Historical Trend Faceplate from the HMI. 0 = No external historical trend, 1 = Datalog historical trend, 2 = Historian historical trend. Default is 0.
PSet_SpeedRef	REAL	Not Visible	Not Required	Input	Program Setting of Run Speed Reference (speed reference engineering units). Valid = any real, will be clamped. Default is 0.0.
PSet_OutDatalink	REAL	Not Visible	Not Required	Input	Program Setting of Output Datalink (output datalink engineering units). Valid = any real, will be clamped. Default is 0.0.
PSet_Owner	DINT	Not Visible	Not Required	Input	Program owner request ID (non-zero) or release (zero). Default is 0.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
XSet_SpeedRef	REAL	Not Visible	Not Required	Input	External setting of Run Speed Reference (speed reference engineering units). Valid = any real, will be clamped. Default is 0.0.
XSet_OutDatalink	REAL	Not Visible	Not Required	Input	External setting of Output Datalink (output datalink engineering units). Valid = any real, will be clamped. Default is 0.0.
PCmd_Virtual	BOOL	Not Visible	Not Required	Input	Program command to select Virtual (simulated) device operation. The instruction clears this operand automatically. Default is false.
PCmd_Physical	BOOL	Not Visible	Not Required	Input	Program command to select Physical device operation (not simulated). The instruction clears this operand automatically. Default is false.
PCmd_StartFwd	BOOL	Not Visible	Not Required	Input	Program command to Start Drive Forward. The instruction clears this operand automatically. Default is false.
PCmd_StartRev	BOOL	Not Visible	Not Required	Input	Program command to Start Drive Reverse. The instruction clears this operand automatically. Default is false.
PCmd_Stop	BOOL	Not Visible	Not Required	Input	Program command to Stop Drive. The instruction clears this operand automatically. Default is false.
PCmd_Prog	BOOL	Not Visible	Not Required	Input	Program command to select Program (Operator to Program). The instruction clears this operand automatically if Cfg_PCmdProgAsLevel = 0. Default is false.
PCmd_Oper	BOOL	Not Visible	Not Required	Input	Program command to select Operator (Program to Operator). The instruction clears this operand automatically. Default is false.
PCmd_Lock	BOOL	Not Visible	Not Required	Input	Program command to lock Program (disallow Operator). The instruction clears this operand automatically if Cfg_PCmdLockAsLevel = 0. Default is false.
PCmd_Unlock	BOOL	Not Visible	Not Required	Input	Program command to unlock Program (allow Operator to acquire). The instruction clears this operand automatically. Default is false.
PCmd_Normal	BOOL	Not Visible	Not Required	Input	Program command to select Normal command source (Operator or Program). The instruction clears this operand automatically. Default is false.
PCmd_Reset	BOOL	Not Visible	Not Required	Input	Program command to reset all alarms and latched shed conditions requiring reset. The instruction clears this operand automatically. Default is false.

Public Input Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
MCmd_Rel	BOOL	Not Visible	Not Required	Input	Maintenance command to release ownership (Maintenance to Operator/Program/External/Override). The instruction clears this operand automatically. Default is false.
OCmd_Unlock	BOOL	Not Visible	Not Required	Input	Operator command to unlock / release (allow Program to acquire) ownership. The instruction clears this operand automatically. Default is false.
XCmd_StartFwd	BOOL	Not Visible	Not Required	Input	External command to Start Drive Forward. The instruction clears this operand automatically. Default is false.
XCmd_StartRev	BOOL	Not Visible	Not Required	Input	External command to Start Drive Reverse. The instruction clears this operand automatically. Default is false.
XCmd_Stop	BOOL	Not Visible	Not Required	Input	External command to Stop Drive. The instruction clears this operand automatically. Default is false.
XCmd_JogFwd	BOOL	Not Visible	Not Required	Input	External command to Jog Drive Forward. The instruction clears this operand automatically if max jog time is reached. Default is false.
XCmd_JogRev	BOOL	Not Visible	Not Required	Input	External command to Jog Drive Reverse. The instruction clears this operand automatically if max jog time is reached.. Default is false.
XCmd_Acq	BOOL	Not Visible	Not Required	Input	External command to acquire ownership (Operator/Program/Override/Maintenance to External). The instruction clears this operand automatically if Cfg_ExtAcqAsLevel = 0. Default is false.
XCmd_Rel	BOOL	Not Visible	Not Required	Input	External command to release ownership if Cfg_ExtAcqAsLevel = 0 (External to Operator/Program/Override/Maintenance). The instruction clears this operand automatically. Default is false.
XCmd_Reset	BOOL	Not Visible	Not Required	Input	External command to clear shed latches and cleared alarms. The instruction clears this operand automatically. Default is false.
XCmd_ResetAckAll	BOOL	Not Visible	Not Required	Input	External command to acknowledge and reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
EnableOut	BOOL	Not Visible	Not Required	Output	Enable Output - System Defined Parameter
Out_SpeedRefData	REAL	Not Visible	Not Required	Output	Speed reference in drive (raw) units (example: 0 to 32767 in drive units represents 0 to max frequency).

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Out_DatalinkData	REAL	Not Visible	Not Required	Output	Auxiliary signal (datalink) output in drive (raw) units.
Out_RunData	BOOL	Not Visible	Not Required	Output	1 = Start/Run Drive, 0 = Stop Drive (for held starter type).
Out_StopData	BOOL	Not Visible	Not Required	Output	1 = Stop Drive, 0 = drive left in current state.
Out_StartData	BOOL	Not Visible	Not Required	Output	1 = Start Drive, 0 = drive left in current state.
Out_ClearFaultData	BOOL	Not Visible	Not Required	Output	1 = Attempt to clear Drive Fault.
Out_FwdData	BOOL	Not Visible	Not Required	Output	1 = Set drive direction to Forward.
Out_RevData	BOOL	Not Visible	Not Required	Output	1 = Set drive direction to Reverse.
Out_HornData	BOOL	Not Visible	Not Required	Output	1 = Sound audible prior to commanded motor start.
Out_Reset	BOOL	Not Visible	Not Required	Output	1 = Reset command has been received and accepted.
Out_OwnerSts	DINT	Not Visible	Not Required	Output	Status of command source, owner command handshake and ready status. 0 = None, .10 = Operator Lock, .11 = Operator Unlock, .12 = Program Lock, .13 = Program Unlock, .14 = Acquire Maintenance, .15 = Release Maintenance, .16 = Acquire External, .17 = Release External, .18 = Has Maintenance, .19 = External Override Lock, .20 = Has External, .21 = Has Operator, .22 = Has Operator Locked, .23 = Has Program, .24 = Has Program Locked, .29 = Echo, .30 = Not Ready.
Val_SpeedRef	REAL	Visible	Not Required	Output	Speed Reference (target) to drive.
Val_SpeedFdbk	REAL	Visible	Not Required	Output	Speed Feedback (actual) from drive.
Val_InpDatalink	REAL	Not Visible	Not Required	Output	Input Datalink value from drive.
Val_OutDatalink	REAL	Not Visible	Not Required	Output	Output Datalink value to drive.
Val_SpeedRefEUMin	REAL	Not Visible	Not Required	Output	Minimum of Speed Reference = MIN (Cfg_SpeedFdbkEUMin, Cfg_SpeedFdbkEUMax).
Val_SpeedRefEUMax	REAL	Not Visible	Not Required	Output	Maximum of Speed Reference = MAX (Cfg_SpeedFdbkEUMin, Cfg_SpeedFdbkEUMax).
Val_SpeedFdbkEUMin	REAL	Not Visible	Not Required	Output	Minimum of Speed Feedback = MIN (Cfg_SpeedFdbkEUMin, Cfg_SpeedFdbkEUMax).
Val_SpeedFdbkEUMax	REAL	Not Visible	Not Required	Output	Maximum of Speed Feedback = MAX (Cfg_SpeedFdbkEUMin, Cfg_SpeedFdbkEUMax).
Sts_Initialized	BOOL	Not Visible	Not Required	Output	1 = Instruction is initialized. Use Inp_InitializeReq to reinitialize.
Sts_Stopped	BOOL	Visible	Not Required	Output	1 = Drive requested to stop and is confirmed stopped.
Sts_StartingFwd	BOOL	Visible	Not Required	Output	1 = Drive requested to run forward and awaiting run feedback.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_StartingRev	BOOL	Not Visible	Not Required	Output	1 = Drive requested to run reverse and awaiting run feedback.
Sts_RunningFwd	BOOL	Visible	Not Required	Output	1 = Drive requested to run and is confirmed running forward.
Sts_RunningRev	BOOL	Not Visible	Not Required	Output	1 = Drive requested to run and is confirmed running reverse.
Sts_StoppingFwd	BOOL	Visible	Not Required	Output	1 = Drive running forward requested to stop and awaiting stopped feedback.
Sts_StoppingRev	BOOL	Not Visible	Not Required	Output	1 = Drive running reverse requested to stop and awaiting stopped feedback.
Sts_JoggingFwd	BOOL	Visible	Not Required	Output	1 = Drive requested to Jog Forward.
Sts_JoggingRev	BOOL	Not Visible	Not Required	Output	1 = Drive requested to Jog Reverse.
Sts_Horn	BOOL	Not Visible	Not Required	Output	1 = Motor Audible Alert (Horn) is Active.
Sts_CommandDir	BOOL	Not Visible	Not Required	Output	1 = Drive commanded to Forward, 0 = Reverse.
Sts_ActualDir	BOOL	Not Visible	Not Required	Output	1 = Motor rotation (actual direction) is Forward, 0 = Reverse.
Sts_Accel	BOOL	Not Visible	Not Required	Output	1 = Drive is Accelerating.
Sts_Decel	BOOL	Not Visible	Not Required	Output	1 = Drive is Decelerating.
Sts_NotReady	BOOL	Not Visible	Not Required	Output	1 = Drive is Not Ready (cannot be started) Check alarms, stops, faults.
Sts_Alarm	BOOL	Not Visible	Not Required	Output	1 = Drive has an Alarm (see drive display or manual).
Sts_AtSpeed	BOOL	Not Visible	Not Required	Output	1 = Drive is running at reference speed.
Sts_SpeedLimited	BOOL	Not Visible	Not Required	Output	1 = Speed Reference Setting exceeds configured Max/Min limit.
Sts_Virtual	BOOL	Not Visible	Not Required	Output	1 = The instruction treats the drive as virtual. The instruction acts as normal but the output is kept de-energized; 0 = The instruction operates the drive normally. Sts_Virtual is a copy of Sts_Virtual.
SrcQ_IO	SINT	Not Visible	Not Required	Output	Source and quality of primary input or output (enumerated): 0=Good, live, confirmed good 1=Good, live, assumed good 2=Good, no feedback, assumed good 8=Test, virtualized 9=Test, loopback 10=Test, manually entered 16=Uncertain, live, off-spec 17=Uncertain, substituted at device or bus 18=Uncertain, substituted at instruction 19=Uncertain, using last known good 20=Uncertain, using replacement value 32=Bad, signal failure 33=Bad, channel fault 34=Bad, module or communication fault 35=Bad, invalid configuration

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
SrcQ	SINT	Not Visible	Not Required	Output	Source and quality of primary value or status (enumerated): 0=Good, live, confirmed good 1=Good, live, assumed good 2=Good, no feedback, assumed good 8=Test, virtualized 9=Test, loopback 10=Test, manually entered 16=Uncertain, live, off-spec 17=Uncertain, substituted at device or bus 18=Uncertain, substituted at instruction 19=Uncertain, using last known good 20=Uncertain, using replacement value 32=Bad, signal failure 33=Bad, channel fault 34=Bad, module or communication fault 35=Bad, invalid configuration
Sts_eCmd	SINT	Not Visible	Not Required	Output	Drive Command: 0 = None, 1 = Stop, 2 = Start Forward, 3 = Start Reverse, 4 = Jog Forward, 5 = Jog Reverse.
Sts_eFdbk	SINT	Not Visible	Not Required	Output	Drive Feedback: 0 = Stopped, 1 = Running Forward, 2 = Running Reverse, 3 = Accelerating, 4 = Decelerating.
Sts_eSts	SINT	Not Visible	Not Required	Output	Drive Status: 0 = Powerup / Unknown, 1 = Stopped, 2 = Running Forward, 3 = Running Reverse, 4 = Starting Forward, 5 = Starting Reverse, 6 = Jogging Forward, 7 = Jogging Reverse, 8 = Stopping, 14 = Horn, 15 = Out Of Service.
Sts_eFault	SINT	Not Visible	Not Required	Output	Drive Fault Status: 0 = None, 15 = Interlock Trip, 16 = Fail to Start, 17 = Fail to Stop, 18 = Drive Fault, 32 = I/O Fault, 34 = Config Error.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_eNotify	SINT	Not Visible	Not Required	Output	All alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyAll	SINT	Not Visible	Not Required	Output	All alarm status enumerated values including related objects: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyIOFault	SINT	Not Visible	Not Required	Output	IOFault alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyFailToStart	SINT	Not Visible	Not Required	Output	Fail to Start alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_eNotifyFailToStop	SINT	Not Visible	Not Required	Output	Fail to Stop alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyIntlkTrip	SINT	Not Visible	Not Required	Output	IntlkTrip alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowledged.
Sts_eNotifyDriveFault	SINT	Not Visible	Not Required	Output	Drive Fault alarm status enumerated values: 0 = Not in alarm, acknowledged, 1 = Not in alarm, unacknowledged or reset required, 2 = Low severity alarm, acknowledged, 3 = Low severity alarm, unacknowledged, 4 = Medium severity alarm, acknowledged, 5 = Medium severity alarm, unacknowledged, 6 = High severity alarm, acknowledged, 7 = High severity alarm, unacknowledged, 8 = Urgent severity alarm, acknowledged, 9 = Urgent severity alarm, unacknowl
Sts_UnackAlmCount	DINT	Not Visible	Not Required	Output	Count of unacknowledged alarms.
Sts_eFaultCode	DINT	Not Visible	Not Required	Output	First Drive Fault Code after reset. See drive manual or Drive Object for description.
Sts_eSrc	INT	Not Visible	Not Required	Output	The current command source is shown with status bits: Sts_eSrc.0: Lock, Sts_eSrc.1: Normal, Sts_eSrc.2: Hand, Sts_eSrc.3: Maintenance, Sts_eSrc.4: Override, Sts_eSrc.5: Program, Sts_eSrc.6: Operator, Sts_eSrc.7: Out of Service, Sts_eSrc.8: External.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_bSrc	INT	Not Visible	Not Required	Output	Active selection bitmap (for HMI totem pole with command source request selection): Sts_bSrc.0: Hand, Sts_bSrc.1: Programmed Out of Service (rung false), Sts_bSrc.2: Maintenance Out of Service, Sts_bSrc.3: Maintenance, Sts_bSrc.4: Override, Sts_bSrc.5: External, Sts_bSrc.6: Program locked, Sts_bSrc.7: Program, Sts_bSrc.8: Operator locked, Sts_bSrc.9: Operator.
Sts_Available	BOOL	Not Visible	Not Required	Output	1 = Device has been acquired by Program and is now available for start/stop control.
Sts_IntlkAvailable	BOOL	Not Visible	Not Required	Output	1 = Device can be acquired by Program and is available for start/stop control when interlocks are OK.
Sts_Bypass	BOOL	Not Visible	Not Required	Output	1 = Bypassable interlocks are bypassed.
Sts_BypActive	BOOL	Visible	Not Required	Output	1 = Interlock bypassing active (bypassed or maintenance).
Sts_MaintByp	BOOL	Not Visible	Not Required	Output	1 = Device has a maintenance bypass function active.
Sts_NotRdy	BOOL	Not Visible	Not Required	Output	1 = Device is not ready, see detail bits (Sts_Nrdyxxx) for reason.
Sts_NrdyCfgErr	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: Configuration error.
Sts_NrdyDriveNotReady	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: Drive Not Ready.
Sts_NrdyFail	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: Device failure (Shed requires Reset).
Sts_NrdyIntlk	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: Interlock not OK.
Sts_NrdyIOFault	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: IO Fault (Shed requires Reset).
Sts_NrdyOoS	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: Device disabled by Maintenance.
Sts_NrdyFwdPerm	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: Forward permissive not OK.
Sts_NrdyRevPerm	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: Reverse permissive not OK.
Sts_NrdyPrioStop	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: Operator or External priority Stop command requires reset.
Sts_NrdyTrip	BOOL	Not Visible	Not Required	Output	1 = Device is not ready: Device Tripped (Drive Fault requires Reset).
Sts_Err	BOOL	Visible	Not Required	Output	1 = Error in configuration: See detail bits (Sts_ErrXxx) for reason.
Sts_ErrAlm	BOOL	Not Visible	Not Required	Output	1 = Error in Logix tag-based alarm settings.
Sts_ErrSpeedFdbkRaw	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Speed Fdbk Raw Min = Max.
Sts_ErrSpeedFdbkEU	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Speed Fdbk EU Min = Max.
Sts_ErrSpeedRefLim	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Speed Ref Limit Min > Max.
Sts_ErrSpeedRefEU	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Speed Ref EU Min = Max.
Sts_ErrSpeedRefRaw	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Speed Ref Raw Min = Max.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_ErrInpDatalinkRaw	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Input Datalink Raw Min = Max.
Sts_ErrInpDatalinkEU	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Input Datalink EU Min = Max.
Sts_ErrOutDatalinkLim	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Output Datalink Limits Min > Max.
Sts_ErrOutDatalinkEU	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Output Datalink EU Min = Max.
Sts_ErrOutDatalinkRaw	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Output Datalink Raw Min = Max.
Sts_ErrVirtualRampTime	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Virtual speed accel / decel time: use 0 to 2147483.
Sts_ErrFailToStartTime	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Fail to Start timer preset: use 0 to 2147483.
Sts_ErrFailToStopTime	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Fail to Stop timer preset: use 0 to 2147483.
Sts_ErrResetPulseTime	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Reset Pulse timer preset: use 0 to 2147483.
Sts_ErrMaxJogTime	BOOL	Not Visible	Not Required	Output	1 = Error in Config: Maximum Jog Time timer preset: use 0 to 2147483.
Sts_Hand	BOOL	Visible	Not Required	Output	1 = Hand is selected (supersedes OoS, Maintenance, Override, External, Program, Operator).
Sts_OoS	BOOL	Visible	Not Required	Output	1 = Out of Service is selected (supersedes Maintenance, Override, External, Program, Operator).
Sts_Maint	BOOL	Visible	Not Required	Output	1 = Maintenance is selected (supersedes Override, External, Program, Operator).
Sts_Ovrd	BOOL	Visible	Not Required	Output	1 = Override is selected (supersedes External, Program, Operator).
Sts_Ext	BOOL	Visible	Not Required	Output	1 = External is selected (supersedes Program and Operator).
Sts_Prog	BOOL	Visible	Not Required	Output	1 = Program is selected.
Sts_ProgLocked	BOOL	Not Visible	Not Required	Output	1 = Program is selected and Locked.
Sts_Oper	BOOL	Visible	Not Required	Output	1 = Operator is selected.
Sts_OperLocked	BOOL	Not Visible	Not Required	Output	1 = Operator is selected and Locked.
Sts_ProgOperSel	BOOL	Not Visible	Not Required	Output	Program/Operator selection (latch) state: 1 = Program, 0 = Operator.
Sts_ProgOperLock	BOOL	Visible	Not Required	Output	Program/Operator lock (latch) state, 1 = Locked, 0 = Unlocked.
Sts_Normal	BOOL	Not Visible	Not Required	Output	1 = Selection equals the Normal (Program or Operator).
Sts_ExtReqInh	BOOL	Not Visible	Not Required	Output	1 = External request inhibited, cannot get to External from current state.
Sts_ProgReqInh	BOOL	Not Visible	Not Required	Output	1 = Program request inhibited, cannot get to Program from current state.
Sts_MAcqRcvd	BOOL	Not Visible	Not Required	Output	1 = Maintenance Acquire command received this scan.
Sts_CmdConflict	BOOL	Not Visible	Not Required	Output	1 = Conflicting commands received this scan.
Sts_Alm	BOOL	Not Visible	Not Required	Output	1 = An alarm is active.
Sts_AlmInh	BOOL	Not Visible	Not Required	Output	1 = An alarm is shelved or disabled.

Public Output Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Sts_IOFault	BOOL	Not Visible	Not Required	Output	IO Fault status: 1 = Bad, 0 = OK. There is a predefined default discrete Logix tag-based alarm for the status. Set standard configuration members of the discrete Logix tag-based alarm. Alarm elements can be accessed as follows: PVSDTag.@Alarms.Alm_IOFault.AlarmElement.
Sts_FailToStart	BOOL	Not Visible	Not Required	Output	1 = Drive failed to Start. There is a predefined default discrete Logix tag-based alarm for the status. Set standard configuration members of the discrete Logix tag-based alarm. Alarm elements can be accessed as follows: PVSDTag.@Alarms.Alm_FailToStart.AlarmElement .
Sts_FailToStop	BOOL	Not Visible	Not Required	Output	1 = Drive failed to Stop. There is a predefined default discrete Logix tag-based alarm for the status. Set standard configuration members of the discrete Logix tag-based alarm. Alarm elements can be accessed as follows: PVSDTag.@Alarms.Alm_FailToStop.AlarmElement.
Sts_IntlkTrip	BOOL	Not Visible	Not Required	Output	1 = Drive stopped by an interlock Not OK. There is a predefined default discrete Logix tag-based alarm for the status. Set standard configuration members of the discrete Logix tag-based alarm. Alarm elements can be accessed as follows: PVSDTag.@Alarms.Alm_IntlkTrip.AlarmElement.
Sts_DriveFault	BOOL	Not Visible	Not Required	Output	1 = Drive Fault, see drive display or manual for detail.
Sts_RdyAck	BOOL	Not Visible	Not Required	Output	1 = An alarm is ready to be acknowledged.
Sts_RdyReset	BOOL	Not Visible	Not Required	Output	1 = A latched alarm or shed condition is ready to be reset.
XRdy_Acq	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Acq, enable HMI button.
XRdy_Rel	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Rel, enable HMI button.
XRdy_StartFwd	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_StartFwd, enable button.
XRdy_StartRev	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_StartRev, enable button.
XRdy_JogFwd	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_JogFwd, enable button.
XRdy_JogRev	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_JogRev, enable button.
XRdy_Stop	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Stop, enable button.
XRdy_Reset	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_Reset, enable button.
XRdy_ResetAckAll	BOOL	Not Visible	Not Required	Output	1 = Ready for XCmd_ResetAckAll, enable button.
Val_Owner	DINT	Not Visible	Not Required	Output	Current object owner ID (0 = not owned).

Private Input Members	Data Type	Description
HMI_BusObjIndex	DINT	This object's index in the bus array, for use by HMI display. Default is 0.
MCmd_Acq	BOOL	Maintenance command to acquire ownership (Operator/Program/External/Override to Maintenance). The instruction clears this operand automatically. Default is false.
MCmd_Bypass	BOOL	Maintenance command to bypass all bypassable interlocks and permissives. The instruction clears this operand automatically. Default is false.

Private Input Members	Data Type	Description
MCmd_Check	BOOL	Maintenance command to check (not bypass) all interlocks and permissives. The instruction clears this operand automatically. Default is false.
MCmd_IS	BOOL	Maintenance command to select In Service. The instruction clears this operand automatically. Default is false.
MCmd_OoS	BOOL	Maintenance command to select Out of Service. The instruction clears this operand automatically. Default is false.
MCmd_Physical	BOOL	Maintenance command to select Physical device operation (not simulated). The instruction clears this operand automatically. Default is false.
MCmd_Virtual	BOOL	Maintenance command to select Virtual (simulated) device operation. The instruction clears this operand automatically. Default is false.
OCmd_JogFwd	BOOL	Operator command to Jog Drive Forward. The instruction clears this operand automatically if max jog time is reached. Default is false.
OCmd_JogRev	BOOL	Operator command to Jog Drive Reverse. The instruction clears this operand automatically if max jog time is reached. Default is false.
OCmd_Lock	BOOL	Operator command to lock Operator (disallow Program). The instruction clears this operand automatically. Default is false.
OCmd_Normal	BOOL	Operator command to select Normal (Operator or Program). The instruction clears this operand automatically. Default is false.
OCmd_Oper	BOOL	Operator command to select Operator (Program to Operator). The instruction clears this operand automatically. Default is false.
OCmd_Prog	BOOL	Operator command to select Program (Operator to Program). The instruction clears this operand automatically. Default is false.
OCmd_Reset	BOOL	Operator command to reset all alarms and latched shed conditions. The instruction clears this operand automatically. Default is false.
OCmd_ResetAckAll	BOOL	Operator command to acknowledge and reset all alarms and latched shed conditions. The use of OCmd_ResetAckAll is restricted to HMI. The instruction clears this operand automatically. Default is false.
OCmd_StartFwd	BOOL	Operator command to Start Drive Forward. The instruction clears this operand automatically. Default is false.
OCmd_StartRev	BOOL	Operator command to Start Drive Reverse. The instruction clears this operand automatically. Default is false.
OCmd_Stop	BOOL	Operator command to Stop Drive. The instruction clears this operand automatically. Default is false.
OSet_OutDatalink	REAL	Operator Setting of Output Datalink, in output datalink engineering units. Valid = any float (clamped) Default is 0.0.

Private Input Members	Data Type	Description
OSet_SpeedRef	REAL	Operator Setting of Speed Reference, in speed reference engineering units. Valid = any float (clamped) Default is 0.0.

Private Output Members	Data Type	Description
MRdy_Acq	BOOL	1 = Ready for MCmd_Acq, enable HMI button.
MRdy_Bypass	BOOL	1 = Ready for MCmd_Bypass, enable HMI button.
MRdy_Check	BOOL	1 = Ready for MCmd_Check, enable HMI button.
MRdy_IS	BOOL	1 = Ready for MCmd_IS, enable HMI button.
MRdy_OoS	BOOL	1 = Ready for MCmd_OoS, enable HMI button.
MRdy_Physical	BOOL	1 = Ready for MCmd_Physical, enable HMI button.
MRdy_Rel	BOOL	1 = Ready for MCmd_Rel, enable HMI button.
MRdy_Virtual	BOOL	1 = Ready for MCmd_Virtual, enable HMI button.
ORdy_JogFwd	BOOL	1 = Ready for OCmd_JogFwd, enable HMI button.
ORdy_JogRev	BOOL	1 = Ready for OCmd_JogRev, enable HMI button.
ORdy_Lock	BOOL	1 = Ready for OCmd_Lock, enable HMI button.
ORdy_Normal	BOOL	1 = Ready for OCmd_Normal, enable HMI button.
ORdy_Oper	BOOL	1 = Ready for OCmd_Oper, enable HMI button.
ORdy_OutDatalink	BOOL	1 = Ready for OSet_OutDatalink, enable data entry field.
ORdy_Prog	BOOL	1 = Ready for OCmd_Prog, enable HMI button.
ORdy_Reset	BOOL	1 = A latched alarm or shed condition is ready to be reset.
ORdy_ResetAckAll	BOOL	1 = A latched alarm or shed condition is ready to be reset or acknowledged.
ORdy_SpeedRef	BOOL	1 = Ready for OSet_SpeedRef, enable data entry field.
ORdy_StartFwd	BOOL	1 = Ready for OCmd_StartFwd, enable HMI button.
ORdy_StartRev	BOOL	1 = Ready for OCmd_StartRev, enable HMI button.
ORdy_Stop	BOOL	1 = Ready for OCmd_Stop, enable HMI button.
ORdy_Unlock	BOOL	1 = Ready for OCmd_Unlock, enable HMI button.
Sts_FaultDesc	STRING	Description of motor controller fault, lookup from last fault code.

Public InOut Members	Data Type	FBD Default Visibility	FBD Wiring required	Usage	Description
Ref_Ctrl_Set	RAC_ITF_DVC_PWRVELOCITY_SET	Visible	Required	InOut	Velocity Automation Device Object Settings Interface.
Ref_Ctrl_Cmd	RAC_ITF_DVC_PWRVELOCITY_CMD	Visible	Required	InOut	Velocity Automation Device Object Command Interface.
Ref_Ctrl_Sts	RAC_ITF_DVC_PWRVELOCITY_STS	Visible	Required	InOut	Velocity Automation Device Object Status Interface.
BusObj	BUS_OBJ	Visible	Required	InOut	Bus component.
Ref_FaultCodeList	RAC_CODE_DESCRIPTION[1]	Visible	Required	InOut	Fault Code to Fault Description lookup table for intelligent motor controller.

RAC_ITF_DVC_PWRVELOCITY_SET Structure

The RAC_ITF_DVC_PWRVELOCITY_SET structure is the first of three structures exchanged with the associated Power Velocity Device Object to

interface with the variable speed drive device. This structure handles settings, such as the speed reference, sent to the drive.

This parameter links the instruction to an external tag that contains necessary data for the instruction to operate. The external tag must be of the data type shown, or may be NULL. If NULL, other pins on the instruction are used to link the necessary data.

Members	Data Type	Description
InhibitCmd	BOOL	1 = Inhibit user Commands from external sources, 0 = Allow Commands.
InhibitSet	BOOL	1 = Inhibit user Settings from external sources, 0 = Allow Settings.
Speed	REAL	Speed reference (Hz). Valid = 0.0 to maximum drive frequency.

RAC_ITF_DVC_PWRVELOCITY_CMD Structure

The RAC_ITF_DVC_PWRVELOCITY_CMD structure is the second of three structures exchanged with the associated Power Velocity Device Object to interface with the variable speed drive device. This structure handles commands, such as start, stop and jog, sent to the drive. It is an InOut parameter configured as optional (May Be Null).

This parameter links the instruction to an external tag that contains necessary data for the instruction to operate. The external tag must be of the data type shown, or may be NULL. If NULL, other pins on the instruction are used to link the necessary data.

Members	Data Type	Description
bCmd	INT	Commands (Bit Overlay):
Physical	BOOL	Operate as a physical device.
Virtual	BOOL	Operate as a virtual device.
ResetWarn	BOOL	Reset warning status.
ResetFault	BOOL	Reset fault status.
Activate	BOOL	Activate output power structure.
Deactivate	BOOL	Deactivate output power structure.
CmdDir	BOOL	Select direction: 0 = Forward, 1 = Reverse.

RAC_ITF_DVC_PWRVELOCITY_STS Structure

The RAC_ITF_DVC_PWRVELOCITY_STS structure is the third of three structures exchanged with the associated Power Velocity Device Object to interface with the variable speed drive device. This structure handles status, such as the speed feedback, active status, commanded and actual direction, received from the drive. It is an InOut parameter configured as optional (May Be Null).

This parameter links the instruction to an external tag that contains necessary data for the instruction to operate. The external tag must be of the data type

shown, or may be NULL. If NULL, other pins on the instruction are used to link the necessary data.

Members	Data Type	Description
eState	DINT	Enumerated state of the device object: 0 = Unused, 1 = Initializing, 2 = Disconnected, 3 = Disconnecting, 4 = Connecting, 5 = Idle, 6 = Configuring, 7 = Available.
FirstWarning	RAC_ITF_EVENT	First warning.
FirstFault	RAC_ITF_EVENT	First fault.
eCmdFail	DINT	Enumerated command failure code.
Speed	REAL	Actual Speed (Hz).
bSts	INT	Status (Bit Overlay):
Physical	BOOL	1 = Operating as a physical device.
Virtual	BOOL	1 = Operating as a virtual device.
Connected	BOOL	1 = Connected and communicating.
Available	BOOL	1 = Device is configured and can be operated.
Warning	BOOL	1 = Device has a warning.
Faulted	BOOL	1 = Device is faulted.
Ready	BOOL	1 = Device is ready to be activated.
Active	BOOL	1 = Device is active (power structure active, drive running).
ZeroSpeed	BOOL	1 = Motor is at zero speed (not rotating).
ObjCtrl	BOOL	0 = Object has control of this device, 1 = Object does not have control of this device (for example, local HIM or I/O has control).
CmdDir	BOOL	Commanded direction: 1 = Reverse, 0 = Forward
ActDir	BOOL	Actual (rotation) direction: 1 = Reverse, 0 = Forward
Accelerating	BOOL	1 = Motor is accelerating.
Decelerating	BOOL	1 = Motor is decelerating.
AtSpeed	BOOL	1 = Motor actual speed has reached speed reference.

BUS_OBJ Structure

The BUS_OBJ structure links the variable speed drive to other devices and instructions in a complex control strategy, typically into a hierarchy. A Bus Object rolls up status and alarm information from lower level devices to higher level control and fans out commands from higher level control to lower level devices, and items link to the bus by referencing a single member of the BUS_OBJ array associated with the bus.

This parameter links the instruction to an external tag that contains necessary data for the instruction to operate. The external tag must be of the data type shown, or may be NULL. If NULL, the Bus functions of this instruction are not available.

Members	Data Type	Description
Inp_Cmd	DINT	Input to assert commands
Out_Cmd	DINT	Resultant commands
Inp_CmdLLH	DINT	Input for level normally high commands
Out_CmdLLH	DINT	Resultant line level high commands
Inp_Sts	DINT	Input to assert status
Out_Sts	DINT	Resultant status
Inp_CmdAck	DINT	Input to assert a command acknowledgement
Out_CmdAck	DINT	Resultant command acknowledgements
Inp_SeverityMax	DINT	Input: maximum alarm severity
Out_SeverityMax	DINT	Resultant of maximum alarm severity
Cfg_CmdMask	DINT	Propagation mask for commands
Cfg_CmdLLHMask	DINT	Propagation mask for line level high commands
Cfg_StsMask	DINT	Propagation mask for status
Ref_Index	DINT	Bus array index

RAC_CODE_DESCRIPTION[x] Structure

The RAC_CODE_DESCRIPTION[x] structure is an array of drive fault code number and fault code description pairs, used as a lookup table. The instruction searches the table for the fault code received from the drive and displays the corresponding fault description text.

This parameter links the instruction to an external tag that contains necessary data for the instruction to operate. The external tag must be of the data type shown, or may be NULL. If NULL, the fault code lookup function is not performed. Fault descriptions will only be shown if provided through the Device Object Status interface.

Members	Data Type	Description
Code	DINT	Code for which to look up Description.
Desc	STRING	Description for given Code.

RAC_EVENTStructure

RAC_EVENTstructures are used by the FirstFault and FirstWarning members in the RAC_ITF_DVC_PWRVELOCITY_STS structure. These items hold the event data received from the drive for the first drive fault and first drive warning records in the drive event history.

Members	Data Type	Description
Type	DINT	Event type: 1 = Status, 2 = Warning, 3 = Fault, 4 ...n = User.
ID	DINT	User-definable event ID.

Members	Data Type	Description
Category	DINT	User-definable category (Electrical, Mechanical, Materials, Utility, etc.).
Action	DINT	User-definable event action code.
Value	DINT	User-definable event value or fault code.
Message	STRING	Event message text.
EventTime_L	LINT	Event timestamp (64-bit microseconds format).
EventTime_D	DINT[7]	Event timestamp (year, month, day, hour, minute, second, microsecond format).

Alarms

Discrete Logix tag-based alarms are defined for these members:

Member	Alarm Name	Description
Sts_FailToStart	Alm_FailToStart	Drive failed to start within the allotted time when commanded to start.
Sts_FailToStop	Alm_FailToStop	Drive failed to stop within the allotted time when commanded to stop.
Sts_IntlkTrip	Alm_IntlkTrip	Drive stopped by an Interlock Not OK.
Sts_IOFault	Alm_IOFault	Drive communication with controller failed.
Sts_DriveFault	Alm_DriveFault	The variable speed drive is reporting it has a fault condition.

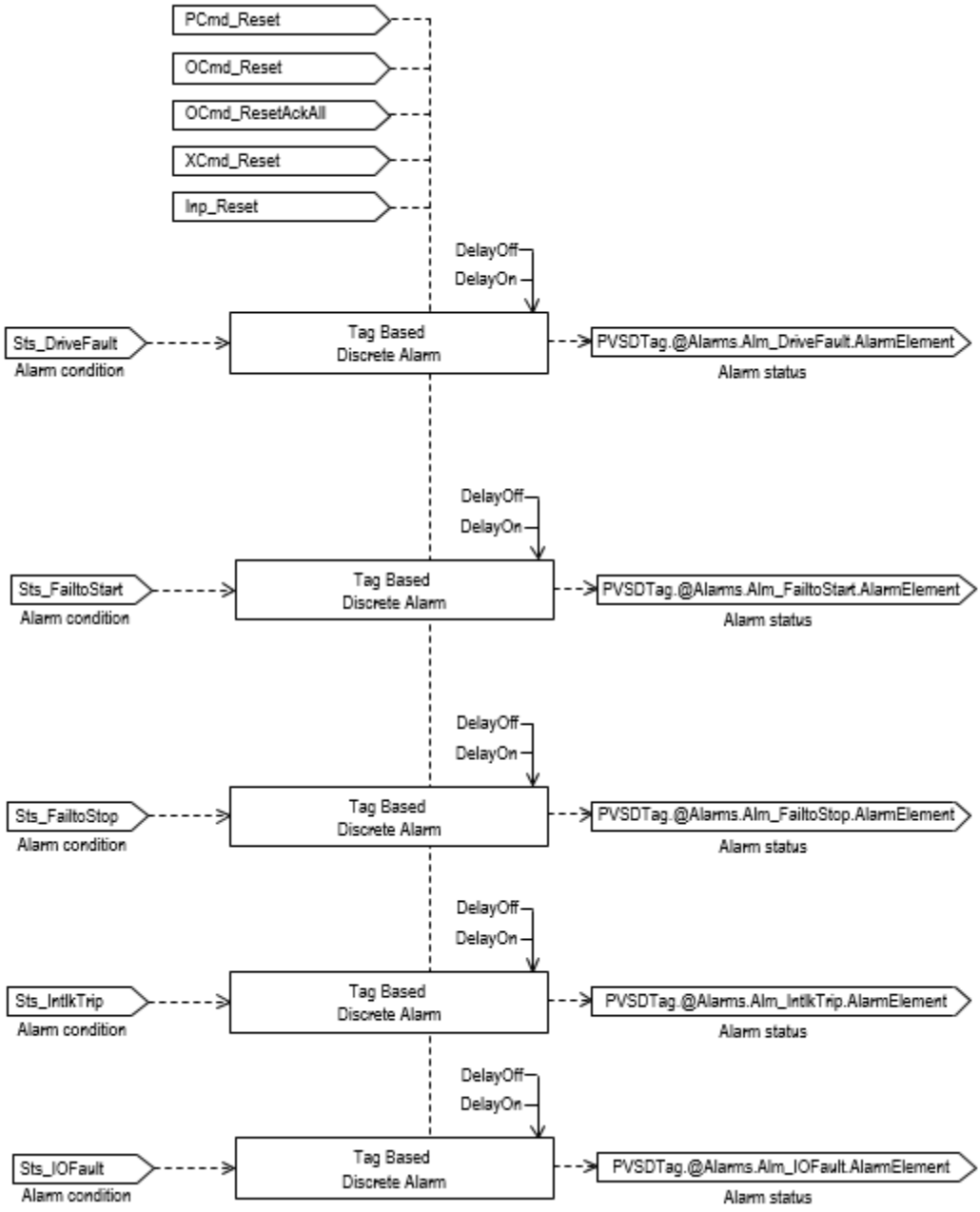
Mark the alarm as used or unused and set standard configuration members of the discrete Logix Tag based alarm. Use this format to access alarm elements:

Tag.@Alarms.AlarmName.AlarmElement

The PVSD instruction uses these alarms:

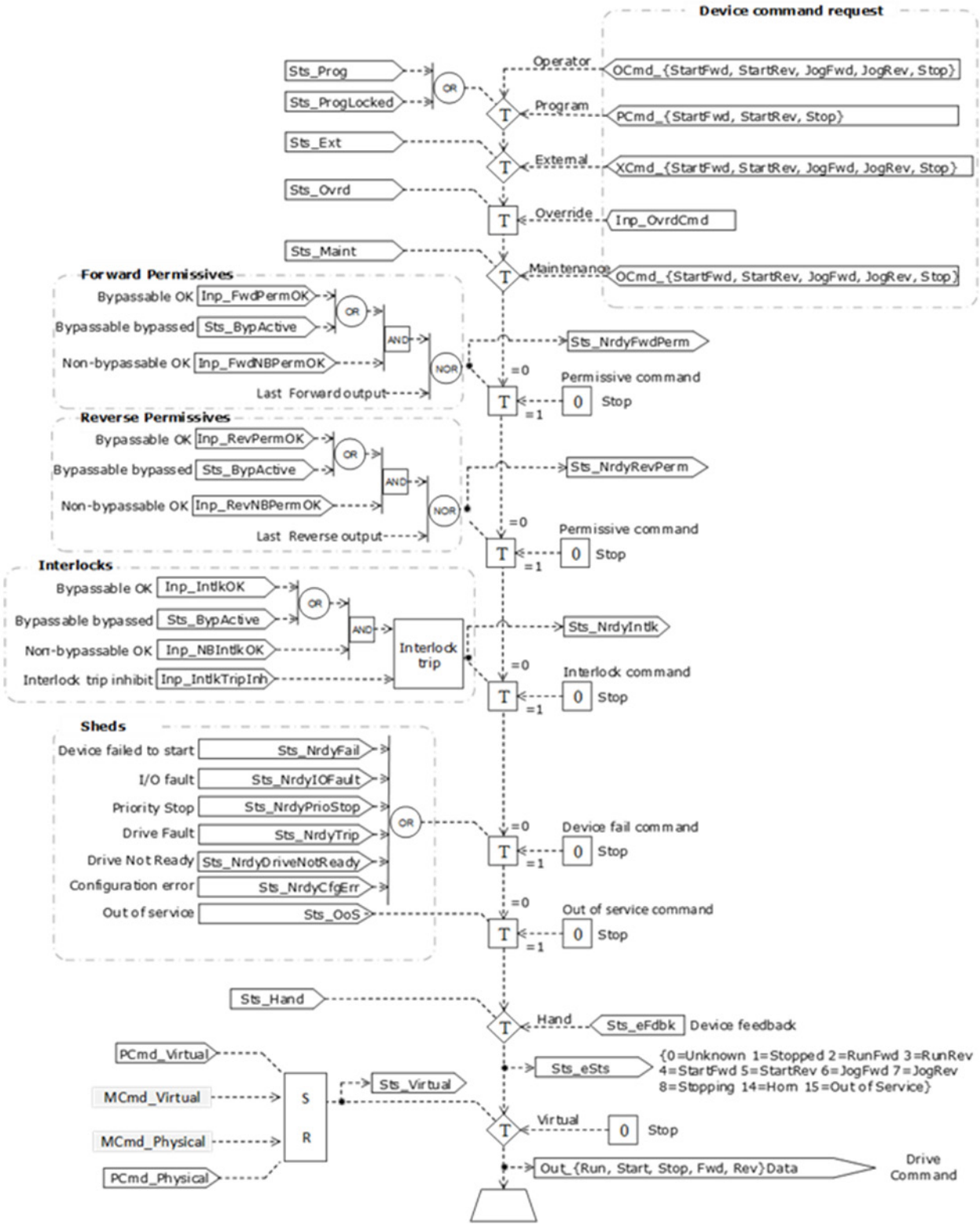
- Raises the Fail to Start alarm when the drive is commanded to start but run feedback is not received within the configured failure time.
- Raises the Fail to Stop alarm when the drive is commanded to stop but run feedback does not drop within the configured failure time.
- Raises the Interlock Trip alarm when the motor is running and an interlock not-OK condition causes the motor to stop. If interlocks are not bypassed, a bypassable interlock or a non-bypassable interlock not-OK condition initiates an interlock trip. If interlocks are bypassed, only a non-bypassable interlock not-OK condition initiates an interlock trip.
- Raises the I/O Fault alarm when I/O communication with the variable speed drive is lost. For the Power Velocity Device interface, this is detected when the Ref_Ctrl_Sts.Connected bit goes false (to 0). For the discrete signal interface, used when Ref_Ctrl_Sts is NULL, this is detected when Inp_IOFault goes true (to 1).
- Raises the Drive Fault alarm when the drive reports a drive faulted condition. For the Power Velocity Device interface, this is detected when the Ref_Ctrl_Sts.Faulted bit goes true (to 1). For the discrete signal interface, which is used when Ref_Ctrl_Sts is NULL, this is detected when Inp_Faulted goes true (to 1).

Program, Operator, and External commands reset latched alarms, and reset and acknowledge all alarms of the instruction (Alarm Set) at the same time. This diagram shows how the commands interact with the PVSD instruction.

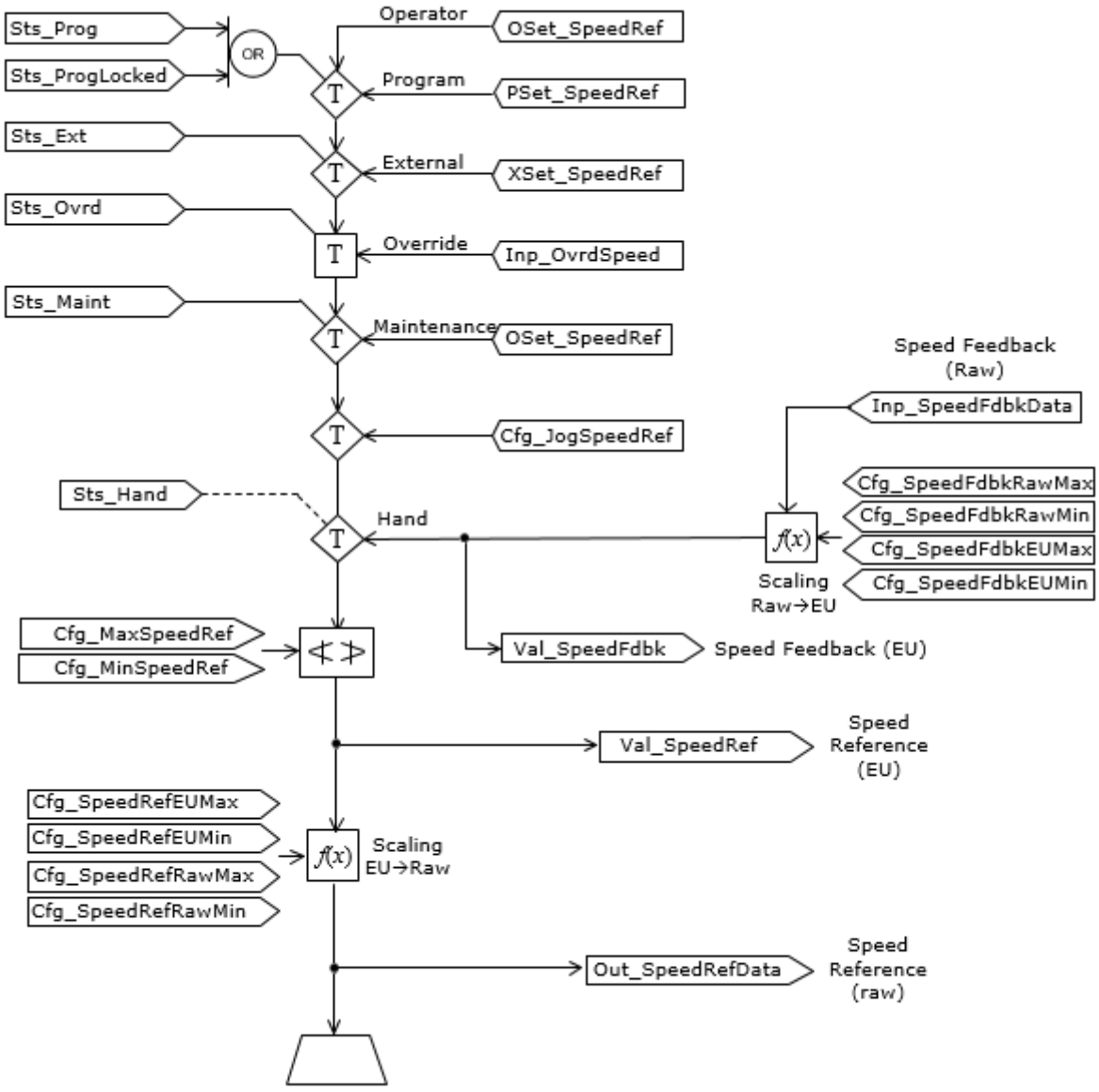


Operation

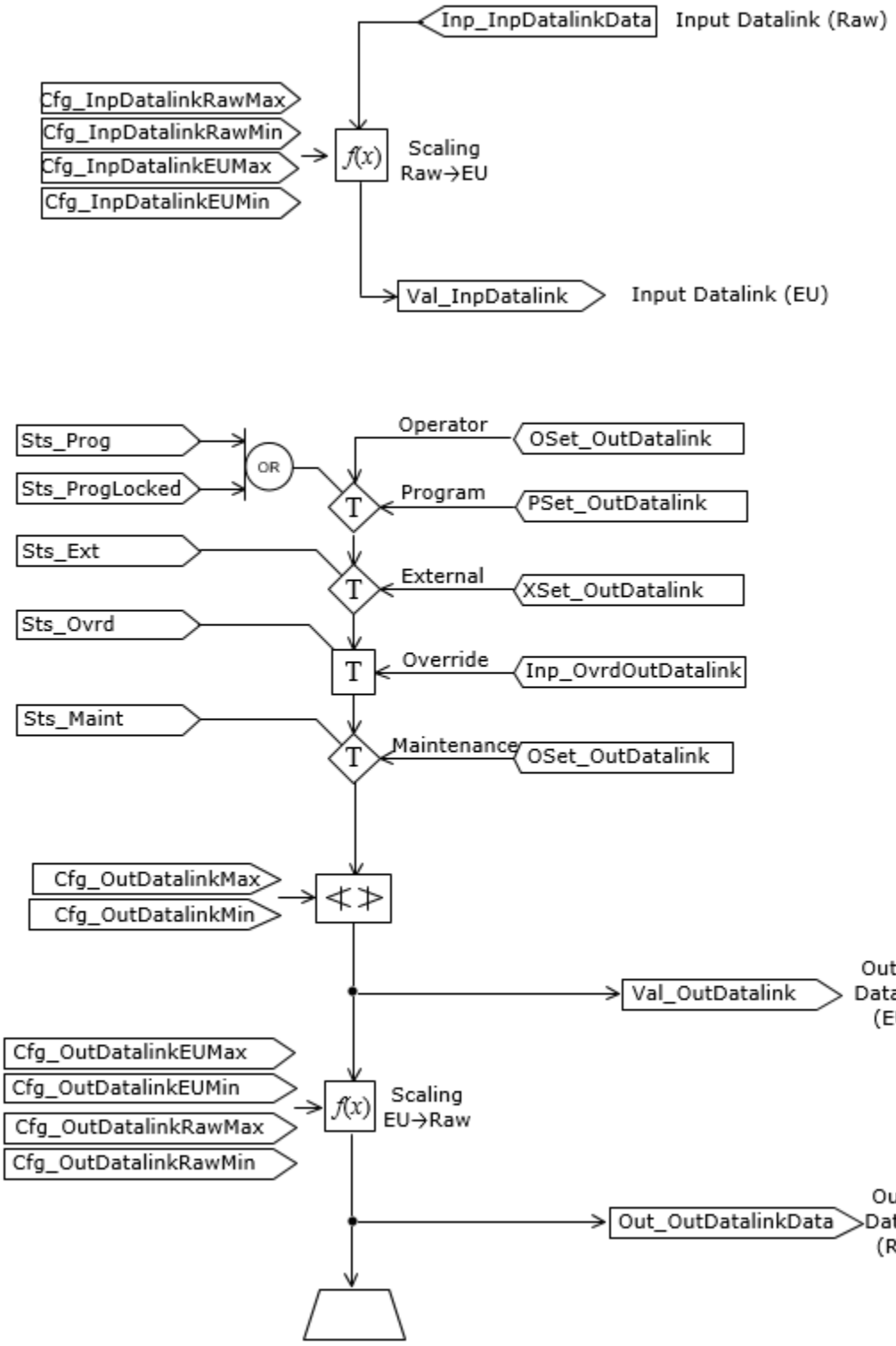
This diagram illustrates functionality of the PVSD instruction:



The second diagram illustrates the handling of the drive speed reference (setpoint) and drive speed feedback:



The third diagram illustrates the handling of the optional input datalink and output datalink functions:



Monitor the PVSD Instruction

Use the operator faceplate from the PlantPAx library of Process objects for monitoring.

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Any commands received before first scan are discarded. The motor is de-energized and treated as if it were commanded to stop.
Instruction first run	Any commands received before first scan are discarded. The motor state is evaluated and the instruction aligns with the current state of the motor, as if the Hand command source were selected.
Rung-condition-in is false	Handled the same as if the motor is taken Out of Service by command. The motor outputs are de-energized, and the motor Command Source is shown as Program Out of Service on the HMI. All alarms are cleared. The rung-condition-out continues as false.
Rung-condition-in is true	Set rung-condition-out to rung-condition-in. The instruction executes.
Postscan	Rung-condition-out is cleared to false.

Function Block Diagram

Condition/State	Action Taken
Prescan	Any commands received before first scan are discarded. The motor is de-energized and treated as if it were commanded to stop.
Instruction first run	Any commands received before first scan are discarded. The motor state is evaluated and the instruction aligns with the current state of the motor, as if the Hand command source were selected.
Instruction first scan	See instruction first run in the function block diagram table.
EnableIn is false	Handled the same as if the motor is taken Out of Service by command. The motor outputs are de-energized, and the motor Command Source is shown as Program Out of Service on the HMI. All alarms are cleared. EnableOut is set to false.
EnableIn is true	EnableOut is set to true. The instruction executes.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

In Structured Text, EnableIn is always true during normal scan. Therefore, if the instruction is in the control path activated by the logic it will execute.

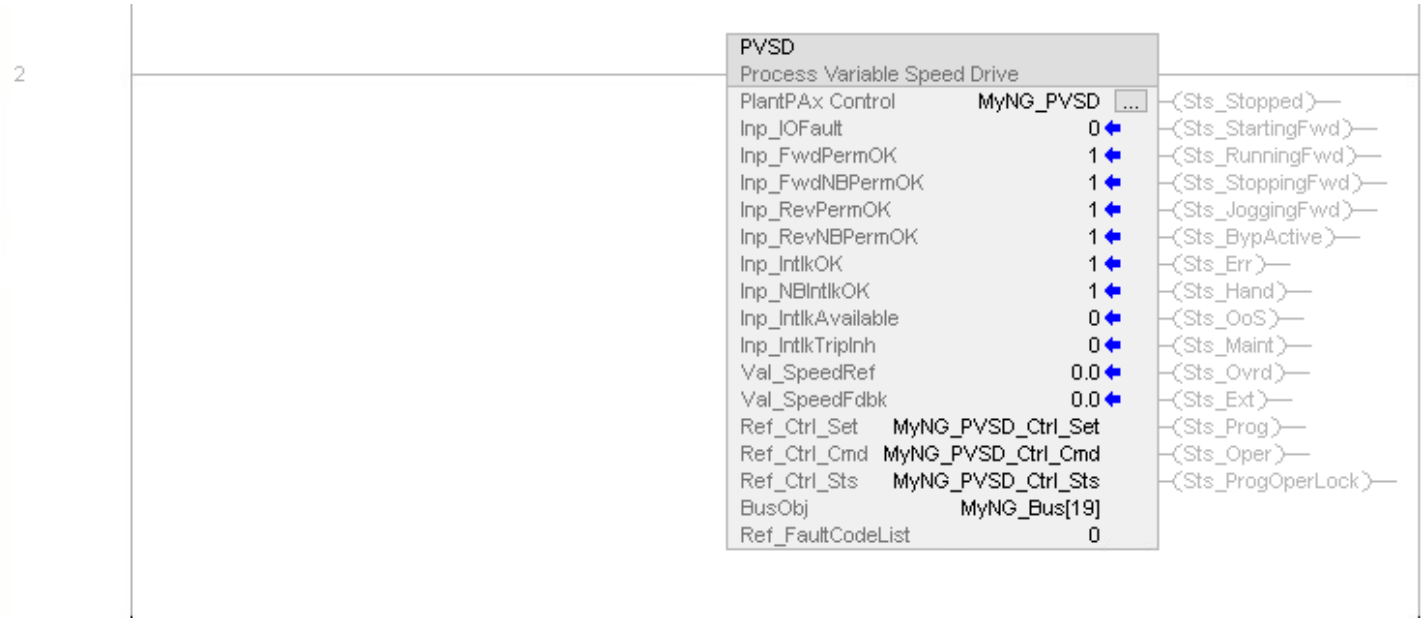
Condition/State	Action Taken
Prescan	See Prescan in the Function Block Diagram table.
Instruction first run	See Instruction first run in the Function Block Diagram table.
EnableIn is true	See EnableIn is true in the Function Block Diagram table.
Postscan	See Postscan in the Function Block Diagram table.

Example

In the following example, the first three reference (InOut parameter) tags are used to interface to a Power Velocity Device object, provided by Commercial Engineering. These tags provide the Speed Reference setting to the drive, the various activate (start), deactivate (stop), and clear fault commands to the drive, and drive status from the drive, including status such as active (running), actual speed, commanded and actual direction, and fault and warning information and test. The next InOut parameter links this drive to a bus of related devices, control modules, equipment modules and unit(s) in a hierarchy used to fan out commands and roll up status and alarm information for use in complex control strategies and sequences.

In this example, the Fault Code List parameter is not used because the textual fault information is received from the Power Velocity Device object, and no lookup from a fault code is required in this instance.

Ladder Diagram



Function Block Diagram

Structured Text

PVSD(MyNG_PVSD, MyNG_PVSD_Ctrl_Set, MyNG_PVSD_Ctrl_Cmd, MyNG_PVSD_Ctrl_Sts, MyNG_Bus[19], 0)

See also

Process Variable Speed Drive (PVSD) Command Source on page 816

Data Conversions on page 1086

Index Through Arrays on page 1094

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

Process Variable Speed Drive (PVSD) Command Source

The Process Variable Speed Drive (PVSD) instruction uses these command sources. The command sources are prioritized in order from highest to lowest.

Command Source	Description
Hand	Hardwired logic or other logic outside the instruction controls the device. The instruction tracks the state of the device for bumpless transfer back to one of the other command sources. This is the highest priority command source.
Out-of-Service	The instruction is disabled. Drive commands and settings from any source are not accepted.
Maintenance	Maintenance controls the device and supersedes Operator, Program, External and Override control. Operator commands and settings from the HMI are accepted.
Override	Priority logic controls the device and supersedes Operator, Program and External control. Override Input (Inp_Ovrd) is accepted.
External	External logic (for example, field pilot control or upstream SCADA) controls the device. External commands (XCmd_) are accepted.
Program locked	Program logic controls the device. Program commands (PCmd_) are accepted. Operator cannot take control from the Program. Override cannot take control from theProgram unless Cfg_OvrdOverLock = 1.
Program	Program logic controls the device. Program commands (PCmd_) are accepted.
Operator locked	The Operator controls the device. Operator commands (OCmd_) from the HMI are accepted. Program cannot take control from Operator. Override cannot take control from Operator unless Cfg_OvrdOverLock = 1.
Operator	The Operator controls the device. Operator commands (OCmd_) from the HMI are accepted. This is the lowest priority command source.

The instruction enables or disables these operations:

- Maintenance Out-of-Service exists
- Maintenance exists
- External exists
- Program (unlocked) exists
- Program locked exists
- Operator (unlocked) exists
- Operator locked exists
- PCmd_Lock used as a Level (1 = Lock, 0 = Unlock)

The instruction checks for invalid configurations of control and forces the nearest valid configuration.

The core control model arbitrates the source of the commands and parameters that the receiving function accepts. The core control model determines if the source is:

- A programmatic entity which resides entirely within the processing environment, or
- An external interface entity which issues commands and parameters external and asynchronously to the processing environment.

Locking a control source prevents the other control source from acquiring privilege.

Core Command Source Model

The core control model consists of these control sources:

- Oper
- OperLocked
- Prog
- ProgLocked

The control model defaults to this configuration. Other control sources may be present in the model but act as overriding control sources, acting independent of the base Operator/Program state machine.

Enable control sources as Configuration

The user can enable and disable individual control sources. The default configuration uses the entire base model; upon power-up of the processing environment the control source will be the designated default. Some combinations of enabled control sources are disallowed as they are either unnecessary or could create unintended changes.

Prog Power Up

Configuration allows the user to specify whether Operator or Program is the power-up default.

Prog Priority

Configuration allows the user to specify whether Operator or Program commands take priority when simultaneously asserted.

Automatic reset of commands

All commands are treated as one-shot-latched. Commands are automatically cleared when the instruction executes and processes them.

Change Destination States

Under certain configurations the destination command source for some commands may change. This is in keeping with the intent of the command. For example, if the Program state is disabled, the destination of the OCmd_Prog command becomes the Program Locked state instead of the Program state. This maintains the intent of the OCmd_Prog command: the operator entity wishes to place the function in control of the program. If the

command was eliminated then there would be no way to accomplish this. This is only done in configurations where it would cause no conflict or race condition, but serves to preserve as much user functionality as is practical.

Higher Priority Command Sources

These Higher priority command sources operate independently within the model:

- External
- Override
- Maintenance
- Out-of-Service
- In-Service
- Hand

Implementation

The PVSD instruction monitors and controls a variable speed drive. The speed reference and the start, stop and jog commands to the drive can come from a variety of sources, determined by an embedded instance of PCMDSRC. Available command sources are:

- Operator, through the HMI
- Program, through logic connected to the block
- External, through logic connected to the block
- Override, through logic connected to the block
- Maintenance, through the HMI
- Out of Service
- Hand (assumes the block has no control of the drive, so aligns with the actual drive status in order to achieve bumpless transfer from Hand back to one of the other command sources)

The PVSD instruction has four aspects, which can be kept by a particular command source whenever the command source selection is Operator, Program or External. Any or all of the aspects can be kept at any given time, or can follow the selection of the PCMDSRC. The aspects are:

- The Speed Reference setting
- Start (forward and reverse) commands
- Jog (forward and reverse) commands
- The Output Datalink setting

The Jog commands cannot be kept by the Program command source.

The PVSD instruction supports virtualization. When selected to Virtual, the instruction provides status to the operator and other blocks as if a working drive were connected while keeping the outputs to the physical drive de-

energized (zero). When selected to Physical, the instruction monitors and controls the physical variable speed drive device. Use Virtualization to provide off-process functional testing of higher-level control strategies or simulation for operator training.

The PVSD instruction supports interlocks, conditions that must be OK for the motor to run and which stop the motor if not OK, and permissives, conditions that must be OK for the motor to start but which are ignored once the motor is running. Bypassable permissives and interlocks can be bypassed for maintenance, while non-bypassble interlocks and permissives are always evalutated.

Analog values (speed reference, speed feedback, input datalink, output datalink) associated with the drive are displayed and entered in engineering units, and linear scaling is used to provide these values in raw (drive) units at the drive interface. The speed reference to the drive has rate limiting and clamping limits; the output datalink to the drive has clamping limits.

The PVSD instruction supports a bus for forwarding commands (fanout) and gathering status (rollup) in a hierarchy of objects. Refer to the Bus Object for more information on the commands and status (including alarm status) sent on the bus.

The PVSD instruction optionally supports the ability to look up the text to display for the most recent drive fault code, given a provided fault code lookup table. This table is an array of Code and Description pairs and is searched whenever the last fault code from the drive changes.

The PVSD instruction's interface to the physical drive can be through a Power Velocity Device Object interface or by connecting individual drive signals to input and output pins of the instruction. Details on the Power Velocity Device Object interface are given below. Three interface tags are used, provided as InOut Parameters. These tags provide drive Settings, such as the Speed Reference, drive Commands, such as start forward, jog reverse and stop, and retrieve drive Status, such as connected, active (running), commanded direction, actual direction, accelerating, decelerating, at speed, warning, faulted, and extended drive warning and fault information.

PVSD Drive Settings: Ref_Ctrl_Set InOut Parameter (RAC_ITF_DVC_PWRVELOCITY_SET) Structure

Private Input Members	Data Type	Description
InhibitCmd	BOOL	1 = Inhibit user Commands from external sources, 0 = allow Commands.
InhibitSet	BOOL	1 = Inhibit user Settings from external sources, 0 = Allow Settings.
Speed	REAL	Speed reference (Hz). Valid = 0.0 to maximum drive frequency.

PVSD Drive Commands: Ref_Ctrl_Cmd InOut Parameter (RAC_ITF_DVC_PWRVELOCITY_CMD) Structure

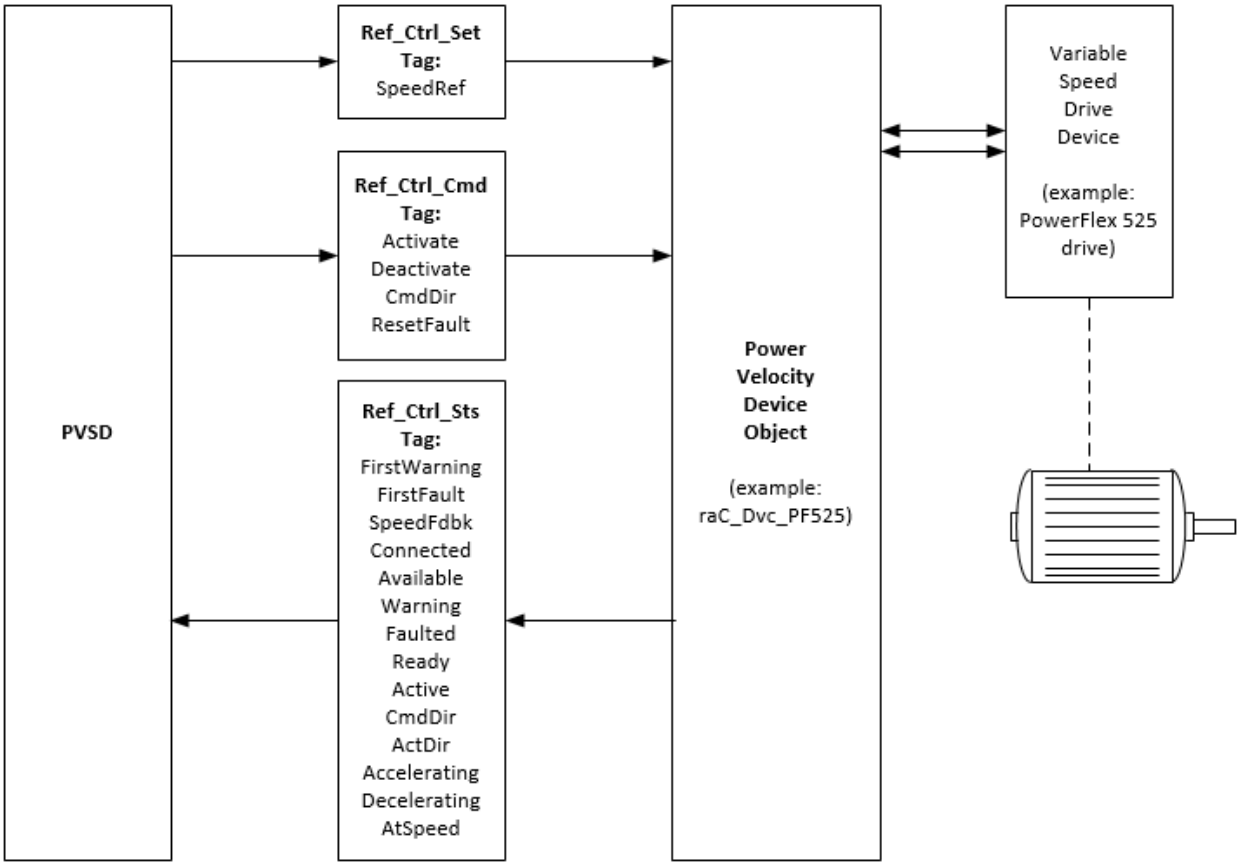
Private Input Members	Data Type	Description
bCmd	INT	Commands (bit overlay), consisting of:
Physical	BOOL	Operate as a physical device
Virtual	BOOL	Operate as a virtual device
ResetWarn	BOOL	Reset warning status
ResetFault	BOOL	Reset fault status
Activate	BOOL	Activate output power structure (if speed reference is not zero, the motor will run)
Deactivate	BOOL	Deactivate output power structure (motor will stop)
CmdDir	BOOL	Select direction: 0 = Forward, 1 = Reverse

PVSD Drive Status: Ref_Ctrl_Sts InOut Parameter (RAC_ITF_DVC_PWRVELOCITY_STS) Structure

Private Input Members	Data Type	Description
eState	DINT	Enumerated state of the device object: 0 = Unused 1 = Initializing 2 = Disconnected 3 = Disconnecting 4 = Connecting 5 = Idle 6 = Configuring 7 = Available
FirstWarning	RAC_ITF_EVENT	First Warning, consisting of:
Type	DINT	Event type: 1 = Status, 2 = Warning, 3 = Fault, 4 ... n = User
ID	DINT	User-definable event ID
Category	DINT	User-definable category (electrical, mechanical, materials, utility, etc.)
Action	DINT	User-definable event action code
Value	DINT	User-definable event value or fault code
Message	STRING	Event message text
EventTime_L	LINT	Event timestamp (64-bit microseconds format)
EventTime_D	DINT[7]	Event timestamp (year, month, day, hour, minute, second, microsecond format)
FirstFault	RAC_ITF_EVENT	First Fault, consisting of:
Type	DINT	Event type: 1 = Status, 2 = Warning, 3 = Fault, 4 ... n = User
ID	DINT	User-definable event ID
Category	DINT	User-definable category (electrical, mechanical, materials, utility, etc.)
Action	DINT	User-definable event action code
Value	DINT	User-definable event value or fault code
Message	STRING	Event message text
EventTime_L	LINT	Event timestamp (64-bit microseconds format)
EventTime_D	DINT[7]	Event timestamp (year, month, day, hour, minute, second, microsecond format)
eCmdFail	DINT	Enumerated command failure code
Speed	REAL	Actual speed (Hz)
bSts	INT	Status, consisting of:
Physical	BOOL	1 = Operating as a physical device

Virtual	BOOL	1 = Operating as a virtual device
Connected	BOOL	1 = Connected and communicating
Available	BOOL	1 = Device is configured and can be operated
Warning	BOOL	1 = Device has a warning
Faulted	BOOL	1 = Device is faulted
Ready	BOOL	1 = Device is ready to be activated
Active	BOOL	1 = Device is active (power structure active, drive running)
ZeroSpeed	BOOL	1 = Motor is at zero speed (not rotating)
ObjCtrl	BOOL	0 = Object has control of this device, 1 = Object does not have control of this device (for example, local HIM or I/O has control)
CmdDir	BOOL	Commanded direction: 1 = reverse, 0 = forward
ActDir	BOOL	Actual direction (of rotation): 1 = reverse, 0 = forward
Accelerating	BOOL	1 = Motor is accelerating
Decelerating	BOOL	1 = Motor is decelerating
AtSpeed	BOOL	1 = Motor is at commanded speed

This illustration shows the relationship between a PVSD instance and its associated Power Velocity Device Object.



See also

Process Variable Speed Drive (PVSD)

Drives

Drives Instructions

The Drives instructions include the following information.

Available Instructions

Ladder Diagram

Not available

Function Block and Structured Text

INTG	PI	PMUL	SCRV	SOC	UPDN	HMIBC

If you want to	Use this instruction
Execute an integral operation.	INTG
Execute a PI algorithm.	PI
Provide an interface from a position input module, such as a resolver or encoder feedback module, to the digital system by computing the change in input from one scan to the next.	PMUL
Perform a ramp function with an added jerk rate	SCRV
Use a gain term, a first order lag, and a second order lead.	SOC
Add and subtract two inputs into an accumulated value.	UPDN
Enable operators to initiate machine control operations, such as jogging a motor or enabling a valve, with a high degree of accuracy and determinism.	HMIBC

See also

Filter Instructions on page 875

Logical and Move Instructions on page 955

Process Control Instructions on page 17

Select/Limit Instructions on page 903

Statistical Instructions on page 935

Integrator (INTG)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380,

CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

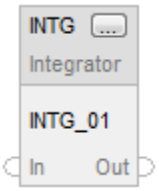
The INTG instruction implements an integral operation. This instruction is designed to execute in a task where the scan rate remains constant.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

INTG(INTG_tag);

Operands

Function Block

Operand	Type	Format	Description
INTG tag	INTEGRATOR	Structure	INTG structure

INTEGRATOR Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0

Input Parameter	Data Type	Description
Initialize	BOOL	Request to initialize control algorithm. Output = InitialValue as long as Initialize is set. Valid = any float Default = 0.0
InitialValue	REAL	The initial value for instruction. Output = InitialValue as long as Initialize is set. Valid = any float Default = 0.0
IGain	REAL	The integral gain multiplier. If IGain < 0; the instruction sets IGain = 0.0, sets the appropriate bit in Status, and leaves the Output unchanged. Valid = 0.0 to maximum positive float Default = 0.0
HighLimit	REAL	The high limit value for Out. If HighLimit ≤ LowLimit, the instruction sets HighAlarm and LowAlarm, sets the appropriate bit in Status, and sets Out = LowLimit. Valid = any float Default = maximum positive float
LowLimit	REAL	The low limit value for Out. If HighLimit ≤ LowLimit, the instruction sets HighAlarm and LowAlarm, sets the appropriate bit in Status, and sets Out = LowLimit. Valid = any float Default = maximum negative float
HoldHigh	BOOL	Hold output high request. When set, Out is not allowed to increase in value. Default is cleared.
HoldLow	BOOL	Hold output low request. When set, Out is not allowed to decrease in value. Default is cleared.
TimingMode	DINT	Selects timing execution mode. 0 = Periodic mode 1 = Oversample mode 2 = Real time sampling mode For more information about timing modes, see Function Block Attributes. Valid = 0 to 2 Default = 0
OversampleDT	REAL	Execution time for oversample mode. Valid = 0 to 4194.303 seconds Default = 0
RTSTime	DINT	Module update period for real time sampling mode Valid = 1 to 32,767ms Default = 1

Input Parameter	Data Type	Description
RTSTimeStamp	DINT	Module time stamp value for real time sampling mode. Valid = 0 to 32,767ms Default = 0

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if Out overflows.
Out	REAL	The calculated output of the algorithm.
HighAlarm	BOOL	High limit alarm indicator. When Out ≥ HighLimit, HighAlarm is set and the output is limited to the value of HighLimit.
LowAlarm	BOOL	Low limit alarm indicator. When Out ≤ LowLimit, LowAlarm is set and the output is limited to the value of LowLimit.
DeltaT	REAL	Elapsed time between updates. This is the elapsed time in seconds used by the control algorithm to calculate the process output.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
IGainInv (Status.1)	BOOL	IGain > maximum or IGain < minimum.
HighLowLimsInv (Status.2)	BOOL	HighLimit ≤ LowLimit.
TimingModeInv (Status.27)	BOOL	Invalid TimingMode value. For more information about timing modes, see Function Block Attributes.
RTSMissed (Status.28)	BOOL	Only used in real time sampling mode. Set when ABS | DeltaT - RTSTime | > 1 (.001 second).
RTSTimeInv (Status.29)	BOOL	Invalid RTSTime value.
RTSTimeStampInv (Status.30)	BOOL	Invalid RTSTimeStamp value.
DeltaTInv (Status.31)	BOOL	Invalid DeltaT value.

Structured Text

Operand	Type	Format	Description
INTG tag	INTEGRATOR	Structure	INTG structure

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Description

The INTG instruction is designed to execute in a task where the scan rate remains constant.

The INTG instruction executes this control algorithm when Initialize is cleared and DeltaT > 0.

$$Out = IGain \times \frac{In + In_{n-1}}{2} \times DeltaT + Out_{n-1}$$

Whenever the value computed for the output is invalid, NAN, or ± INF, the instruction sets Out = the invalid value. The internal parameters are not updated. In each subsequent scan, the output is computed using the internal parameters from the last scan when the output was valid.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	The internal parameters and Out are set to 0.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

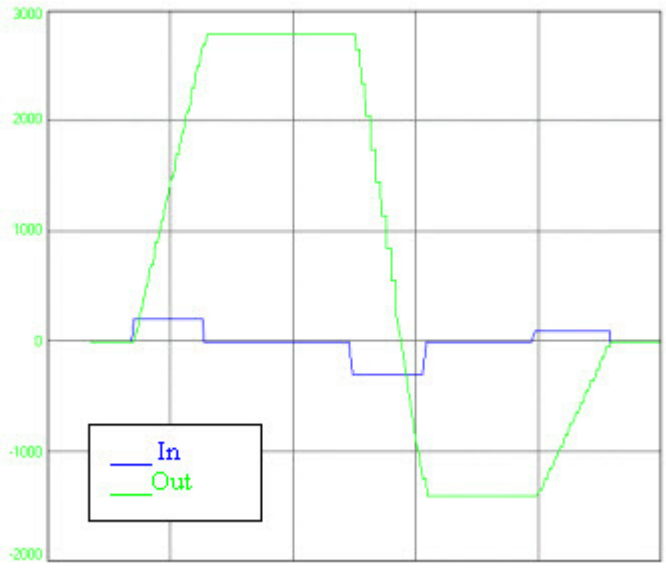
Examples

In many applications an integral gain component is included in the closed loop regulator design in order to eliminate or minimize error in the system being regulated. A straight proportional-only regulator will not tend to drive error in the system to zero. A regulator that uses proportional and integral gain, however, tends to drive the error signal to zero over a period of time. The INTG instruction uses the following equation to calculate its output.

$$Out = IGain \times \frac{In + In_{n-1}}{2} \times DeltaT + Out_{n-1}$$

In the chart below, the input to the block moves from 0 to +200 units. During this period, the output of the block integrates to 2800 units. As In changes from +200 units to 0 units, Out maintains at 2800 units. When In transitions from 0 to -300 units, Out slowly integrates down to -1400 units until In transitions back to 0. Finally, as In moves from 0 to +100, Out integrates back to 0 where In is set to 0 coincidentally with Out reaching 0.

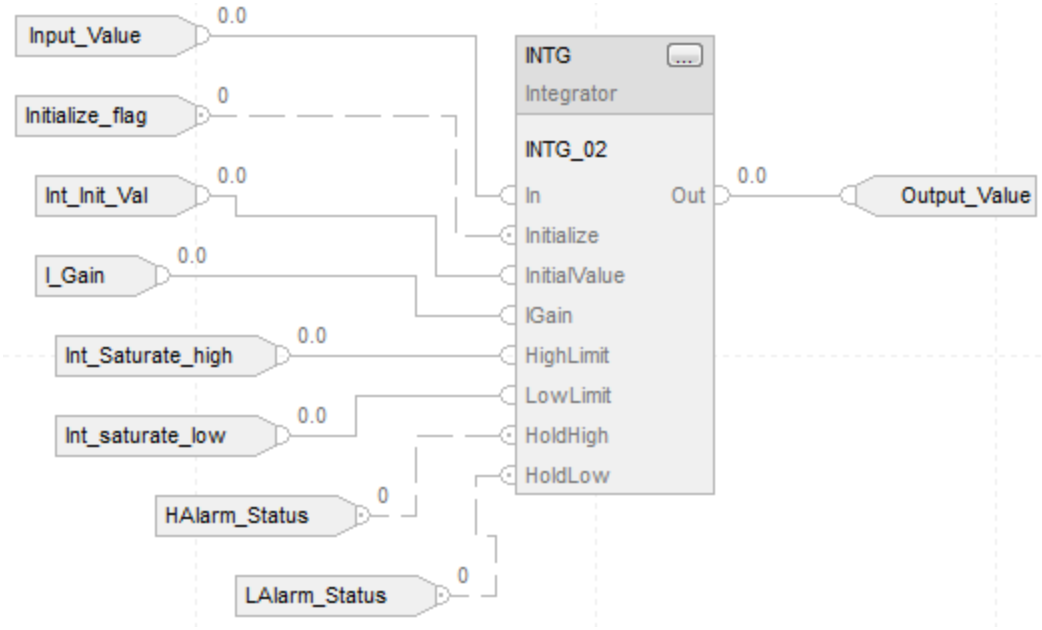
This characteristic of the integrator - continually driving in a specific direction while any input to the function is present or holding at any level during the point where the input is at zero - is what causes a regulator using integral gain to drive toward zero error over a period of time.



The following example shows how the INTG instruction can be used. In many instances, the HighLimit and LowLimit inputs limit the total percentage of control that the integral gain element might have as a function of the regulator's total output. The HoldHigh and HoldLow inputs, on the other hand, can be used to prevent the output from moving further in either the positive or negative direction. The HoldHigh and HoldLow inputs prevent the INTG instruction from "winding-up" in a direction which is already beyond the limits of the controlled variable.

Function Block

This example is the minimal legal programming of the INTG function block and is only used to show the neutral text and generated code for this instruction. This is for internal purposes only and is not a testable case.



Structured Text

INTG_01.In := Input_Value;

INTG_01.Initialize := Initialize_flag;

INTG_01.InitialValue := Int_Init_Val;

INTG_01.IGain := I_Gain;

INTG_01.HighLimit := Int_saturate_high;

INTG_01.LowLimit := Int_saturate_low;

INTG_01.HoldHigh :=HAlarm_Status;

INTG_01.HoldLow := LAlarm_Status;

INTG(INTG_01);

See also

Function Block Attributes on page 1043

Common Attributes on page 1083

Structured Text Syntax on page 1057

Proportional + Integral (PI)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

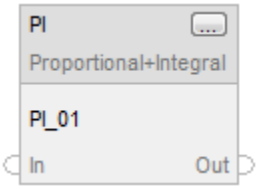
The PI instruction provides two methods of operation. The first method follows the conventional PI algorithm in that the proportional and integral gains remain constant over the range of the input signal (error). The second method uses a non-linear algorithm where the proportional and integral gains vary over the range of the input signal. The input signal is the deviation between the setpoint and feedback of the process.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram.

Function Block



Structured Text

PI(PI_tag);

Operands

Function Block

Operand	Type	Format	Description
PI tag	PROP_INT	structure	PI structure

Structured Text

Operand	Type	Format	Description

PI tag	PROP_INT	structure	PI structure

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

PROP_INT Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
In	REAL	The process error signal input. This is the difference between setpoint and feedback. Valid = any float Default = 0.0
Initialize	BOOL	The instruction initialization command. When set, Out and internal integrator are set equal to the value of InitialValue. Default is cleared.
InitialValue	REAL	The initial value input. When Initialize is set, Out and integrator are set to the value of InitialValue. The value of InitialValue is limited using HighLimit and LowLimit. Valid = any float Default = 0
Kp	REAL	The proportional gain. This affects the calculated value for both the proportional and integral control algorithms. If invalid, the instruction clamps Kp at the limits and sets the appropriate bit in Status. Valid = any float > 0.0 Default = minimum positive float
Wld	REAL	The lead frequency in radians/second. This affects the calculated value of the integral control algorithm. If invalid, the instruction clamps Wld at the limits and sets the appropriate bit in Status. Valid = see the Description section below for valid ranges Default = 0.0
HighLimit	REAL	The high limit value. This is the maximum value for Out. If HighLimit $\leq$ LowLimit, the instruction sets HighAlarm and LowAlarm, sets the appropriate bit in Status, and sets Out = LowLimit. Valid = LowLimit < HighLimit $\leq$ maximum positive float Default = maximum positive float
LowLimit	REAL	The low limit value. This is the minimum value for Out. If HighLimit $\leq$ LowLimit, the instruction sets HighAlarm and LowAlarm, sets the appropriate bit in Status, and sets Out = LowLimit. Valid = maximum negative float $\leq$ LowLimit < HighLimit Default = maximum negative float
HoldHigh	BOOL	The hold high command. When set, the value of the internal integrator is not allowed to increase in value. Default is cleared.
HoldLow	BOOL	The hold low command. When set, the value of the internal integrator is not allowed to decrease in value. Default is cleared.
ShapeKpPlus	REAL	The positive Kp shaping gain multiplier. Used when In is $\geq$0. If invalid, the instruction clamps ShapeKpPlus at the limits and sets the appropriate bit in Status. Not used when NonLinearMode is cleared. Valid = 0.1 to 10.0 Default = 1.0

Input Parameter	Data Type	Description
ShapeKpMinus	REAL	The negative Kp shaping gain multiplier. Used when In is < 0. If invalid, the instruction clamps ShapeKpMinus at the limits and sets the appropriate bit in Status. Not used when NonLinearMode is cleared. Valid = 0.1 to 10.0 Default = 1.0
KpInRange	REAL	The proportional gain shaping range. Defines the range of In (error) over which the proportional gain increases or decreases as a function of the ratio of | In | / KpInRange. When | In | > KpInRange, the instruction calculates the change in proportional error using entered the Kp shaping gain x (In - KpInRange). If invalid, the instruction clamps KpInRange at the limits and sets the appropriate bit in Status. Not used when NonLinearMode is cleared. Valid = any float > 0.0 Default = maximum positive float
ShapeWldPlus	REAL	The positive Wld shaping gain multiplier. Used when In is ≥ 0. If invalid, the instruction clamps ShapeWldPlus at the limits and sets the appropriate bit in Status. Not used when NonLinearMode is cleared. Valid = 0.0 to 10.0 Default = 1.0
ShapeWldMinus	REAL	The negative Wld shaping gain multiplier. Used when In is < 0. If invalid, the instruction clamps ShapeWldMinus at the limits and sets the appropriate bit in Status. Not used when NonLinearMode is cleared. Valid = 0.0 to 10.0 Default = 1.0
WldInRange	REAL	The integral gain shaping range. Defines the range of In (error) over which integral gain increases or decreases as a function of the ratio of | In | / WldInRange. When |In| > WldInRange, the instruction limits In to WldInRange when calculating integral error. If invalid, the instruction clamps WldInRange at the limits and sets the appropriate bit in Status. Not used when NonLinearMode is cleared. Valid = any float > 0.0 Default = maximum positive float
NonLinearMode	BOOL	Enable the non-linear gain mode. When set, the instruction uses the non-linear gain mode selected by ParabolicLinear to compute the actual proportional and integral gains. When cleared, the instruction disables the non-linear gain mode and uses the Kp and Wld values as the proportional and integral gains. Default is cleared.
ParabolicLinear	BOOL	Selects the non-linear gain mode. The modes are linear or parabolic. When set, the instruction uses the parabolic gain method of $y=a * x^2 + b$ to calculate the actual proportional and integral gains. If cleared, the instruction uses the linear gain method of y=a * x + b. Default is cleared.
TimingMode	DINT	Selects timing execution mode. 0 = Periodic mode 1 = Oversample mode 2 = Real time sampling mode For more information about timing modes, see Function Block Attributes. Valid = 0 to 2 Default = 0
OversampleDT	REAL	Execution time for oversample mode. Valid = 0 to 4194.303 seconds Default = 0

Input Parameter	Data Type	Description
RTSTime	DINT	Module update period for real time sampling mode Valid = 1 to 32,767ms Default = 1
RTSTimeStamp	DINT	Module time stamp value for real time sampling mode. Valid = 0 to 32,767ms Default = 0

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates the execution status of the instruction.
Out	REAL	The calculated output of the PI algorithm. Math status flags are set for this output.
HighAlarm	BOOL	The maximum limit alarm indicator. Set when the calculated value for Out ≥ HighLimit and the output and integrator are clamped at HighLimit.
LowAlarm	BOOL	The minimum limit alarm indicator. Set when the calculated value for Out ≤ LowLimit and output and integrator are clamped at LowLimit.
DeltaT	REAL	Elapsed time between updates. This is the elapsed time in seconds used by the control algorithm to calculate the process output.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
KpInv (Status.1)	BOOL	Kp < minimum or Kp > maximum.
WIdInv (Status.2)	BOOL	WId < minimum or WId > maximum.
HighLowLimsInv (Status.3)	BOOL	HighLimit ≤ LowLimit.
ShapeKpPlusInv (Status.4)	BOOL	ShapeKpPlus < minimum or ShapeKpPlus > maximum.
ShapeKpMinusInv (Status.5)	BOOL	ShapeKpMinus < minimum or ShapeKpMinus > maximum.
KpInRangeInv (Status.6)	BOOL	KpInRange < minimum or KpInRange > maximum.
ShapeWIdPlusInv (Status.7)	BOOL	ShapeWIdPlus < minimum or ShapeWIdPlus > maximum.
ShapeWIdMinusInv (Status.8)	BOOL	ShapeWIdMinus < minimum or ShapeWIdMinus > maximum.
WIdInRangeInv (Status.9)	BOOL	WIdInRange < minimum or WIdInRange > maximum.
TimingModeInv (Status.27)	BOOL	Invalid TimingMode. For more information about timing modes, see Function Block Attributes.
RTSMissed (Status.28)	BOOL	Only used in real time sampling mode. Set when ABS | DeltaT - RTSTime | > 1 (.001 second).
RTSTimeInv (Status.29)	BOOL	Invalid RTSTime value.
RTSTimeStampInv (Status.30)	BOOL	Invalid RTSTimeStamp value.
DeltaT (Status.31)	BOOL	Invalid DeltaT value.

Description

The PI instruction uses the position form of the PI algorithm. This means the gain terms are applied directly to the input signal, rather than to the change in the input signal. The PI instruction is designed to execute in a task where the scan rate remains constant.

In the non-linear algorithm, the proportional and integral gains vary as the magnitude of the input signal changes. The PI instruction supports two non-linear gain modes: linear and parabolic. In the linear algorithm, the gains vary linearly as the magnitude of input changes. In the parabolic algorithm, the gains vary according to a parabolic curve as the magnitude of input changes.

The PI instruction calculates Out using this equation:

$$Kp \times \frac{s + Wld}{s}$$

Whenever the value computed for the output is invalid, NAN, or ± INF, the instruction sets Out = the invalid value and sets the math overflow status flag. The internal parameters are not updated. In each subsequent scan, the output is computed using the internal parameters from the last scan when the output was valid.

Operating in Linear Mode

In linear mode, the non-linear gain mode is disabled. The Kp and Wld values are the proportional and integral gains used by the instruction. The instruction calculates the value for Out using these equations:

Value	Equation
ITerm	$Kp \times Wld \times \frac{WldInput + WldInput_{n-1}}{2} \times DeltaT + ITerm_{n-1}$ where DeltaT is in seconds
PTerm	*Kp x In*
Out	*ITerm + PTerm*

with these limits on Wld:

- Low Limit > 0.0
- High Limit = 0.7π/DeltaT
- WldInput = In

Operating in Non-Linear Mode

In non-linear mode, the instruction uses the non-linear gain mode selected by ParabolicLinear to compute the actual proportional and integral gains.

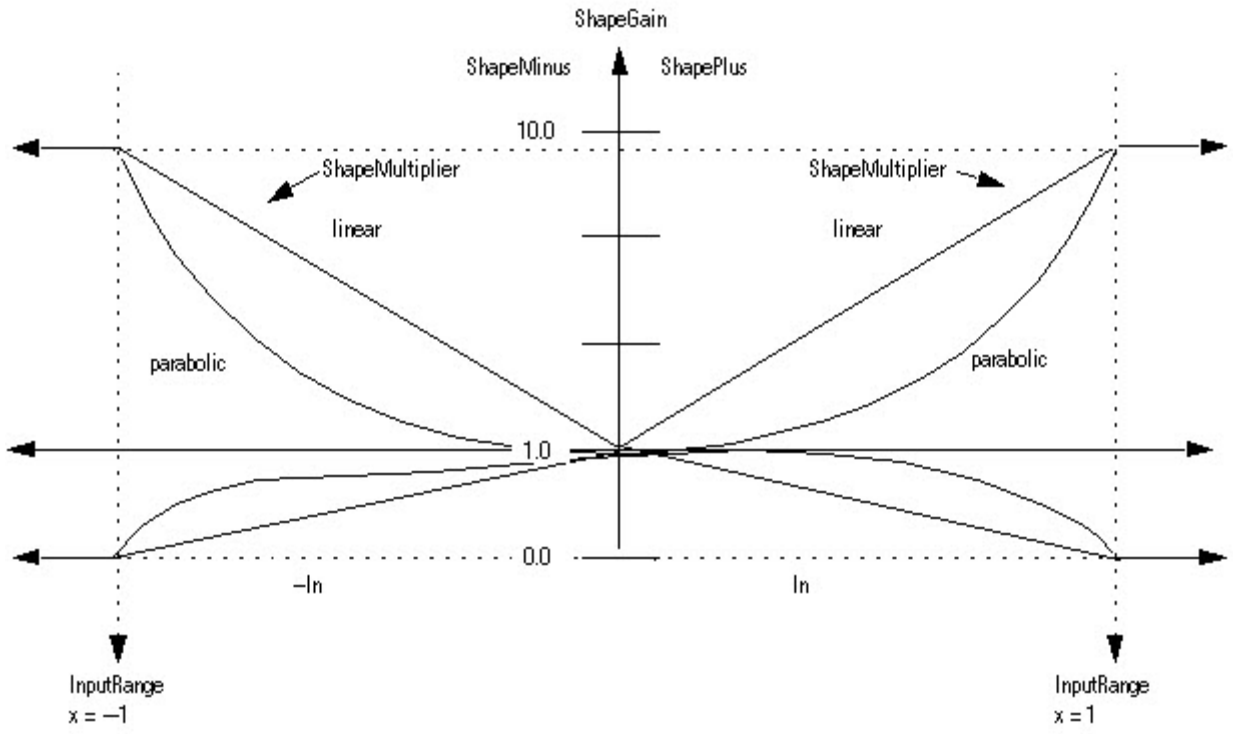
The gains specified by Kp and Wld are multiplied by 1.0 when In = 0. Separate proportional and integral algorithms increase or decrease the proportional or integral gain as the magnitude of error changes. These algorithms use the input range and shaping gain parameters to compute the actual proportional and integral gains. Input range defines the range of In (i.e. error) over which the gain is shaped. Input ranges are set by the two KpInRange and WldInRange. Shaping gain defines the gain multiplier for the quadrant controlled by the shaping gain parameter. Shaping gains are set by ShapeKpPlus, ShapeKpMinus, ShapeWldPlus and ShapeWldMinus.

The ParabolicLinear input selects the non-linear gain mode. If ParabolicLinear is cleared, linear mode is selected. If ParabolicLinear is set, parabolic mode is selected.

To configure a particular shaping gain curve, enter a shaping gain 0.0-10.0 for integral shaping, a shaping gain 0.1-10.0 for proportional shaping, and the input range over which shaping is to be applied. Kp and Wld are multiplied by the calculated ShapeMultiplier to obtain the actual proportional and integral gains. Entering a shaping gain of 1.0 disables the non-linear algorithm that calculates the proportional or integral gain for the quadrant.

When the magnitude of In (error) is greater then InRange then the ShapeMultiplier equals the value computed when | In | was equal to InRange.

The following diagram illustrates the maximum and minimum gain curves that represent the parabolic and linear gain equations.



The instruction calculates the value for Out using these equations:

Value	Equation
Kp shaping gain multiplier	If In ≥ 0 then: $KpShapeGain = ShapeKpPlus$ $KpRange = KpInRange$ Else: $KpShapeGain = ShapeKpMinus$ $KpRange = -KpInRange$
Kp input ratio	If |In| ≤ KpInRange: $KpInputRatio = \|In\| \times \frac{1}{KpInRange}$ Else: $KpInputRatio = 1$
Kp ratio	If not parabolic mode: $KpRatio = KpInputRatio \times 0.5$ If parabolic mode: $KpRatio = KpInputRatio^2 \times 0.333$
Kps shaping gain	$Kps = Kp \times (((KpShapeGain - 1) \times KpRatio) + 1)$
Proportional output	If |In| ≤ KpInRange: $PTerm = Kps \times In$ Else, limit gain: $PTerm = Kps \times KpRange + (In - KpRange) \times KpShape$
Wld shaping gain	If In ≥ 0 then: $WldShapeGain = ShapeWldPlus$ Else: $WldShapeGain = ShapeWldMinus$
Wld input	If In > WldRange then: $WldInput = WldInRange$ Else if In < –WldInRange then: $WldInput = -WldInRange$ Else: $WldInput = In$
Wld input ratio	If |In| ≤ WldInRange: $WldInputRange = \|In\| \times \frac{1}{WldInRange}$ Else: $WldInputRange = 1$
Wld ratio	If not parabolic mode: $WldRatio = WldInputRatio$ If parabolic mode: $WldRatio = WldInputRatio^2$
Wlds shaping gain	$Wlds = Wld \times (((WldShapeGain - 1) \times WldRatio) + 1)$
Wlds limits	$LowLimit > 0$ $HighLimit = \frac{0.7\pi}{DeltaT}$
Integral output	$ITerm = Kps \times Wlds \times \frac{(WldInput + WldInput_{n-1})}{2} \times DeltaT + ITerm_{n-1}$

Value	Equation
Output	$Out = PTerm + ITerm$

Limiting

The instruction stops the ITerm windup based on the state of the hold inputs.

Condition	Action
If HoldHigh is set **and** ITerm > $ITerm_{n-1}$	ITerm = $ITerm_{n-1}$
If HoldLow is set **and** ITerm < $ITerm_{n-1}$	ITerm = $ITerm_{n-1}$

The instruction also stops integrator windup based on the HighLimit and LowLimit values.

Condition	Action
Integrator > HighLimit	Integrator = HighLimit
Integrator > LowLimit	Integrator = LowLimit

The instructions limits the value of Out based on the HighLimit and LowLimit values.

Condition	Action
HighLimit ≤ LowLimit	Out = LowLimit ITerm = LowLimit HighLowLimsInv is set HighAlarm is set LowAlarm is set WldInput = 0
Out ≥ HighLimit	Out = HighLimit ITerm = $ITerm_{n-1}$ HighAlarm is set
ITerm > HighLimit	ITerm = HighLimit
Out ≤ LowLimit	Out = LowLimit ITerm = $ITerm_{n-1}$ LowAlarm is set
ITerm < LowLimit	ITerm = LowLimit

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

Condition	Action Taken
Prescan	EnableIn and EnableOut are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bit is set to true. The instruction executes.
Instruction first run	Out n-1 = 0 The algorithm used to calculate Out will not be executed.
Instruction first scan	Out n-1 = 0 The algorithm used to calculate Out will not be executed.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

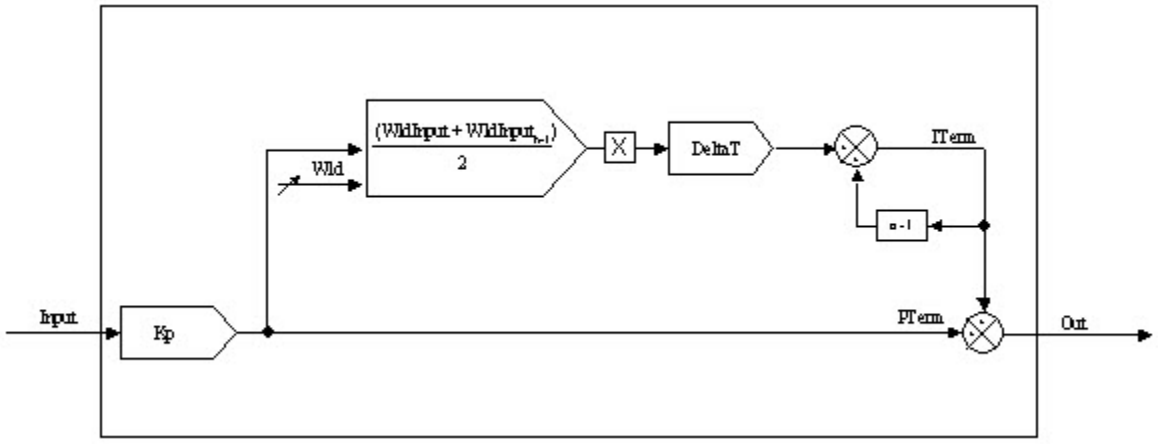
Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

The PI instruction is a regulating instruction with proportional and integral gain components. The integral gain component is set by the user in radians/sec; this sets the basic frequency response of the PI regulator. The proportional gain sets the overall gain of the block, including the proportional AND integral gain of the block.

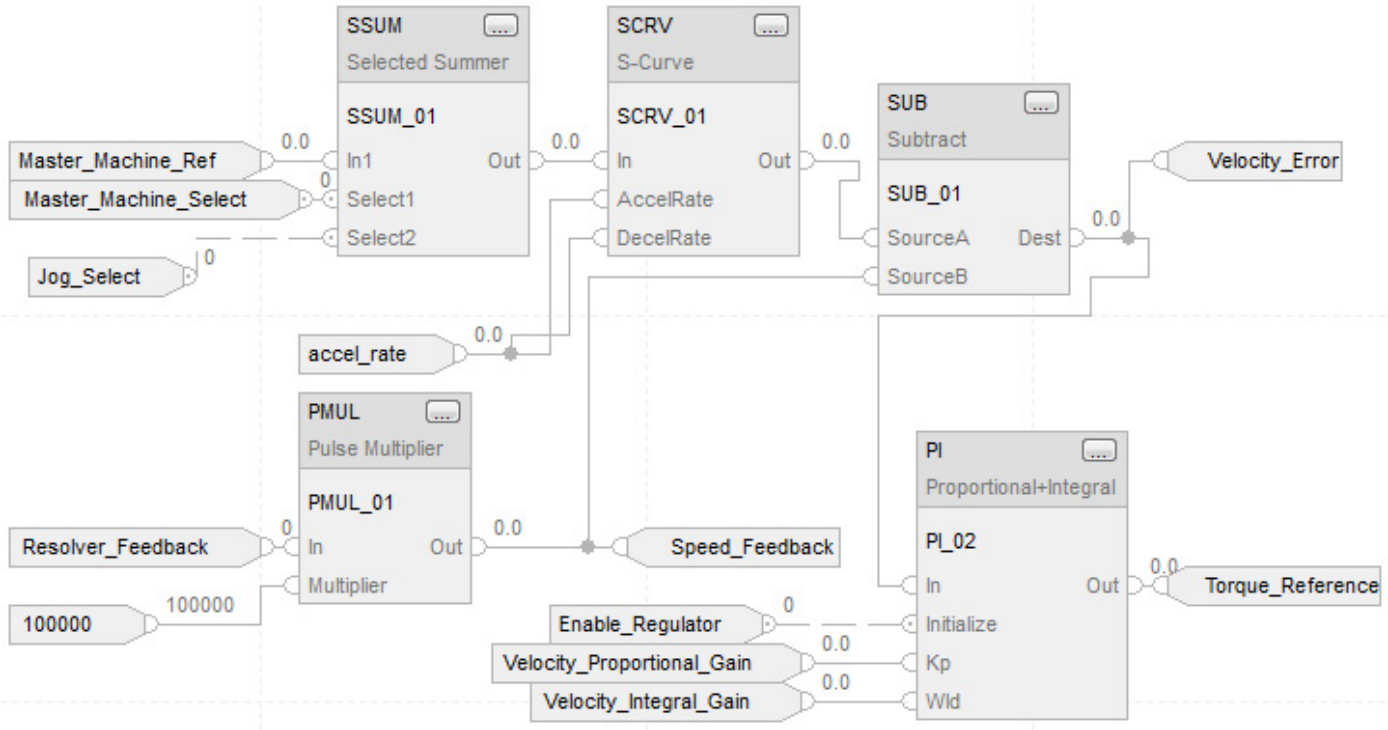
Excluding initialization and holding/clamping functionality, the following diagram shows the PI block's basic regulating loop while in the linear mode.

PI Instruction: Linear Mode



The following example shows the PI instruction used as a velocity regulator. In this example, velocity error is created by subtracting the velocity feedback signal (see the PMUL instruction example) from the system's velocity reference (through the SCRV instruction). Velocity error is driven directly into the PI instruction, which acts on this signal according to the function shown in the diagram above.

Function Block



Structured Text

```
Reference_Select.In1 := Master_Machine_Ref;
Reference_Select.Select1 := Master_Machine_Select;
Reference_Select.In2 := Section_Jog;
Reference_Select.Select2 := Jog_Select;
SSUM(Reference_Select);

S_Curve.In := Reference_Select.Out;
S_Curve.AccelRate := accel_rate;
S_Curve.DecelRate := accel_rate;
SCRV(S_Curve);

PMUL_01.In := Resolver_Feedback;
PMUL_01.WordSize := 12;
PMUL_01.Multiplier := 100000;
PMUL(PMUL_01);

Speed_Feedback := PMUL_01.Out;
Velocity_Error := S_Curve.Out - Speed_Feedback;

PI_01.In := Velocity_Error;
PI_01.Initialize := Enable_Regulator;
PI_01.Kp := Velocity_Proportional_Gain;
PI_01.Wld := Velocity_Integral_Gain;
PI(PI_01);

Torque_Reference := PI_01.Out;
```

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Pulse Multiplier (PMUL)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

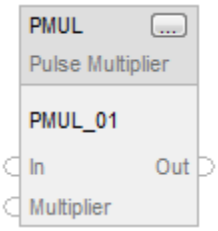
The PMUL instruction provides an interface from a position input module, such as a resolver or encoder feedback module, to the digital system by computing the change in input from one scan to the next. By selecting a specific word size, you configure the PMUL instruction to differentiate through the rollover boundary in a continuous and linear fashion.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram.

Function Block



Structured Text

PMUL(PMUL_tag);

Operands

Function Block

Operand	Type	Format	Description
PMUL tag	PULSE_MULTIPLIER	Structure	PMUL structure

Structured Text

Operand	Type	Format	Description
PMUL tag	PULSE_MULTIPLIER	Structure	PMUL structure

See Structured Text Syntax for more information on the syntax of expressions within structured text.

PULSE_MULTIPLIER Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
In	DINT	The analog input signal to the instruction. Valid = any DINT Default = 0

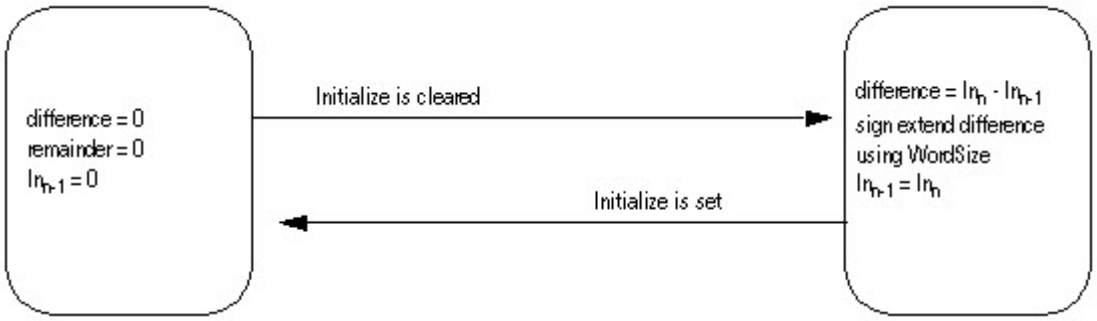
Initialize	BOOL	The initialize input. When set, Out is held at 0.0 and all the internal registers are set to 0. On a set to cleared transition of initialize, In_{n-1} = InitialValue (not valid for Absolute mode). When cleared, the instruction executes normally.
InitialValue	DINT	The initial value input. On a set to cleared transition of initialize, In_{n-1} = InitialValue Valid= any DINT Default = 0.
Mode	BOOL	The mode input. Set to enable Relative mode. Clear to enable Absolute mode. Default is set.
WordSize	DINT	The word size in bits. Specify the number of bits to use when computing (In_n-In_{n-1}) in Relative mode. WordSize is not used in Absolute mode. When the change in In is greater than 1/2 * $2^{(Wordsize-1)}$, Out changes sign. When WordSize is invalid,Out is held and the instruction sets the appropriate bit in Status. Valid = 2 to 32 Default = 14
Multiplier	DINT	The multiplier. Divide this value by 100,000 to control the ratio of In to Out. If invalid, the instruction limits the value and sets the appropriate bit in Status. Valid = -1,000,000 to 1,000,000 Default = 100,000

Output Parameter	Data Type	Description
EnableOut	BOOL	Enable output.
Out	REAL	The instruction's Out. If the Out calculation overflows, Out is forced to +/- and the appropriate bit in Status is set. Math status flags are set for this output.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
WordSizeInv (Status.1)	BOOL	Invalid WordSize value.
OutOverflow (Status.2)	BOOL	The internal output calculation overflowed.
LostPrecision (Status.3)	BOOL	Out < -2^{24} or Out > 2^{24}. When the instruction converts Out from an integer to a real value, data is lost if the result is greater than $\|2^{24}\|$ because the REAL data type is limited to 2^{24}.
MultiplierInv (Status.4)	BOOL	Invalid Multiplier value.

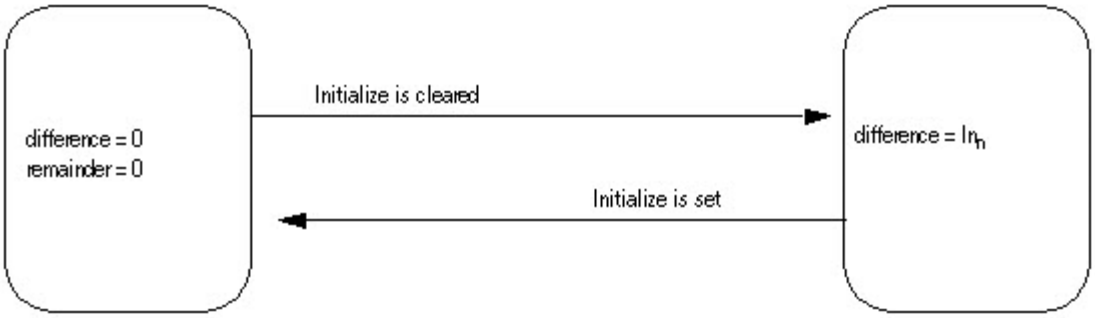
Description

The PMUL instruction operates in Relative or Absolute mode.

In Relative mode, the instruction's output is the differentiation of the input from scan to scan, multiplied by the (Multiplier/100,000). In Relative mode, the instruction saves any remainder after the divide operation in a scan and adds it back in during the next scan. In this manner, position information is not lost over the course of the operation.



In the Absolute mode, the instruction can scale an input, such as position, without losing any information from one scan to the next.



Calculating the Output and Remainder

The PMUL instruction uses these equations to calculate Out in either relative or absolute mode:

Ans = ((DiffInput x Multiplier) + INT_Remainder)

INT_Out = Ans / 100,000

INT_Remainder= Ans - (INT_Out * 100,000)

Out = INT_Out

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

Condition	Function Block Action
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	In_{n-1} = In. Out_{n-1} =0. Reminder = 0.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

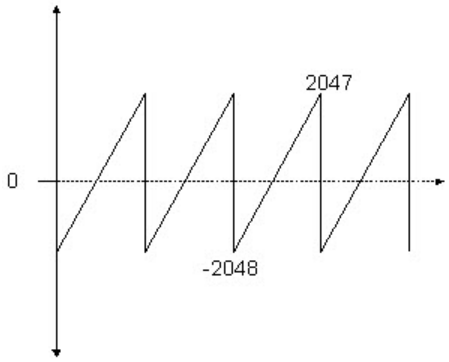
Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Examples

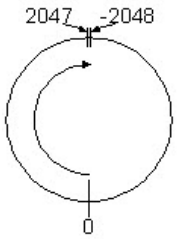
Example 1

The most common use of the PMUL instruction is in the relative mode of operation. In this mode, the PMUL instruction serves several purposes. First, in the relative mode, the PMUL instruction differentiates the information that it receives at its input from scan to scan. As data is received, the instruction outputs the difference of the input from one scan to the next. This means that if In = 500 at scan "n", and then In = 600 at scan "n+1", Out = 100 at scan "n+1."

Secondly, while in this mode of operation, the PMUL instruction also compensates for "rollover" values of binary data originating from a feedback module. For example, a resolver feedback module may have 12 bits of resolution, represented as a binary value, with sign, ranging from -2048 to 2047. In terms of raw data coming from the feedback module, the rotation of the feedback device might be represented as shown below:



In this example, as the value of the feedback data moves from 2047 to -2048, the effective change in position is equivalent to a jump of 4095 counts in position. In reality, however, this change in position is only 1 part in 4096 in terms of the rotation of the resolver feedback device. By understanding the true word size of the data that is being input from the feedback module, the PMUL instruction views the data in a rotary fashion as shown in the following diagram:



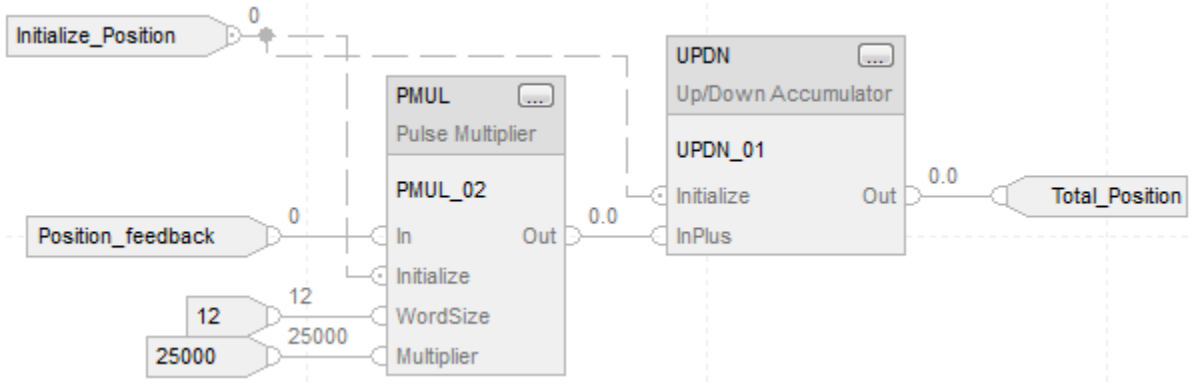
By knowing the word size of the data that is input to this block, the PMUL instruction differentiates an output of 1 count as the input to the block moves from 2047 to -2048, instead of the mathematically calculated 4095.

Tip: When applying this block, it is important to note that the feedback data should not change by more than one-half of the word size from one scan to the next, if rotational direction is to be properly differentiated.

In the example above, if the feedback device is moving in a clockwise direction such that at scan 'A' it reads 0 and then scan 'B' it reads -2000, actual change in position is equivalent to +2096 counts in the clockwise direction. However, since these two values are more than one half the words size, (or more than one half the rotation of the physical device,) the PMUL instruction calculates that the feedback device rotated in the opposite direction and returns a value of -2000 instead of +2096.

The third attribute of the pulse multiplier block is that it retains the fractional components from one scan to the next of any remainders that exist as a result of the Multiplier/100,000 scaling factor. As each execution of the block is completed, the remainder from the previous scan is added back into the total of the current value so that all counts or "pulses" are ultimately accounted for and no data is lost in the system. The output of the block, Out always yields a whole number in a floating point data type.

Function Block



Assuming Initial_Position = 0 and Multiplier = 2500 => (25,000/100,000)

Scan	Position_Feedback	PMUL_02.Out	Total_Position
n	0	0	0
n + 1	1	0	0
n + 2	2	0	0
n + 3	3	0	0
n + 4	4	1	1
n + 5	5	0	1

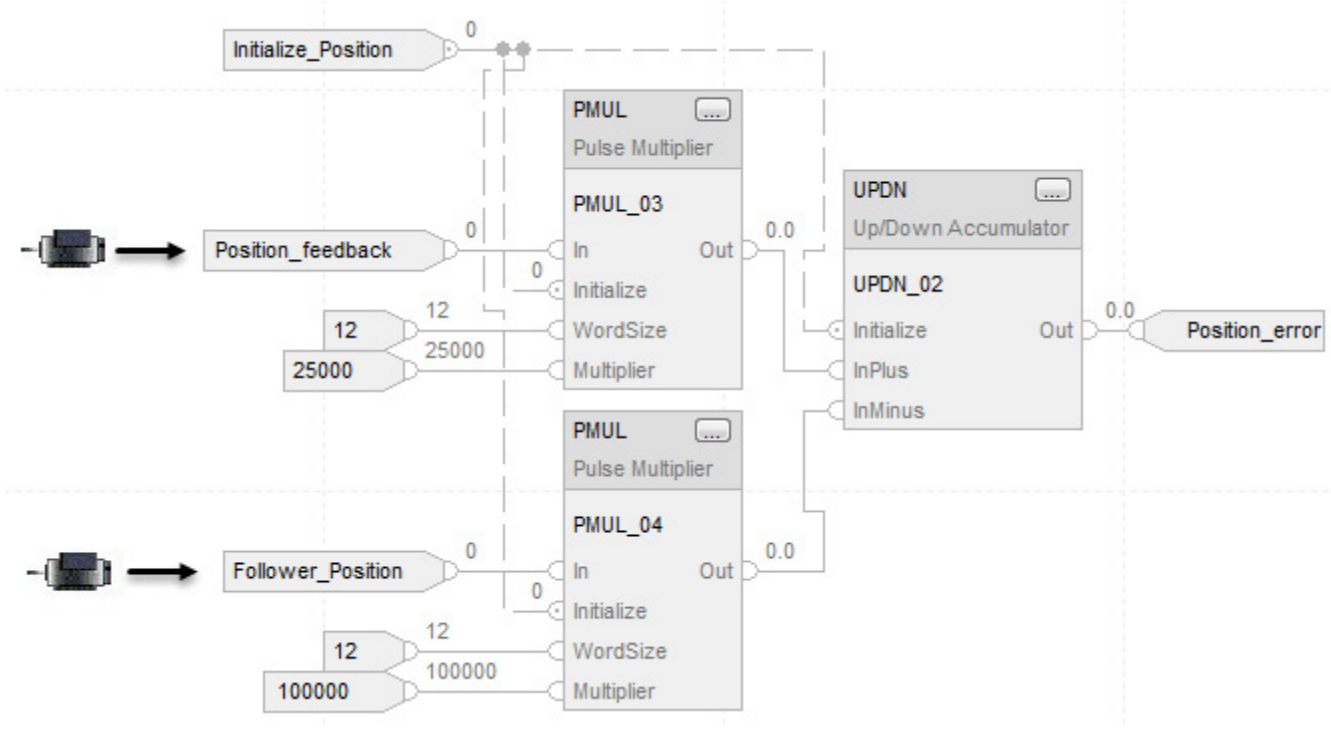
Structured Text

```
MUL_02.In := Position_feedback;
PMUL_02.Initalize := Initialize_Position;
PMUL_02.WordSize := 12;
PMUL_02.Multiplier := 25000;
PMUL(PMUL_02);

UPDN_02.Initialize := Initialize_Position;
UPDN_02.InPlus := PMUL_02.Out;
UPDN(UPDN_02);

Total_Position := UPDN_02.Out;
```

Example 2

In this electronic line shaft application, motor A's feedback acts as a master reference which motor B needs to follow. Motor A's feedback is aliased to "Position_feedback." Motor B's feedback is aliased to "Follower_Position." Due to the multipliers of both instructions being a ratio of 1/4, motor B needs to rotate once for every four revolutions of Motor A in order to maintain an accumulated value of zero in the UPDN accumulator. Any value other than zero on the output of the UPDN instruction is viewed as Position_error and can be regulated and driven back out to motor B in order to maintain a phase-lock between the two motors.

Function Block



Structured Text

```
PMUL_02.In := Position_feedback;
PMUL_02.Initalize := Initialize_Position;
PMUL_02.WordSize := 12;
PMUL_02.Multiplier := 25000;
PMUL(PMUL_02);

PMUL_03.In := Follower_Position;
PMUL_03.Initalize := Initialize_Position;
PMUL_03.WordSize := 12;
PMUL_03.Multiplier := 100000;
PMUL(PMUL_03);

UPDN_02.Initialize := Initialize_Position;
UPDN_02.InPlus := PMUL_02.Out;
UPDN_02.InMinus := PMUL_03.Out;
UPDN(UPDN_02);

Position_error := UPDN_02.Out;
```

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

S-Curve (SCRV)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

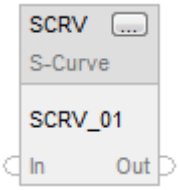
The SCRV instruction performs a ramp function with an added jerk rate. The jerk rate is the maximum rate of change of the rate used to ramp output to input.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram.

Function Block



Structured Text

SCRV(SCRV_tag);

Operands

Function Block

Operand	Type	Format	Description
SCRV tag	S_CURVE	Structure	SCRV structure

Structured Text

Operand	Type	Format	Description
SCRV tag	S_CURVE	Structure	SCRV structure

See Structured Text Syntax for more information of the syntax of expressions within structured text.

S_CURVE Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0
Initialize	BOOL	The Initialize input to the instruction. When set, the instruction holds Out = InitialValue Default is cleared.
InitialValue	REAL	Initial value of S-Curve. When Initialize is set, Out = InitialValue. Valid = any float Default = 0.0
AbsAlgRamp	BOOL	Ramp type. If set, the instruction functions as an absolute value ramp. If cleared, the instruction functions as an algebraic ramp. Default is set
AccelRate	REAL	Acceleration rate in input units per second2. A value of zero prevents Out form accelerating. When AccelRate < 0, the instruction assumes AccelRate = 0 and sets the appropriate bit in Status. Valid = 0.0 to maximum positive float Default = 0.0
DecelRate	REAL	Deceleration rate in input units per second2. A value of zero prevents Out form decelerating. When the DecelRate < 0, the instruction assumes DecelRate = 0 and sets the appropriate bit in Status. Valid = 0.0 to maximum positive float Default = 0.0
JerkRate	REAL	Deceleration rate in input units per second2. A value of zero prevents Out form decelerating. When the DecelRate < 0, the instruction assumes DecelRate = 0 and sets the appropriate bit in Status. Valid = 0.0 to maximum positive float Default = 0.0
HoldMode	BOOL	S-Curve hold mode parameter. This parameter is used with the HoldEnable parameter. If HoldMode is set when HoldEnable is set and Rate = 0, the instruction holds Out constant. In this situation, the instruction holds Out as soon as HoldEnable is set, the JerkRate is ignored, and Out produces a "corner" in its profile. If HoldMode is cleared when HoldEnable is set, the instruction uses the JerkRate to bring Out to a constant value. Out is held when Rate = 0. Do not change HoldMode once HoldEnable is set because the instruction will ignore the change. Default is cleared.
HoldEnable	BOOL	S-Curve hold enable parameter. When set, Out is held. When cleared, Out moves from its current value until it equals In. Default is cleared.
TimingMode	DINT	Selects timing execution mode. 0 = periodic mode 1 = oversample mode 2 = real time sampling mode For more information about timing modes, see Function Block Attributes. Valid = 0 to 2 Default = 0
OversampleDT	REAL	Execution time for oversample mode. Valid = 0 to 4194.303 seconds Default = 0
RTSTime	DINT	Module update period for real time sampling mode Valid = 1 to 32,767ms Default = 1

Input Parameter	Data Type	Description
RTSTimeStamp	DINT	Module time stamp value for real time sampling mode. Valid = 0 to 32,767ms Default = 0

Output Parameter	Data Type	Description
EnableOut	BOOL	Enable output.
S_Mode	BOOL	S_Mode Output. When (Jerk * DeltaT) ≤ Rate and Rate < Accel or Decel, S_Mode is set. Otherwise, S_Mode is cleared.
Out	REAL	The output of the S-Curve instruction. Math status flags are set for this output.
Rate	REAL	Internal change in the Out in units per second.
DeltaT	REAL	Elapsed time between updates. This is the elapsed time in seconds used by the control algorithm to calculate the process output.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
AcceRateInv (Status.1)	BOOL	AccelRate is negative.
DecelRateInv (Status.2)	BOOL	DecelRate is negative.
JerkRateInv (Status.3)	BOOL	JerkRate is negative.
TimingModeInv (Status.27)	BOOL	Invalid timing mode. For more information about timing modes, see Function Block Attributes.
RTSMissed (Status.28)	BOOL	Only used in real time sampling mode. Set when ABS | DeltaT - RTSTime | > 1 (.001 second).
RTSTimeInv (Status.29)	BOOL	Invalid RTSTime value.
RTSTimeStampInv (Status.30)	BOOL	Invalid RTSTimeStamp value.
DeltaT (Status.31)	BOOL	Invalid DeltaT value.

Description

The primary requirement of the SCRV instruction is to ensure that the rate never changes by more than the specified jerk rate.

You can configure the SCRV instruction to produce an S-Curve profile or a Ramp profile for a step input.

S-Curve Profile

To produce an S-Curve profile, set JerkRate such that (JerkRate * DeltaT) < AccelRate and/or DecelRate.

In S-Curve profile mode, the SCRV instruction ensures that the rate never changes more than the specified JerkRate. The algorithm used to produce the S-Curve profile is designed to produce a smooth, symmetric S-Curve for a step input. A trapezoidal integration of Out is incorporated to facilitate this. As a result, changes in Rate will be less than JerkRate during portions of the profile.

When a step change occurs on the input, rate is increased to the programmed AccelRate or DecelRate. The AccelRate or DecelRate is maintained until a point at which rate must begin decreasing in order for the output to reach input when rate reaches zero.

In some cases, depending on the values of acceleration, deceleration, and jerk, the acceleration rate or deceleration rate might not be reached before the rate must begin decreasing by jerk rate.

For very small step changes, the SCRV instruction will not attempt to produce an 'S' profile. In this mode the entire step will be output and Rate will reflect the change in output. This behavior will occur if Out = In and the next step change to In can be output with a rate less than or equal to the programmed JerkRate.

The SCRV instruction supports an algebraic ramp and an absolute value ramp. For an algebraic ramp, the acceleration condition is defined by an input that is becoming more positive, and the deceleration condition is defined by an input that is becoming more negative. For an absolute value ramp, the acceleration condition is defined by an input moving away from zero, and the deceleration condition is defined by an input moving towards zero.

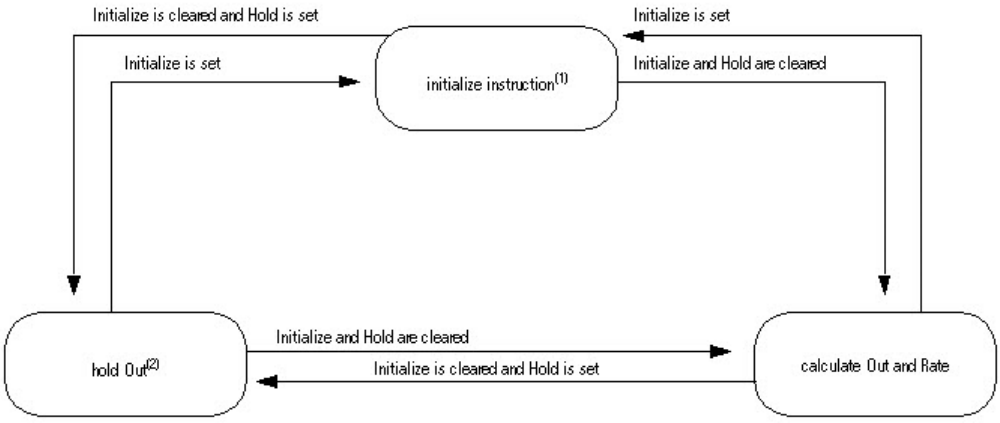
Ramp Profile

To produce a Ramp profile, set JerkRate such that (JerkRate * DeltaT) ≤ AccelRate and/or DecelRate.

In Ramp Profile mode, the SCRV instruction always produces a rate of change equal to the programmed AccelRate or DecelRate until the difference between Out and In requires less then AccelRate or DecelRate to reach endpoint.

HoldMode = 0 operates the same as HoldMode = 1. When HoldEnable is set, Out is immediately held and Rate becomes zero.

The following diagram illustrates how the instruction modifies Out.



(1) When Initialize is set, the instruction sets the following:

Outn = InitialValue

Outn-1 = Outn

Raten = 0

Raten-1 = 0

(2) When HoldMode is cleared, Out is moving toward In, and HoldEnable is set, the rate begins decreasing towards zero at the jerk rate. Due to the JerkRate, Out is held at whatever value it had when the rate reached zero. When the Out is finally held constant, it has a value that is different from the value it had the instant that HoldEnable was set.

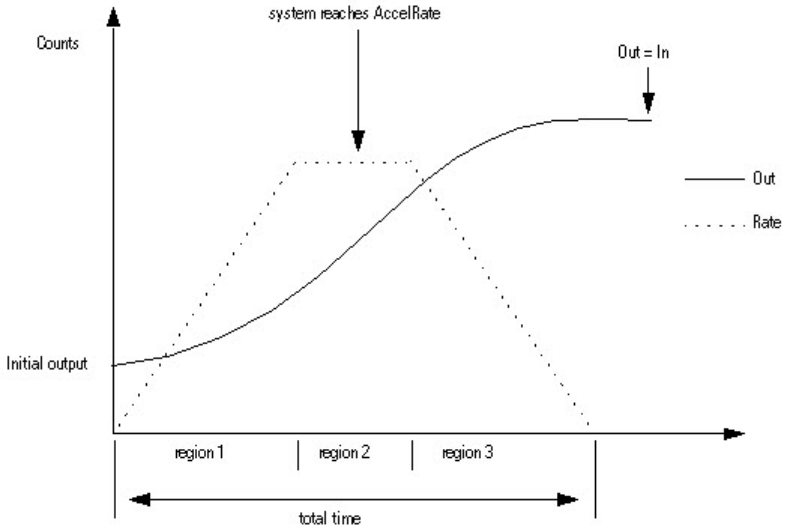
When HoldMode is set, Out is moving toward In, and HoldEnable is set, the rate is immediately set to zero. Out is held at whatever value it had when HoldEnable was set.

Reducing the JerkRate during a transition might cause Out to overshoot the In. If overshoot occurs, it is the result of enforcing the entered JerkRate. You can avoid an overshoot by decreasing JerkRate in small steps while tuning or by changing JerkRate while Out = In (not during a transition).

The time that is required for Out to equal a change in the input is a function of AccelRate, JerkRate, and the difference between In and Out.

Calculating Output and Rate Values

In transition from an initial value to final value, Out goes through three regions. In region 1 and region 3, the rate of change of Out is based on JerkRate. In region 2, the rate of change of Out is based on AccelRate or DecelRate.



The Out is calculated for each region as follows:

$$TotalTime = \frac{FinalOutput - InitialOutput}{AccelRate} + \frac{AccelRate}{JerkRate}$$

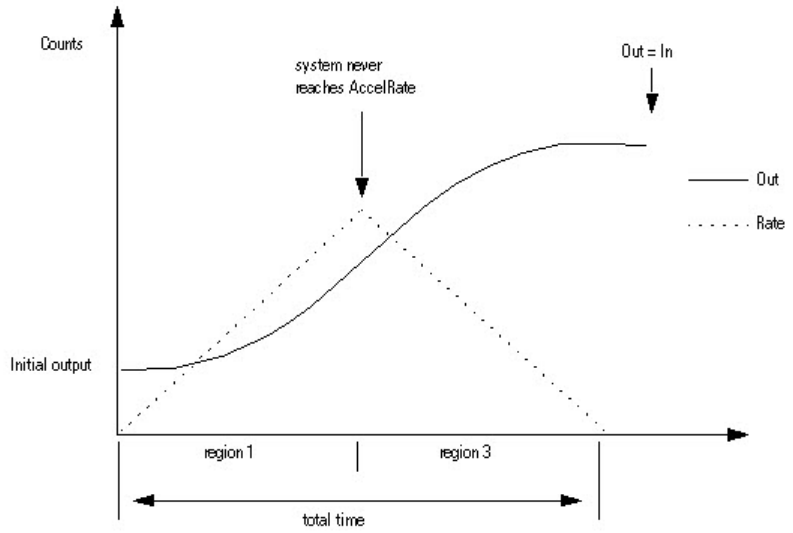
with these equations for each region:

Region	Equation
region 1	$Time_1 = \frac{AccelRate}{JerkRate}$ $Y(Time) = InitialOutput + \frac{1}{2}(JerkRate) \times Time^2$
region 2	$Time_2 = \frac{JerkRate \times (FinalOutput - InitialOutput) - AccelRate^2}{JerkRate \times AccelRate}$ $Y(Time) = InitialOutput + (AccelRate \times Time) - \frac{AccelRate^2}{2 \times JerkRate}$
region 3	$Time_3 = \frac{AccelRate}{JerkRate}$ $Y(Time) = FinalOutput - \frac{1}{2}(JerkRate) \times \left(Time - \frac{FinalOutput - Initia Output}{AccelRate} - \frac{AccelRate}{JerkRate}\right)^2$

When:

$$|InitialOutput - FinalOutput| < \frac{AccelRate^2}{JerkRate}$$

the SCRV block does not reach the AccelRate or DecelRate. The Out does the following:



where:

$$TotalTime = 2 \times \sqrt{\frac{|InitialOutput - FinalOutput|}{JerkRate}}$$

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

Condition	Function Block Action
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bit bits are set to true The instruction executes.
Instruction first run	N/A
Instruction first scan	Clear previous scan data.
Postscan	EnableIn and EnableOut bits are cleared to false

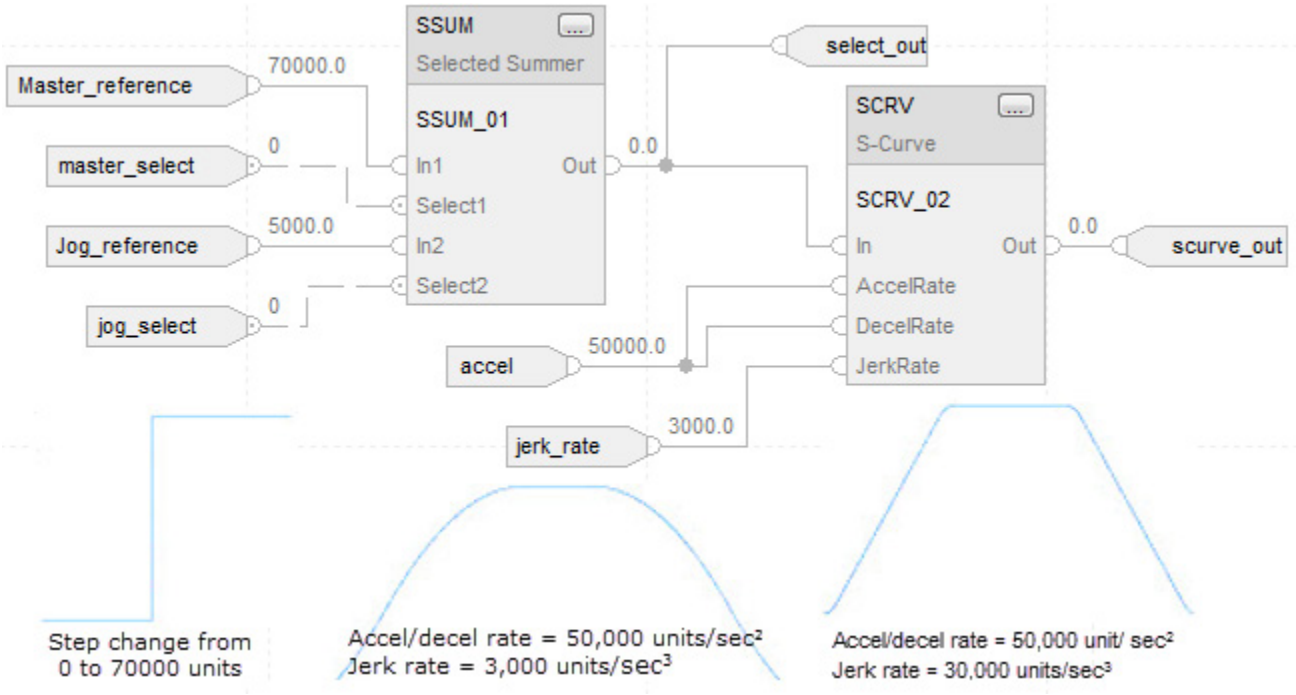
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

In most coordinated drive applications, a master reference commands line speed for an entire group of drives. As various references are selected, the drives cannot be presented with "step" changes in speed reference because differences in load inertia, motor torque, and tuning would not allow the individual drive sections to react in a coordinated manner. The SCRV instruction is designed to ramp and shape the reference signal to the drive sections so that acceleration, deceleration, and jerk, (derivative of acceleration,) are controlled. This instruction provides a mechanism to allow the reference to the drives to reach the designated reference setpoint in a manner that eliminates excessive forces and excessive impact on connected machinery and equipment.

Function Block



Structured Text

```
SSUM_01.In1 := Master_reference;
SSUM_01.Select1 := master_select;
SSUM_01.In2 := Jog_reference;
SSUM_01.Select2 := jog_select;
SSUM(SSUM_01);

select_out := SSUM_01.Out;

SCRV_01.In := select_out;
SCRV_01.AccelRate := accel;
SCRV_01.DecelRate := accel;
SCRV_01.JerkRate := jerk_rate;
SCRV(SCRV_01);

scurve_out := SCRV_01.Out
```

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Second-Order Controller (SOC)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

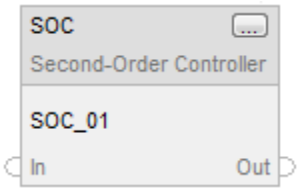
The SOC instruction is designed for use in closed loop control systems in a similar manner to the PI instruction. The SOC instruction provides a gain term, a first order lag, and a second order lead.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram.

Function Block



Structured Text

SOC(SOC_tag);

Operands

Function Block

Operand	Type	Format	Description
SOC tag	SEC_ORDER_CONTROLLER	Structure	SOC structure

SEC_ORDER_CONTROLLER Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0
Initialize	BOOL	The instruction initialization command. When set, Out and internal integrator are set equal to the value of InitialValue. Default is cleared.
InitialValue	REAL	The initial value input. When Initialize is set, Out and integrator are set to the value of InitialValue. The value of InitialValue is limited using HighLimit and LowLimit. Valid = any float Default = 0.0
Gain	REAL	The proportional gain for the instruction. If the value is out of range, the instruction limits the value and sets the appropriate bit in Status. Valid = any float > 0.0 Default = minimum positive float
WLag	REAL	First order lag corner frequency in radians/second. If the value is out of range, the instruction limits the value and sets the appropriate bit in Status. Valid = see the Description section below for valid ranges Default = 0.0

Input Parameter	Data Type	Description
WLead	REAL	Second order lead corner frequency in radians/second. If the value is out of range, the instruction limits the value and sets the appropriate bit in Status. Valid = see the Description section below for valid ranges Default = 0.0
ZetaLead	REAL	Second order lead damping factor. If the value is out of range, the instruction limits the value and sets the appropriate bit in Status. Valid = 0.0 to 10.0 Default = 0.0
HighLimit	REAL	The high limit value. This is the maximum value for Out. If HighLimit ≤ LowLimit, the instruction sets HighAlarm and LowAlarm, sets the appropriate bit in Status, and sets Out = LowLimit. Valid = LowLimit < HighLimit ≤ maximum positive float Default = maximum positive float
LowLimit	REAL	The low limit value. This is the minimum value for Out. If HighLimit ≤ LowLimit, the instruction sets HighAlarm and LowAlarm, sets the appropriate bit in Status, and sets Out = LowLimit. Valid = maximum negative float ≤ LowLimit < HighLimit Default = maximum negative float
HoldHigh	BOOL	The hold high command. When set, the value of the internal integrator is not allowed to increase in value. Default is cleared.
HoldLow	BOOL	The hold low command. When set, the value of the internal integrator is not allowed to decrease in value. Default is cleared.
TimingMode	DINT	Selects timing execution mode. 0 = Periodic mode 1 = Oversample mode 2 = Real time sampling mode For more information about timing modes, see Function Block Attributes. Valid = 0 to 2 Default = 0
OversampleDT	REAL	Execution time for oversample mode. Valid = 0 to 4194.303 seconds Default = 0
RTSTime	DINT	Module update period for real time sampling mode Valid = 1 to 32,767ms Default = 1
RTSTimeStamp	DINT	Module time stamp value for real time sampling mode. Valid = 0 to 32,767ms Default = 0

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if Out overflows.

Output Parameter	Data Type	Description
Out	REAL	The calculated output of the algorithm.
HighAlarm	BOOL	The maximum limit alarm indicator. Set when the calculated value for Out ≥HighLimit and the output is clamped at HighLimit.
LowAlarm	BOOL	The minimum limit alarm indicator. Set when the calculated value for Out ≤ LowLimit and the output is clamped at LowLimit.
DeltaT	REAL	Elapsed time between updates. This is the elapsed time in seconds used by the control algorithm to calculate the process output.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
GainInv (Status.1)	BOOL	Gain < minimum positive float.
WLagInv (Status.2)	BOOL	WLag > maximum or WLag < minimum.
WLeadInv (Status.3)	BOOL	WLead > maximum or WLead < minimum.
ZetaLeadInv (Status.4)	BOOL	ZetaLead > maximum or ZetaLead < minimum.
HighLowLimsInv (Status.5)	BOOL	HighLimit ≤ LowLimit.
TimingModeInv (Status.27)	BOOL	Invalid timing mode. For more information about timing modes, see Function Block Attributes.
RTSMissed (Status.28)	BOOL	Only used in real time sampling mode. Set when ABS | DeltaT - RTSTime | > 1 (.001 second).
RTSTimeInv (Satus.29)	BOOL	Invalid RTSTime value.
RTSTimeStampInv (Status.30)	BOOL	Invalid RTSTimeStamp value.
DeltaT (Status.31)	BOOL	Invalid DeltaT value.

Structured Text

Operand	Type	Format	Description
SOC tag	SEC_ORDER_CONTROLLER	structure	SOC structure

See Structured Text Syntax for more information of the syntax of expressions within structured text.

Description

The SOC instruction provides a gain term, a first order lag, and a second order lead. The frequency of the lag is adjustable and the frequency and damping of the lead is adjustable. The zero pair for the second order lead can be complex (damping is less than unity) or real (damping ≥ to unity). The SOC

instruction is designed to execute in a task where the scan rate remains constant.

The SOC instruction uses the following Laplace Transfer equation.

$$H(s) = \frac{K\left(\frac{s^2}{\omega_{Lead}^2} + \frac{2 \times \xi_{Lead} \times s}{\omega_{Lead}} + 1\right)}{s\left(\frac{s}{\omega_{Lag}} + 1\right)}$$

Parameter Limitations

The following SOC parameters have these limits on valid values.

Parameter	Limit
WLead	$LowLimit = \frac{0.00001}{DeltaT}$ $HighLimit = \frac{0.07\pi}{DeltaT}$ where DeltaT is in seconds
WLag	$LowLimit = \frac{0.0000001}{DeltaT}$ $HighLimit = \frac{0.07\pi}{DeltaT}$ where DeltaT is in seconds
ZetaLead	LowLimit = 0.0 HighLimit = 10.0

Whenever the value computed for the output is invalid or NAN, the instruction sets Out = the invalid value. The internal parameters are not updated. In each subsequent scan, the output is computed using the internal parameters from the last scan when the output was valid.

Limiting

The instruction stops wind-up based on state of the Hold inputs.

If:	Then:
HoldHigh is set and Integrator > Integrator $_{n-1}$	Integrator = Integrator$_{n-1}$
HoldLow is set and Integrator < Integrator$_{n-1}$	Integrator = Integrator$_{n-1}$

The instruction also stops integrator windup based on the HighLimit and LowLimit values.

If:	Then:
Integrator > IntegratorHighLimit	Integrator = IntegratorHighLimit

Integrator < IntegratorLowLimit	Integrator = IntegratorLowLimit

where:

$$IntegratorHighLimit = HighLimit \times \frac{Gain \times WLag}{WLead^2}$$

$$IntegratorLowLimit = LowLimit \times \frac{Gain \times WLag}{WLead^2}$$

The instruction also limits the value of Out based on the HighLimit and LowLimit values.

If:	Then:
HighLimit ≤ LowLimit	Out = LowLimit Integrator = IntegratorLowLimit HighLowLimsInv is set HighAlarm is set LowAlarm is set
Out ≥ HighLimit	Out = HighLimit $IntegratorLowLimit_{n-1}$ HighAlarm is set
Out ≤ LowLimit	Out = LowLimit Integrator = $Integrator_{n-1}$ LowAlarm is set

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bit is set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	The internal parameters and Out are set to 0. Force recalculation of equation coefficients.

Postscan	EnableIn and EnableOut bits are cleared to false.

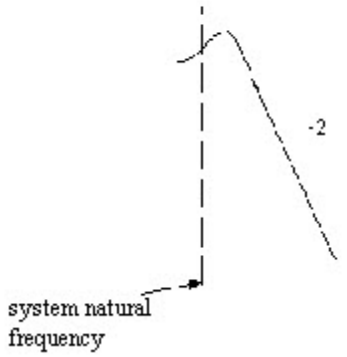
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

The SOC instruction is a specialized function block that is used in applications where energy is transferred between two sections through a spring-mass system. Typically in these types of applications, the frequency response of the process itself can be characterized as shown in the bode diagram A below:

Diagram A: Process characteristics

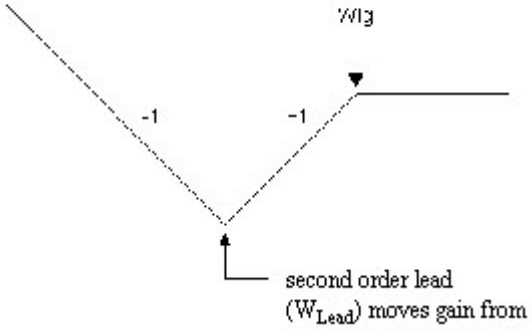


The SOC instruction implements a first order lag filter followed by a PID controller to implement a transfer function with an integration, a second order zero, (lead,) and a first order pole (lag.) With this instruction, PID tuning is simplified because the regulating terms are arranged so that you have WLead and ZLead as inputs to the SOC instruction, rather than Kp, Ki, and Kd values. The transfer function for the SOC instruction is:

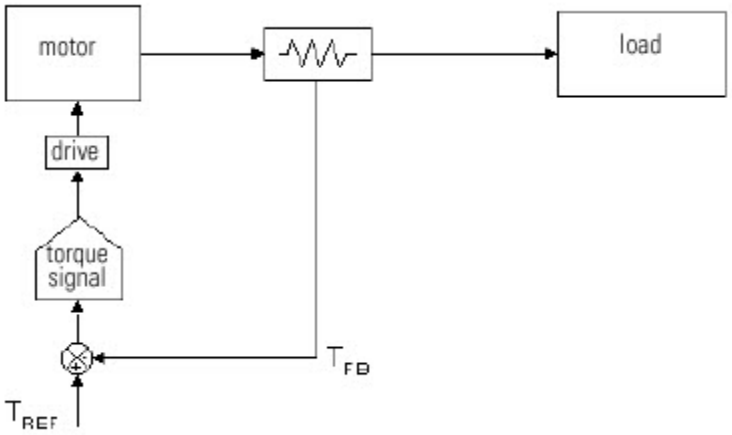
$$H(s) = \frac{K\left(\frac{s^2}{\omega_{Lead}^2} + \frac{2 \times \xi_{Lead} \times s}{\omega_{Lead}} + 1\right)}{s\left(\frac{s}{\omega_{Lag}} + 1\right)}$$

Its corresponding bode diagram is shown in Diagram B below.

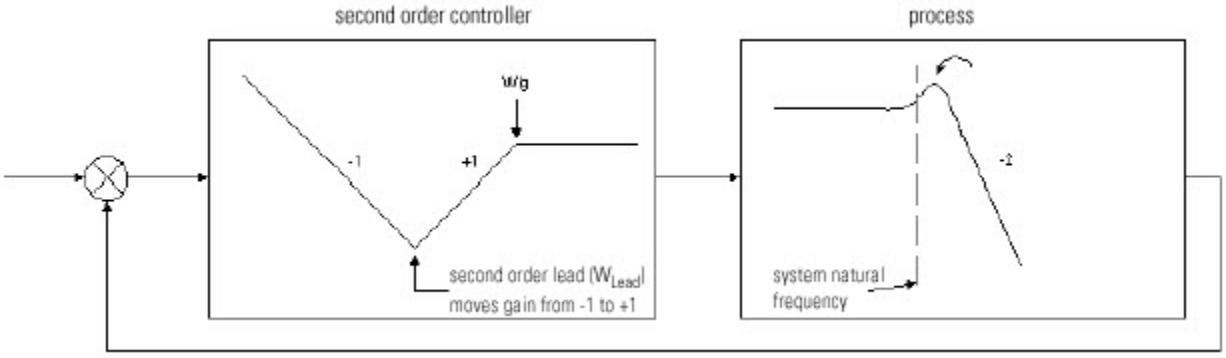
Diagram B: Second order controller



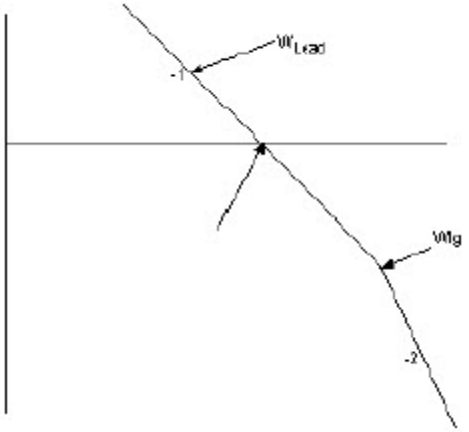
The SOC instruction can be used in a torque or tension regulating application where a load cell or force transducer is used as feedback and the output of the regulating scheme operates directly on the torque (current) minor loop of the drive. In many such applications, the controlled system may be mechanically under-damped and have a natural frequency which is difficult to stabilize as it becomes reflected through the feedback device itself.



Using the SOC instruction, PID tuning is simplified because the regulating terms can be arranged so that you have WLead and ZLead as inputs to the SOC instruction, rather than Kp, Ki, and Kd values. In this manner, the corner frequencies of the controller/regulator are easier to adjust and setup against the real world process. During startup, the natural frequency of the system and the damping factor can be measured empirically or on-site. Afterward, the parameters of the regulator can be adjusted to match the characteristics of the process, allowing more gain and more stable control of the final process.



In the system above, if Wlead is set equal to the system natural frequency, and if Wlag is set substantially above the desired crossover frequency, (> 5 times crossover), the resulting system response would look like the following:



In an actual application, the steps in using and setting up this instruction include:

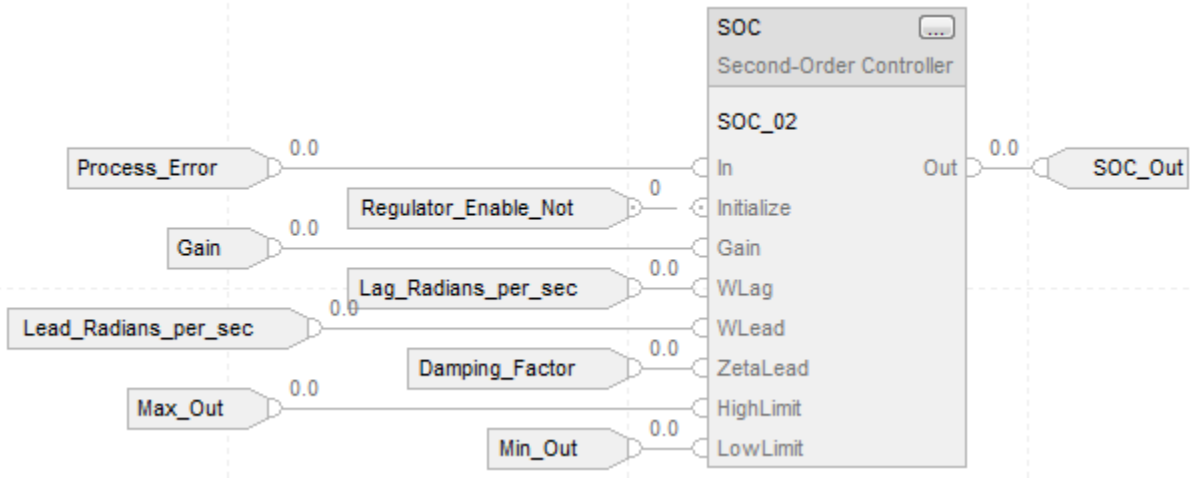
Recognize the type of process that is being controlled. If the system's response to a step function results in a high degree of ringing or can be characterized by the process curve shown above, this block may provide the regulating characteristics required for stable control.

Determine the natural frequency of the system/process. This can may be arrived at empirically - or it might be measured on-site. Adjust WLead so that it corresponds with, or is slightly ahead of, the natural frequency of the process itself.

Tune damping factor, Zlead, so that it cancels out any of the overshoot in the system.

Move WLag out far enough past the system crossover frequency (>5 times) and begin increasing overall Gain to achieve

Function Block



Structured Text

```
SOC_01.In := Process_Error;
SOC_01.Initialize := Regulator_Enable_Not;
SOC_01.Gain := Gain;
SOC_01.WLag := Lag_Radians_per_sec;
SOC_01.WLead := Lead_radians_per_sec;
SOC_01.ZetaLead := Damping_Factor;
SOC_01.HighLimit := Max_Out;
SOC_01.LowLimit := Min_Out;
SOC(SOC_01);

SOC_Out := SOC_01.Out;
```

See also

Function Block Attributes on page 1043

Common Attributes on page 1083

Structured Text Syntax on page 1057

Up/Down Accumulator (UPDN)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

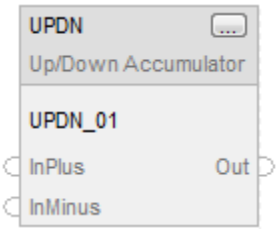
The UPDN instruction adds and subtracts two inputs into an accumulated value.

Available Languages

Ladder Diagram

This instruction is not available for ladder diagram diagram.

Function Block



Structured Text

UPDN(UPDN_tag)

Operands

Function Block

Operand	Type	Format	Description
UPDN tag	UP_DOWN_ACCUM	Structure	UPDN structure

UPDN Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
Initialize	BOOL	The initialize input request for the instruction. When Initialize is set, the instruction sets Out and the internal accumulator to InitialValue. Default is cleared.
InitialValue	REAL	The initialize value of the instruction. Valid = any float Default = 0.0

InPlus	REAL	The input added to the accumulator. Valid = any float Default = 0.0
InMinus	REAL	The input subtracted from the accumulator. Valid = any float Default = 0.0
Hold	BOOL	The hold input request for the instruction. When Hold is set and Initialize is cleared, Out is held. Default is cleared.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if Out overflows.
Out	REAL	The output of the instruction.

Structured Text

Operand	Type	Format	Description
UPDN tag	UP_DOWN_ACCUM	Structure	UPDN structure

See Structured Text Syntax for more information of the syntax of expressions within structured text.

Description

The UPDN instruction follows these algorithms.

Condition	Action
Hold is cleared and Initialize is cleared	$AccumValue_n = AccumValue_{n-1} + InPlus - InMinus$ $Out = AccumValue_n$
Hold is set and Initialize is cleared	$AccumValue_n = AccumValue_{n-1}$ $Out = AccumValue_n$
Initialize is set	$AccumValue_n$ = InitialValue $Out = AccumValue_n$

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

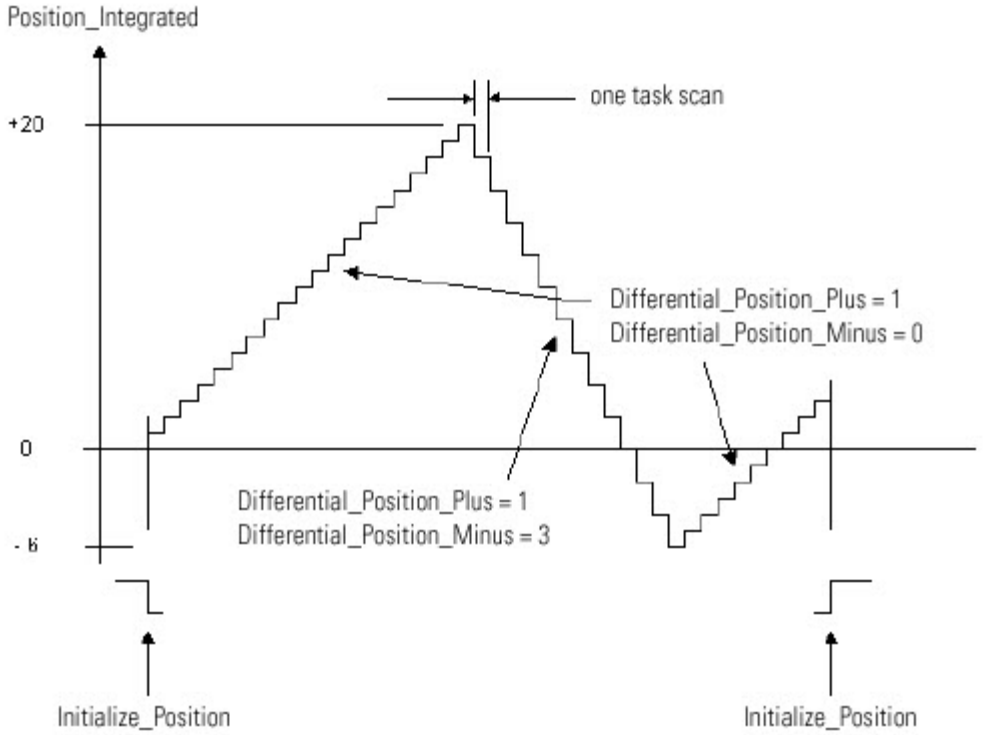
Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	Internal accumulator is set to zero.
Instruction first scan	N/A
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

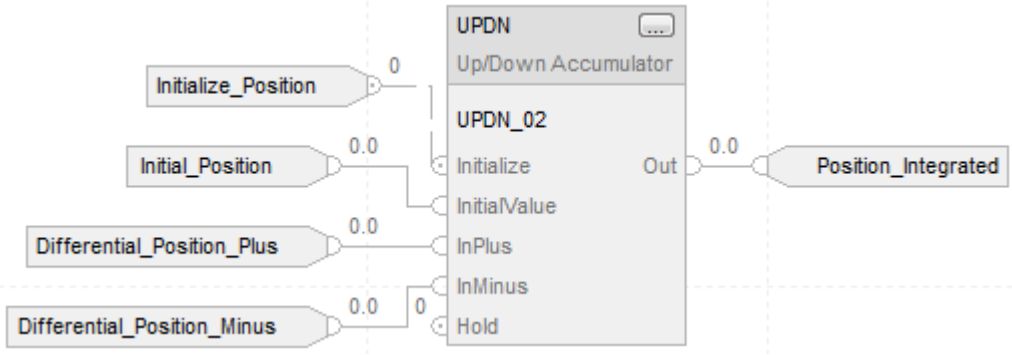
Example

The UPDN instruction integrates counts from one scan to the next. This instruction can be used for simple positioning applications or for other types of applications where simple integration is required to create an accumulated value from a process's differentiated feedback signal. In the example below, Initial_Position is set to zero, while Differential_Position_Plus and Differential_Position_Minus take varying values over a period of time. With this instruction, InPlus and InMinus could also accept negative values.



Function Block

The derivative instruction calculates the amount of change of a signal over time in per-second units. This instruction is often used in closed loop control to create a feed forward path in the regulator to compensate for processes that have a high degree of inertia.



Structured Text

```
UPDN_01.Initialize := Initialize_Position;
UPDN_01.InitialValue := Initial_Position;
UPDN_01.InPlus := Differential_Position_Plus;
UPDN_01.InMinus := Differential_Position_Minus;
UPDN(UPDN_01);

Position_Integrated := UPDN_01.Out;
```

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

HMI Button Control (HMIBC)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

Use the HMI Button Control (HMIBC) instruction with a PanelView 5500 Human Machine Interface (HMI) to enable operators to initiate machine control operations, such as jogging a motor or enabling a valve, with a high degree of accuracy and determinism. The HMIBC instruction also provides built-in communications diagnostics that permit the instruction to automatically reset if the communications from the controlling HMI become unavailable.

Each Logix controller supports up to 256 HMIBC tags and up to 32 PanelView 5500 HMI's to simultaneously communicate and control the instruction.The HMIBC instruction goes active and enables its output when a PanelView 5500 HMI device initiates a button control operation associated with the instance tag of the instruction.

IMPORTANT A PanelView 5500 module is required to use the HMIBC instruction.

To function, the Logix controller I/O configuration must include all of the PanelView 5500 HMIs that need to interact with the HMIBC instruction. Additionally, the application created for each PanelView 5500 HMI must include button actions configured to reference each tag associated with the HMIBC instructions.

ATTENTION: Execute this instruction at least once per scan, and do not jump over.

The HMIBC data type:

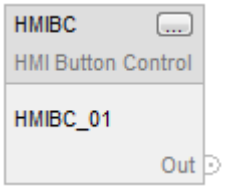
- Is available at Controller and Program scope.
- Is not available within Add-On Instruction scope.
- Is used in a Jump to Subroutine (JSR).
- Cannot be used with input and output program parameters
- Is not available within a safety program.
- Must have an external access value of Read/Write. You are not given the option to choose other external access values.

The HMIBC tag has import and export formats for .L5K, .L5X, and .CSV.

Available Language

Ladder Diagram

Function Block



Tip: For the HMIBC tag, use only the Out parameter, and optionally, the ProgFB parameter in Function Block diagrams.

Structured Text

HMIBC (HMIBC tag)

Operands

These operands are located on the instruction.

Operand	Type	Format	Description
HMIBC tag	HMIBC	tag	Goes active when the data bit is set

HMIBC Structure

Input parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute.
Prog FB	BOOL	Program Feedback. This value is not processed by the instruction, but transmitted to all registered HMI devices. The purpose or meaning of this value is user defined. For example, use this to determine if the expected action actually executes when pressing the button and displays that status on the HMI device.

Output parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled.
Button State	BOOL	Cleared to false when no registered HMI device buttons are pressed. Set to true when at least one registered HMI button is pressed. Default value is false.
Out	BOOL	When EnableIn is true: Cleared to false when none of the registered HMI devices buttons are pressed. Set to true when at least one registered HMI button is pressed. When EnableIn is false : Cleared to false Default value is false.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition	Action Taken
Prescan	The rung-condition-out is set to false.
Rung-condition-in is false	The rung-condition-out is set to false.
Rung-condition-in is true	The rung-condition-out is set to true if any HMI device buttons control operation associated with the instruction instance tag are pressed. Otherwise, rung-condition-out is set to false.
Postscan	The rung-condition-out is set to false.

Function Block

Condition/State	Action Taken
Prescan	N/A
Tag.EnableIn is false	The instruction does not execute.

Condition/State	Action Taken
Tag.EnableIn is true	The instruction does not execute.
Instruction first scan	N/A
Instruction first run	N/A
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	The instruction executes.
Normal Execution	The instruction executes.
Postscan	The instruction executes.

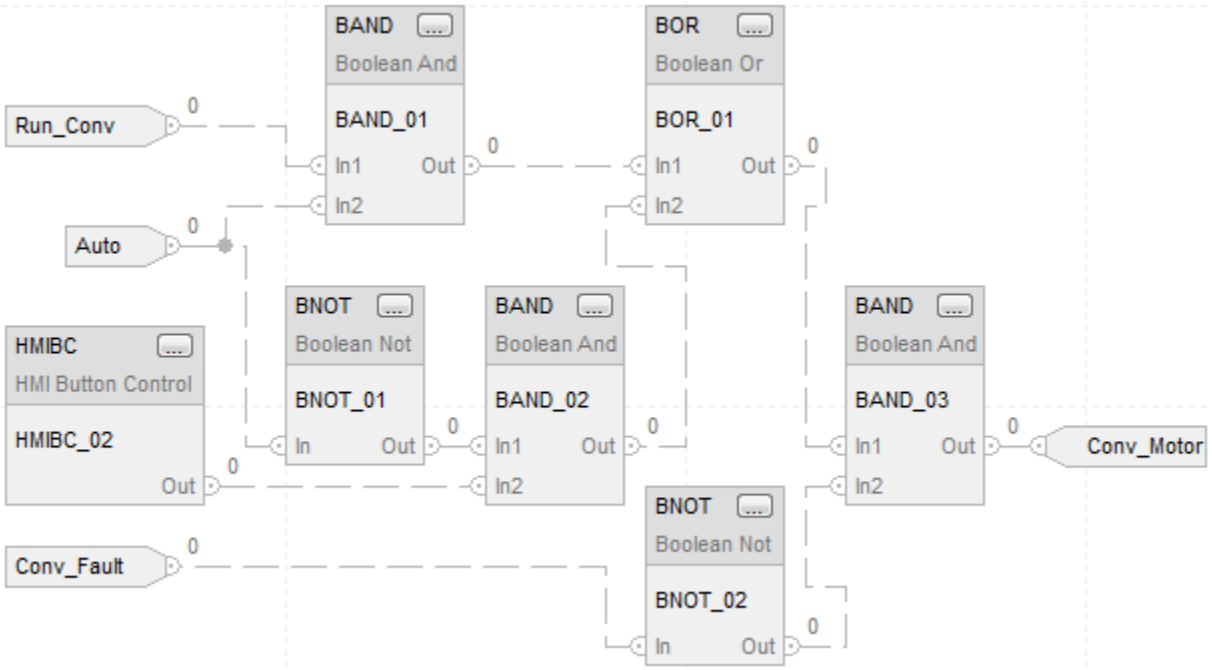
Examples

Ladder Diagram

- An HMIBC instruction is an input instruction and cannot be placed on a rung by itself.
- An HMIBC instruction is highlighted when active.

Function Block

The following example shows the HMIBC instruction as it appears in a function block diagram.



Structured Text

HMIBC (HMIBC_Conv);

IF(((Auto AND Run_Conv) Or (NOT Auto AND HMIBC_Conv.Out)) AND NOT Conv_Fault)

THEN Conv_Motor: = 1;

ELSE Conv_Motor : = 0;

END_IF;

See also

Index Through Arrays on page 1094

Filter

Filter Instructions

The Filter instructions include these instructions:

Available Instructions

Ladder Diagram

This instruction is not available in Ladder Diagram

Function Block and Structured Text

DERV	HPF	LDL2	LPF	NTCH

If you want to	Use this instruction
Calculate the amount of change of a signal over time in per-second units.	DERV
Filter input frequencies that are below the cutoff frequency.	HPF
Filter with a pole pair and a zero pair.	LDL2
Filter input frequencies that are above the cutoff frequency.	LPF
Filter input frequencies that are at the notch frequency.	NTCH

See also

Drives Instructions on page 823

Logical and Move Instructions on page 955

Process Control Instructions on page 17

Select/Limit Instructions on page 903

Statistical Instructions on page 935

Derivative (DERV)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

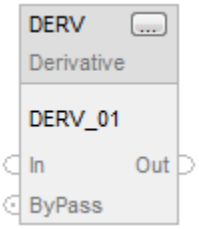
The DERV instruction calculates the amount of change of a signal over time in per-second units.

Available Languages

Ladder Diagram

This instruction is not available for ladder diagram.

Function Block



Structured Text

DERV(DERV_tag);

Operands

Function Block

Operand	Type	Format	Description
DERV tag	DERIVATIVE	structure	DERV structure

DERIVATIVE Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0
Gain	REAL	Derivative multiplier Valid = any float Default = 1.0

Input Parameter	Data Type	Description
ByPass	BOOL	Request to bypass the algorithm. When ByPass is true, the instruction sets Out = In. Default is false.
TimingMode	DINT	Selects timing execution mode. 0 = periodic mode 1 = oversampling mode 2 = Real time sampling mode For more information about timing modes, see Function Block Attributes Valid = 0 to 2 Default = 0
OversampleDT	REAL	Execution time for oversample mode. Valid = 0 to 4194.303 seconds Default = 0
RTSTime	DINT	Module update period for real time sampling mode Valid = 1 to 32,767ms Default = 1
RTSTimeStamp	DINT	Module time stamp value for real time sampling mode. Valid = 0 to 32,767ms Default = 0

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if Out overflows.
Out	REAL	The calculated output of the algorithm.
DeltaT	REAL	Elapsed time between updates. This is the elapsed time in seconds used by the control algorithm to calculate the process output.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
TimingModeInv (Status.27)	BOOL	Invalid TimingMode value. For more information about timing modes, see Function Block Attributes.
RTSMissed (Status.28)	BOOL	Only used in real time sampling mode. Set to true when ABS (DeltaT - RTSTime) > 1 millisecond.
RTSTimeInv (Status.29)	BOOL	Invalid RTSTime value.
RTSTimeStampInv (Status.30)	BOOL	Invalid RTSTimeStamp value.
DeltaTInv (Status.31)	BOOL	Invalid DeltaT value.

Structured Text

Operand	Type	Format	Description
DERV tag	DERIVATIVE	structure	DERV structure

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Description

The DERV instruction supports a bypass input that lets you stop calculating the derivative and pass the signal directly to the output.

When Bypass is	The instruction uses this equation
Cleared and DeltaT > 0	$Out = Gain\frac{In_n - In_{n-1}}{DeltaT}$ $In_{n-1} = In_n$ where DeltaT is in seconds
Set	$Out = In_n$ $In_{n-1} = In_n$

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false. Structured Text: NA
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	Recalculate coefficients.
Postscan	EnableIn and EnableOut bits are cleared to false.

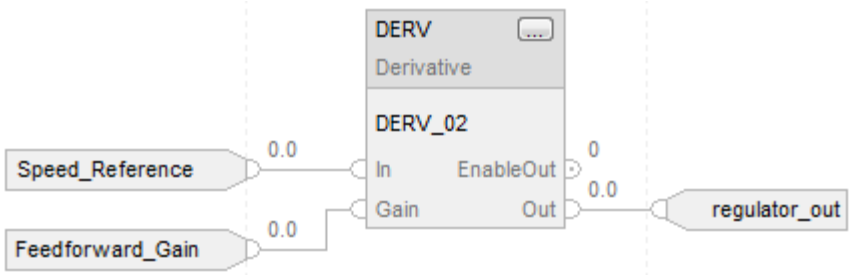
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Examples

This example is the minimal legal programming of the DERV function block and is only used to show the neutral text and generated code for this instruction. This is for internal purposes only and is not a testable case.

Function Block



Structured Text

```
DERV_01.In := Speed_Reference;
DERV_01.Gain := Feedforward_Gain;
DERV(DERV_01);

PI_01.In := Speed_Reference - Speed_feedback;
PI_01.Kp := Proportional_Gain;
PI_01.Wld := Integral_Gain;
PI(PI_01);

regulator_out := DERV_01.Out + PI_01.Out;
```

See also

Function Block Attributes on page 1043

Common Attributes on page 1083

Structured Text Syntax on page 1057

High Pass Filter (HPF)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380,

CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

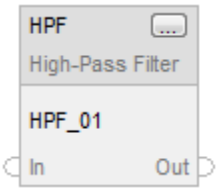
The HPF instruction provides a filter to attenuate input frequencies that are below the cutoff frequency.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

HPF(HPF_tag);

Operands

Function Block

Operand	Type	Format	Description
HPF tag	FILTER_HIGH_PASS	structure	HPF structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0
Initialize	BOOL	Request to initialize filter control algorithm. When true, the instruction sets Out = In. Default is false.

Input Parameter	Data Type	Description
WLead	REAL	The lead frequency in radians/second. If WLead < minimum or WLead > maximum, the instruction sets the appropriate bit in Status and limits WLead. Valid = see Description section below for valid ranges. Default = 0.0
Order	REAL	Order of the filter. Order controls the sharpness of the cutoff. If Order is invalid, the instruction sets the appropriate bit in Status and uses Order = 1. Valid = 1 to 3 Default = 1
OversampleDT	REAL	Execution time for oversample mode. Valid = 0 to 4194.303 seconds Default = 0
RTSTime	DINT	Module update period for real time sampling mode Valid = 1 to 32,767ms Default = 1
RTSTimeStamp	DINT	Module time stamp value for real time sampling mode. Valid = 0 to 32,767ms Default = 0

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if Out overflows.
Out	REAL	The calculated output of the algorithm.
DeltaT	REAL	Elapsed time between updates. This is the elapsed time in seconds used by the control algorithm to calculate the process output.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
WLeadInv (Status.1)	BOOL	WLead < minimum value or WLead > maximum value.
OrderInv (Status.2)	BOOL	Invalid Order value.
TimingModeInv (Status.27)	BOOL	Invalid TimingMode value. For more information about timing modes, see Function Block Attributes.
RTSMissed (Status.28)	BOOL	Only used in real time sampling mode. Set to true when ABS (DeltaT - RTSTime) > 1 millisecond.
RTSTimeInv (Status.29)	BOOL	Invalid RTSTime value.
RTSTimeStampInv (Status.30)	BOOL	Invalid RTSTimeStamp value.
DeltaTInv (Status.31)	BOOL	Invalid DeltaT value.

Structured Text

Operand	Type	Format	Description
HPF tag	FILTER_HIGH_PASS	Structure	HPF structure

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Description

The HPF instruction uses the Order parameter to control the sharpness of the cutoff. The HPF instruction is designed to execute in a task where the scan rate remains constant.

The HPF instruction uses these equations:

When:	The instruction uses this transfer function:
Order = 1	$\frac{s}{s+\omega}$
Order = 2	$\frac{s^2}{s^2+\sqrt{2}\times s\times \omega+\omega^2}$
Order = 3	$\frac{s^3}{s^3+(2\times s^2\times \omega)+2\times s\times \omega^2+\omega^3}$

with these parameter limits (where DeltaT is in seconds):

Parameter	Limitations
WLead first order LowLimit	$\frac{0.0000001}{DeltaT}$
WLead second order LowLimit	$\frac{0.00001}{DeltaT}$
WLead third order LowLimit	$\frac{0.001}{DeltaT}$
HighLimit	$\frac{0.7\pi}{DeltaT}$

Whenever the value computed for the output is invalid, NAN, or $\pm$ INF, the instruction sets Out = the invalid value. When the value computed for the output becomes valid, the instruction initializes the internal parameters and sets Out = In.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	Recalculate coefficients.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

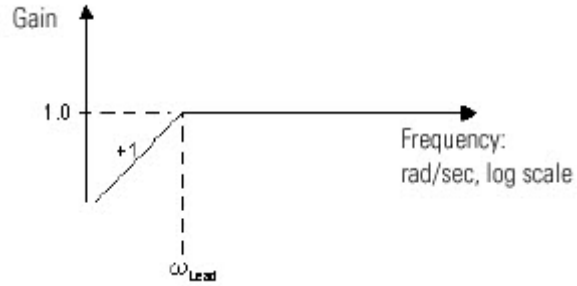
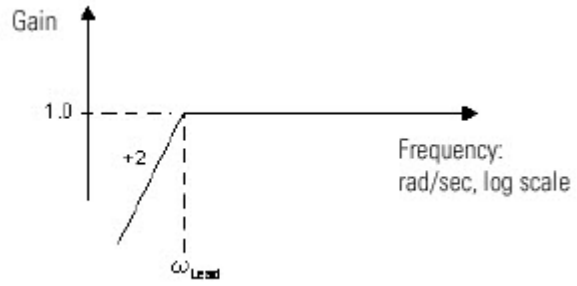
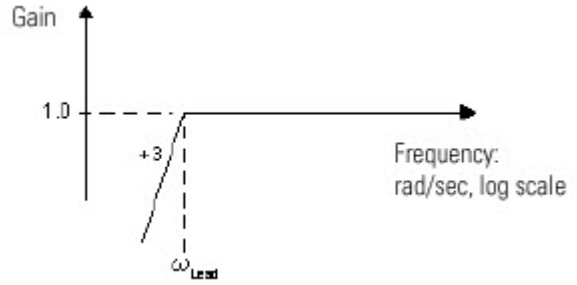
Example

The HPF instruction attenuates signals that occur below the configured cutoff frequency. This instruction is typically used to filter low frequency "noise" or disturbances that originate from either electrical or mechanical sources. You can select a specific order of the filter to achieve various degrees of attenuation. Note that higher orders increase the execution time for the filter instruction.

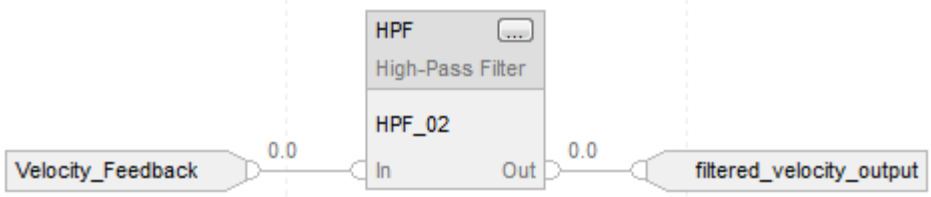
The following graphs illustrate the effect of the various orders of the filter for a given cutoff frequency. For each graph, ideal asymptotic approximations are given with gain and frequency in logarithmic scales. The actual response of the filter approaches these curves but does not exactly match these curves.

This example is the minimal legal programming of the HPF function block and is only used to show the neutral text and generated code for this instruction. This is for internal purposes only and is not a testable case.

Filter	Graph

1st order filter	
2nd order filter	
3rd order filter	

Function Block



Structured Text

```
HPF_01.In := Velocity_Feedback;
HPF_01.WLead := Cutoff_frequency;
HPF_01.Order := 2;

HPF(HPF_01);

filtered_velocity_output := HPF_01.Out
```

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Low Pass Filter (LPF)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

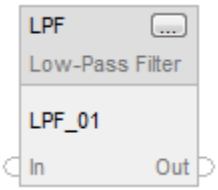
The LPF instruction provides a filter to attenuate input frequencies that are above the cutoff frequency.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

LPF(LPF_tag);

Operands

Function Block

Operand	Type	Format	Description
LPF tag	FILTER_LOW_PASS	Structure	LPF structure

FILTER_LOW_PASS Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0
Initialize	BOOL	Request to initialize filter control algorithm. When true, the instruction sets Out = In. Default is false.
WLag	REAL	The lag frequency in radians/second. If WLag < minimum or WLag > maximum, the instruction sets the appropriate bit in Status and limits WLag. Valid = see Description section below for valid ranges Default = 0.0
Order	REAL	Order of the filter. Order controls the sharpness of the cutoff. If Order is invalid, the instruction sets the appropriate bit in Status and uses Order = 1. Valid = 1 to 3 Default = 1
TimingMode	DINT	Selects timing execution mode. 0 = Period mode 1 = Oversample mode 2 = Real-time sampling mode For more information about timing modes, see Function Block Attributes. Valid = 0 to 2 Default = 0
OversampleDT	REAL	Execution time for oversample mode. Valid = 0 to 4194.303 seconds Default = 0
RTSTime	DINT	Module update period for real time sampling mode Valid = 1 to 32,767ms Default = 1
RTSTimeStamp	DINT	Module time stamp value for real time sampling mode. Valid = 0 to 32,767ms Default = 0

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled Cleared to false if Out overflows.
Out	REAL	The calculated output of the algorithm.

Output Parameter	Data Type	Description
DeltaT	REAL	Elapsed time between updates. This is the elapsed time in seconds used by the control algorithm to calculate the process output.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
WLagInv (Status.1)	BOOL	WLag < minimum value or WLag > maximum value.
OrderInv (Status.2)	BOOL	Invalid Order value.
TimingModeInv (Status.27)	BOOL	Invalid TimingMode value.
RTSMissed (Status.28)	BOOL	Only used in real time sampling mode. Set to true when ABS(DeltaT – RTSTime) > 1 millisecond.
RTSTimeInv (Status.29)	BOOL	Invalid RTSTime value.
RTSTimeStampInv (Status.30)	BOOL	Invalid RTSTimeStamp value.
DeltaTInv (Status.31)	BOOL	Invalid DeltaT value.

Structured Text

Operand	Type	Format	Description
LPF tag	FILTER_LOW_PASS	structure	LPF structure

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Description

The LPF instruction uses the Order parameter to control the sharpness of the cutoff. The LPF instruction is designed to execute in a task where the scan rate remains constant.

The LPF instruction uses these equations:

When:	The instruction uses this Laplace transfer function:
Order = 1	$\frac{\omega}{s+\omega}$
Order = 2	$\frac{\omega^2}{s^2+\sqrt{2}\times s\times \omega+\omega^2}$
Order = 3	$\frac{\omega^3}{s^3+(2\times s^2\times \omega)+(2\times s\times \omega^2)\times \omega^3}$

with these parameters limits (where DeltaT is in seconds):

Parameter	Limitations
WLag first order LowLimit	$\frac{0.0000001}{DeltaT}$
WLag second order LowLimit	$\frac{0.00001}{DeltaT}$
WLag third order LowLimit	$\frac{0.001}{DeltaT}$
HighLimit	$\frac{0.7\pi}{DeltaT}$

Whenever the value computed for the output is invalid, NAN, or ± INF, the instruction sets Out = the invalid value. When the value computed for the output becomes valid, the instruction initializes the internal parameters and sets Out = In.

Affects Math Status Flags

Controllers	Affects Math Status Flags
ControlLogix 5580	No
CompactLogix 5370, ControlLogix 5570	Yes for the output

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	Recalculate coefficients.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.

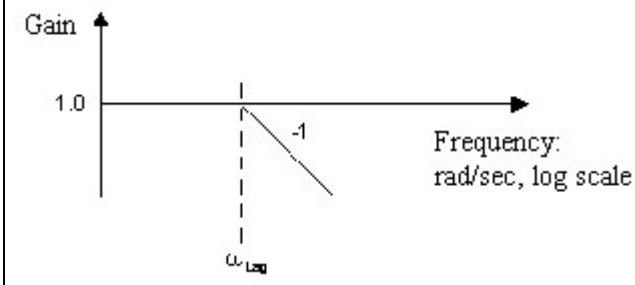
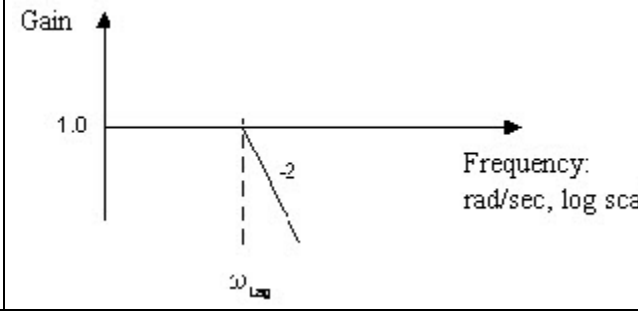
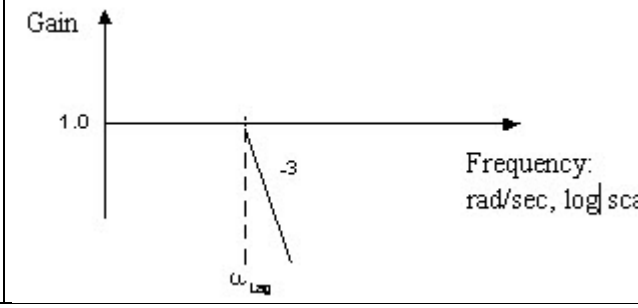
Condition/State	Action Taken
Postscan	See Postscan in the Function Block table.

Example

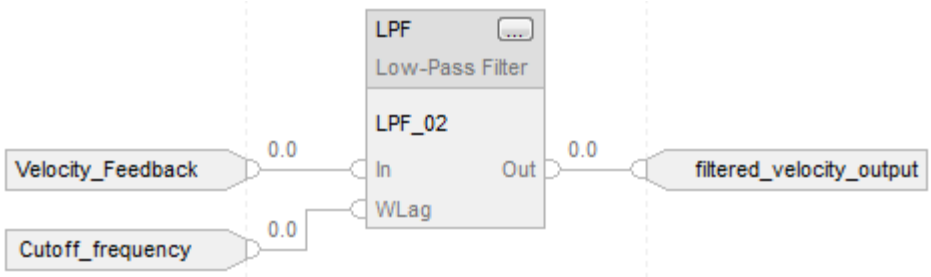
The LPF instruction attenuates signals that occur above the configured cutoff frequency. This instruction is typically used to filter out high frequency "noise" or disturbances that originate from either electrical or mechanical sources. You can select a specific order of the filter to achieve various degrees of attenuation. Note that higher orders increase the execution time for the instruction.

The following graphs illustrate the effect of the various orders of the filter for a given cutoff frequency. For each graph, ideal asymptotic approximations are given with gain and frequency in logarithmic scales. The actual response of the filter approaches these curves but does not exactly match these curves.

This example is the minimal legal programming of the LPF function block and is only used to show the neutral text and generated code for this instruction. This is for internal purposes only and is not a testable case.

Filter	Graph
1st order filter	
2nd order filter	
3rd order filter	

Function Block



Structured Text

```
LPF_01.In := Velocity_Feedback;
LPF_01.WLag := Cutoff_frequency;

LPF(LPF_01);

filtered_velocity_output := LPF_01.Out;
```

See also

Function Block Attributes on page 1043

Common Attributes on page 1083

Structured Text Syntax on page 1057

Notch Filter (NTCH)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

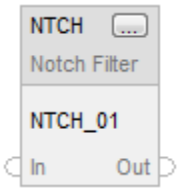
The NTCH instruction provides a filter to attenuate input frequencies that are at the notch frequency.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

NTCH(NTCH_tag);

Operands

Function Block

Operand	Type	Format	Description
NTCH tag	FILTER_NOTCH	Structure	NTCH structure

FILTER_NOTCH Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0
Initialize	BOOL	Request to initialize filter control algorithm. When true, the instruction sets Out = In. Default is false.
WNotch	REAL	The filter center frequency in radians/second. If WNotch < minimum or WNotch > maximum, the instruction sets the appropriate bit in status and limits WNotch. Valid = see Description section below for valid ranges Default = maximum positive float

Input Parameter	Data Type	Description
QFactor	REAL	Controls the width and depth ratio. Set QFactor = 1 / (2*desired damping factor). If QFactor < minimum or QFactor > maximum value, the instruction sets the appropriate bit in Status and limits QFactor. Valid = 0.5 to 100.0 Default = 0.5
Order	REAL	Order of the filter. Order controls the sharpness of the cutoff. If Order is invalid, the instruction sets the appropriate bit in Status and uses Order = 2. Valid = 2 or 4 Default = 2
TimingMode	DINT	Selects timing execution mode. 0 = Periodic mode 1 = Oversample mode 2 = Real time sampling mode2 For more information about timing modes, see Function Block Attributes. Valid = 0 to 2 Default = 0
OversampleDT	REAL	Execution time for oversample mode. Valid = 0 to 4194.303 seconds Default = 0
RTSTime	DINT	Module update period for real time sampling mode Valid = 1 to 32,767ms Default = 1
RTSTimeStamp	DINT	Module time stamp value for real time sampling mode. Valid = 0 to 32,767ms Default = 0

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if Out overflows.
Out	REAL	The calculated output of the algorithm.
DeltaT	REAL	Elapsed time between updates. This is the elapsed time in seconds used by the control algorithm to calculate the process output.
Status	DINT	Status of the function block.

InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
WNotchInv (Status.1)	BOOL	WNotch < minimum or WNotch > maximum
QFactorInv (Status.2)	BOOL	QFactor < minimum or QFactor > maximum
OrderInv (Status.3)	BOOL	Invalid Order value.
TimingModeInv (Status.27)	BOOL	Invalid TimingMode value. For more information about timing modes, see Function Block Attributes.
RTSMissed (Status.28)	BOOL	Only used in real time sampling mode. Set to true when ABS (DeltaT - RTSTime) > 1 millisecond.
RTSTimeInv (Status.29)	BOOL	Invalid RTSTime value.
RTSTimeStampInv (Status.30)	BOOL	Invalid RTSTimeStamp value.
DeltaTInv (Status.31)	BOOL	Invalid DeltaT value.

Structured Text

Operand	Type	Format	Description
NTCH tag	FILTER_NOTCH	structure	NTCH structure

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Description

The NTCH instruction uses the Order parameter to control the sharpness of the cutoff. The QFactor parameter controls the width and the depth ratio of the notch. The NTCH instruction is designed to execute in a task where the scan rate remains constant.

The NTCH instruction uses this equation:

$$\frac{\left(s^2+\omega^2\right)^i}{\left(s^2+s\times\frac{\omega}{Q}+\omega^2\right)^i}$$

where i is the Order operator with these parameters limits (where DeltaT is in seconds):

Parameter	Limitations
WNotch second order LowLimit	$\frac{0.0000001}{DeltaT}$

WNotch fourth order LowLimit	$\frac{0.001}{DeltaT}$
HighLimit	$\frac{0.7\pi}{DeltaT}$
QFactor	LowLimit = 0.5 HighLimit = 100.0

Whenever the value computed for the output is invalid, NAN, or ± INF, the instruction sets Out = the invalid value. When the value computed for the output becomes valid, the instruction initializes the internal parameters and sets Out = In.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	Recalculate coefficients.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

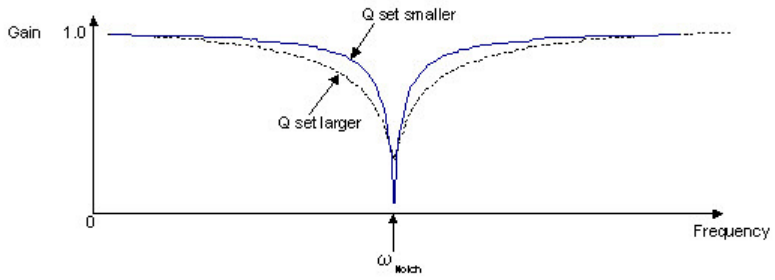
Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

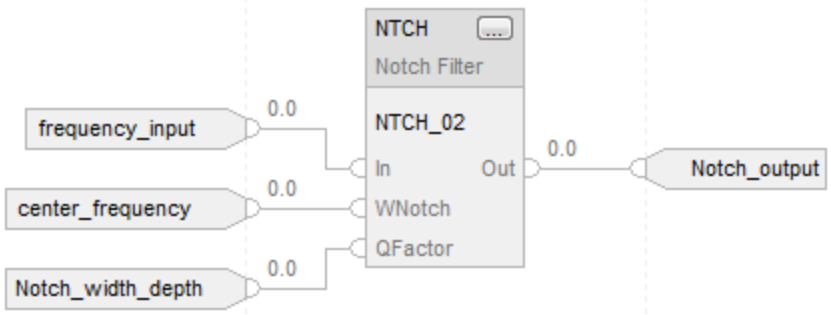
The NTCH instruction attenuates a specific resonance frequency. Typically, these resonance frequencies are directly in the range of response being regulated by the closed loop control system. Often, they are generated by loose mechanical linkages that cause backlash and vibration in the system. Although the best solution is to correct the mechanical compliance in the machinery, the notch filter can be used to soften the effects of these signals in the closed loop regulating scheme.

The following diagram shows the ideal gain curve over a frequency range for a specific center frequency and Q factor. As increases, the notch becomes wider and shallower. As decreases; the notch becomes deeper and narrower. The instruction may be set for an order of 2 or an order of 4. Higher orders take more execution time.

This example is the minimal legal programming of the NTCH function block and is only used to show the neutral text and generated code for this instruction. This is for internal purposes only and is not a testable case.



Function Block



Structured Text

```
NTCH_01.In := frequency_input;
NTCH_01.WNotch := center_frequency;
NTCH_01.QFactor := Notch_width_depth;
NTCH(NTCH_01);
```

```
Notch_output := NTCH_01.Out;
```

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Second-Order Lead Lag (LDL2)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

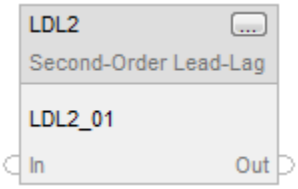
The LDL2 instruction provides a filter with a pole pair and a zero pair. The frequency and damping of the pole and zero pairs are adjustable. The pole or zero pairs can be either complex (damping less than unity) or real (damping greater than or equal to unity).

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

LDL2(LDL2_tag);

Operands

Function Block

Operand	Type	Format	Description
LDL2 tag	LEAD_LAG_SEC_ORDER	Structure	LDL2 structure

LEAD_LAG_SEC_ORDER Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0
Initialize	BOOL	Request to initialize filter control algorithm. When true, the instruction sets Out = In. Default is cleared.
WLead	REAL	The lead corner frequency in radians/second. If WLead < minimum or WLead > maximum, the instruction sets the appropriate bit to true in Status and limits WLead. If the WLag:WLead ratio > maximum ratio, the instruction sets the appropriate bit in Status to true and limits WLag Valid = see Description section below for valid ranges. Default = 0.0
WLag	REAL	The lag corner frequency in radians/second. If WLag < minimum or WLag > maximum, the instruction sets the appropriate bit to true in Status and limits WLag. If the WLag:WLead ratio > maximum ratio, the instruction sets the appropriate bit to true in Status and limits WLag. Valid = see Description section below for valid ranges Default = 0.0
ZetaLead	REAL	Second order lead damping factor. Only used when Order = 2. If ZetaLead < minimum or ZetaLead > maximum, the instruction sets the appropriate bit to true in Status and limits ZetaLead. Valid = 0.0 to 4.0 Default = 0.0
ZetaLag	REAL	Second order lag-damping factor. Only used when Order = 2. If ZetaLag < minimum or ZetaLag > maximum, the instruction sets the appropriate bit to true in Status and limits ZetaLag. Valid = 0.05 to 4.0 Default = 0.05

Input Parameter	Data Type	Description
Order	REAL	Order of the filter. Selects the first or second order filter algorithm. If invalid, the instruction sets the appropriate bit to true in Status and uses Order = 2. Valid = 1 to 2 Default = 2
TimingMode	DINT	Selects timing execution mode. 0 = Periodic mode 1 = Oversample mode 2 = Real time sampling mode Valid = 0 to 2 Default = 0
RTSTimeStamp	DINT	Module time stamp value for real time sampling mode. Valid = 0 to 32,767ms Default = 0

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if Out overflows.
Out	REAL	The calculated output of the algorithm.
DeltaT	REAL	Elapsed time between updates. This is the elapsed time in seconds used by the control algorithm to calculate the process output.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
WLeadInv (Status.1)	BOOL	WLead < minimum value or WLead > maximum value.
WLagInv (Status.2)	BOOL	WLag < minimum value or WLag > maximum value.
ZetaLeadInv (Status.3)	BOOL	Lead damping factor < minimum value or lead damping factor > maximum value.
ZetaLagInv (Status.4)	BOOL	Lag damping factor < minimum value or lag damping factor > maximum value.
OrderInv (Status.5)	BOOL	Invalid Order value.
WLagRatioInv (Status.6)	BOOL	WLag:WLead ratio greater than maximum value.

Output Parameter	Data Type	Description
TimingModeInv (Status.27)	BOOL	Invalid TimingMode value. For more information about timing modes, see Function Block Attributes.
RTSMissed (Status.28)	BOOL	Only used in real time sampling mode. Set when ABS (DeltaT - RTSTime) > 1 millisecond.
RTSTimeInv (Status.29)	BOOL	Invalid RTSTime value.
RTSTimeStampInv (Status.30)	BOOL	Invalid RTSTimeStamp value.
DeltaTInv (Status.31)	BOOL	Invalid DeltaT value.

Structured Text

Operand	Type	Format	Description
LDL2 tag	LEAD_LAG_SEC_ORDER	structure	LDL2 structure

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Description

The LDL2 instruction filter is used in reference forcing and feedback forcing control methodologies. The LDL2 instruction is designed to execute in a task where the scan rate remains constant.

The LDL2 instruction uses these equations:

When:	The instruction uses this Laplace transfer function:
Order = 1	$H(s) = \frac{\frac{s}{\omega_{Lead}} + 1}{\frac{s}{\omega_{Lag}} + 1}$
Order = 2	$H(s) = \frac{\frac{s^2}{\omega_{Lead}^2} + \frac{2 \times \xi_{Lead} \times s}{\omega_{Lead}} + 1}{\frac{s^2}{\omega_{Lag}^2} + \frac{2 \times \xi_{Lag} \times s}{\omega_{Lag}} + 1}$ Normalize the filter such that $\omega_{Lead} = 1$ $H(s) = \frac{s^2 + 2 \times \xi_{Lead} \times s + 1}{\frac{s^2}{\omega_{Lag}^2} + \frac{2 \times \xi_{Lag} \times s}{\omega_{Lag}} + 1}$

with these parameter limits (where DeltaT is in seconds):

Parameter	Limitations
WLead first order LowLimit	$\frac{0.0000001}{DeltaT}$

WLead second order LowLimit	$\frac{0.00001}{DeltaT}$
HighLimit	$\frac{0.7\pi}{DeltaT}$
WLead:WLag ratio	If WLead > WLag, no limitations If WLag > WLead: • No minimum limitation for WLag:WLead • First order maximum for WLag:WLead = 40:1 and the instruction limits WLag to enforce this ratio • Second order maximum for WLag:WLead = 10:1 and the instruction limits WLag to enforce this ratio
ZetaLead second order only	LowLimit = 0.0 HighLimit = 4.0
ZetaLag second order only	LowLimit = 0.05 HighLimit = 4.0

Whenever the value computed for the output is invalid, NAN, or ± INF, the instruction sets Out = the invalid value. When the value computed for the output becomes valid, the instruction initializes the internal parameters and sets Out = In.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	Recalculate coefficients.
Postscan	EnableIn and EnableOut bits are cleared to false.

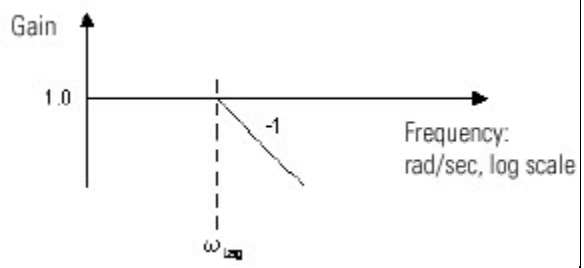
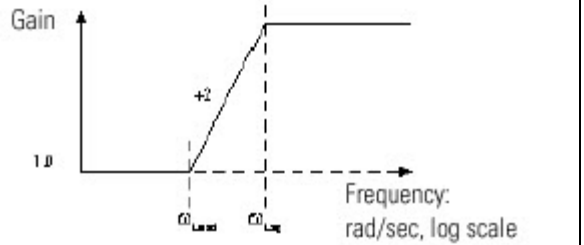
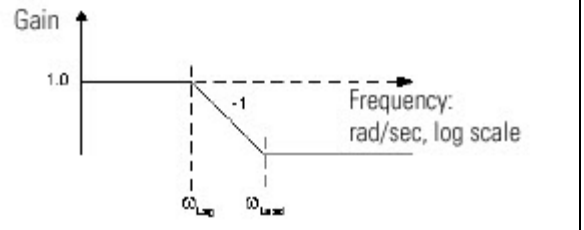
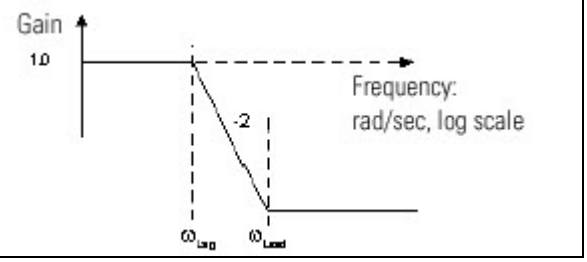
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

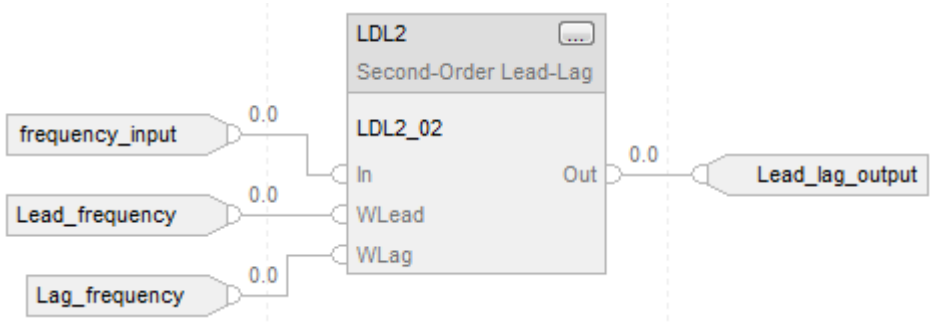
Example

The LDL2 instruction can attenuate between two frequencies or can amplify between two frequencies, depending on how you configure the instruction. Since the Lead and Lag frequencies can be set to values that are larger or smaller than each other, this instruction may behave as a Lead-Lag block, or, as a Lag-Lead block, depending on which frequency is configured first. Note that higher orders increase the execution time for the filter instruction.

This example is the minimal legal programming of the LDL2 function block and is only used to show the neutral text and generated code for this instruction. This is for internal purposes only and is not a testable case.

Filter	Graph
1st order lead-lag (wLead < wLag)	
2nd order lead-lag (wLead < wLag)	
1st order lead-lag (wLag < wLead)	
2nd order lead-lag (wLag < wLead)	

Function Block



Structured Text

```
LDL2_01.In := frequency_input;
LDL2_01.WLead :=
Lead_frequency;
LDL2_01.WLag := Lag_frequency;
LDL2(LDL2_01);
Lead_lag_output := LDL2_01.Out;
```

See also

Function Block Attributes on page 1043

Common Attributes on page 1083

Structured Text Syntax on page 1057

Select_Limit Instructions

Select/Limit Instructions

The Select/Limit instructions include these instructions:

Available Instructions

Ladder Diagram

This instruction is not available in Ladder Diagram.

Function Block and Structured Text

ESEL	HLL	MUX	RLIM	SEL	SNEG	SSUM

If you want to	Use this instruction
Select one of as many as six inputs.	ESEL
Limit an analog input between two values.	HLL
Select one of eight inputs.	MUX
Limit the amount of change of a signal over time.	RLIM
Select one of two inputs.	SEL
Select between the input value and the negative of the input value.	SNEG
Select real inputs to be summed.	SSUM

See also

Filter Instructions on page 875

Logical and Move Instructions on page 955

Process Control Instructions on page 17

Drives Instructions on page 823

Statistical Instructions on page 935

Enhanced Select (ESEL)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380,

CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The ESEL instruction lets you select one of as many as six inputs. Selection options include:

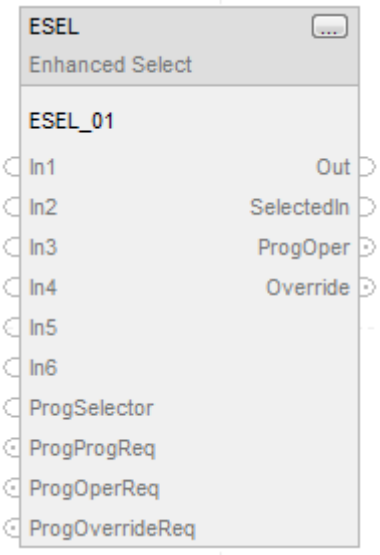
- Manual select (by operator or by program)
- High select
- Low select
- Median select
- Average (mean) select

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

ESEL(ESEL_tag);

Operands

Function Block

Operand	Type	Format	Description
ESEL tag	SELECT_ENHANCED	Structure	ESEL structure

SELECT_ENHANCED Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
In1	REAL	The first analog signal input to the instruction. Valid = any float Default = 0.0
In2	REAL	The second analog signal input to the instruction. Valid = any float Default = 0.0
In3	REAL	The third analog signal input to the instruction. Valid = any float Default = 0.0
In4	REAL	The fourth analog signal input to the instruction. Valid = any float Default = 0.0
In5	REAL	The fifth analog signal input to the instruction. Valid = any float Default = 0.0
In6	REAL	The sixth analog signal input to the instruction. Valid = any float Default = 0.0
In1Fault	BOOL	Bad health indicator for In1. If In1 is read from an analog input, then In1Fault is normally controlled by the fault status on the analog input. If all the InnFault inputs are true, the instruction sets the appropriate bit in Status, the control algorithm is not executed, and Out is not updated. Default = false

Input Parameter	Data Type	Description
In2Fault	BOOL	Bad health indicator for In2. If In2 is read from an analog input, then In2Fault is normally controlled by the fault status on the analog input. If all the InnFault inputs are true, the instruction sets the appropriate bit in Status, the control algorithm is not executed, and Out is not updated. Default = false
In3Fault	BOOL	Bad health indicator for In3. If In3 is read from an analog input, then In3Fault is normally controlled by the fault status on the analog input. If all the InnFault inputs are true, the instruction sets the appropriate bit in Status, the control algorithm is not executed, and Out is not updated. Default = false
In4Fault	BOOL	Bad health indicator for In4. If In4 is read from an analog input, then In4Fault is normally controlled by the fault status on the analog input. If all the InnFault inputs are true, the instruction sets the appropriate bit in Status, the control algorithm is not executed, and Out is not updated. Default = false
In5Fault	BOOL	Bad health indicator for In5. If In5 is read from an analog input, then In5Fault is normally controlled by the fault status on the analog input. If all the InnFault inputs are true, the instruction sets the appropriate bit in Status, the control algorithm is not executed, and Out is not updated. Default = false
In6Fault	BOOL	Bad health indicator for In6. If In6 is read from an analog input, then In6Fault is normally controlled by the fault status on the analog input. If all the InnFault inputs are true, the instruction sets the appropriate bit in Status, the control algorithm is not executed, and Out is not updated. Default = false

Input Parameter	Data Type	Description
InsUsed	DINT	Number of inputs used. This defines the number of inputs the instruction uses. The instruction considers only In1 through $In_{InsUsed}$ in high select, low select, median select, and average select modes. If this value is invalid, the instruction sets the appropriate bit in status. The instruction does not update Out if InsUsed is invalid and if the instruction is not in manual select mode and if Override is cleared. Valid =1 to 6 Default = 1
Selector Mode	DINT	Selector mode input. This value determines the action of the instruction. 0 = manual select 1 = High select 2 = Low select 3 = Median select 4 = Average select If this value is invalid, the instruction sets the appropriate bit in Status and does not update Out. Valid = 0 to 4 Default = 0
ProgSelector	DINT	Program selector input. When the selector mode is manual select and the instruction is in Program control, ProgSelector determines which input (In1-In6) to move into Out. If ProgSelector = 0, the instruction does not update Out. If ProgSelector is invalid, the instruction sets the appropriate bit in Status. If invalid and the instruction is in Program control, and the selector mode is manual select or Override is set, the instruction does not update. Out. Valid = 0 to 6 Default = 0
OperSelector	DINT	Operator selector input. When the selector mode is manual select and the instruction is in Operator control, OperSelector determines which input (In1-In6) to move into Out. If OperSelector = 0, the instruction does not update Out. If OperSelector is invalid, the instruction sets the appropriate bit in Status. If invalid and the instruction is in Operator control, and the selector mode is manual select or Override is set, the instruction does not update Out. Valid = 0 to 6 Default = 0

Input Parameter	Data Type	Description
ProgProgReq	BOOL	Program program request. Set to true by the user program to request Program control. Ignored if ProgOperReq is true. Holding this true and ProgOperReq false locks the instruction into Program control. Default is false.
ProgOperReq	BOOL	Program operator request. Set to true by the user program to request Operator control. Holding this true locks the instruction into Operator control. Default is false.
ProgOverrideReq	BOOL	Program override request. Set to true by the user program to request the device to enter Override mode. In Override mode, the instruction will act as a manual select. Default is false.
OperProgReq	BOOL	Operator program request. Set to true by the operator interface to request Program control. The instruction clears this input to false. Default is false.
OperOperReq	BOOL	Operator operator request. Set to true by the operator interface to request Operator control. The instruction clears this input to false. Default is false.
ProgValueReset	BOOL	Reset program control values. When true, all the program request inputs are cleared to false on each execution of the instruction. Default is false.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if Out overflows.
Out	REAL	The calculated output of the algorithm.
SelectedIn	DINT	Number of input selected. The instruction uses this value to display the number of the input currently being placed into the output. If the selector mode is average select, the instruction sets SelectedIn = 0.
ProgOper	BOOL	Program/Operator control indicator. Set to true when in Program control. Cleared to false when in Operator control.
Override	BOOL	Override mode. Set to true when the instruction is in Override mode.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.

InsFaulted (Status.1)	BOOL	InnFault inputs for all the used Inn inputs are true.
InsUsedInv (Status.2)	BOOL	Invalid InsUsed value.
SelectorModeInv (Status.3)	BOOL	Invalid SelectorMode value.
ProgSelectorInv (Status.4)	BOOL	Invalid ProgSelector value.
OperSelectorInv (Status.5)	BOOL	Invalid OperSelector value.

Structured Text

Operand	Type	Format	Description
ESEL tag	SELECT_ENHANCED	structure	ESEL structure

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Description

The ESEL instruction operates as follows:

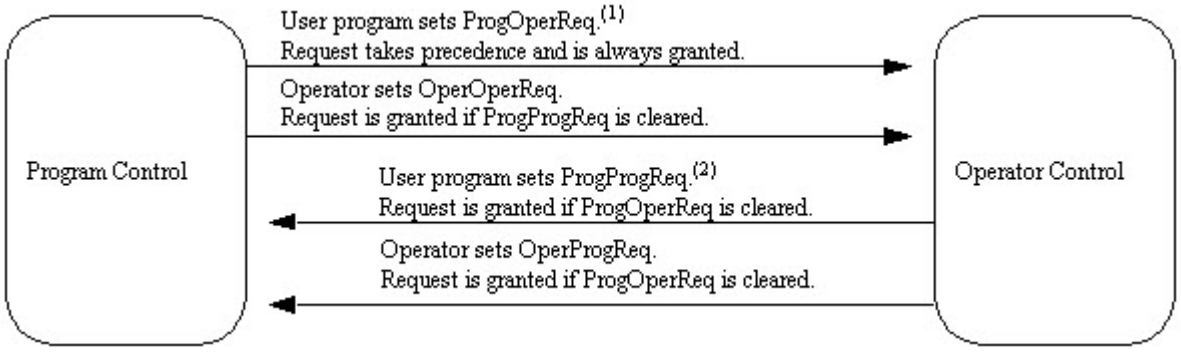
Condition	Action
SelectorMode = 0 (manual select) or Override is true, ProgOper is false and OperSelector is not equal to 0	Out = In[OperSelector] SelectedIn = OperSelector
SelectorMode = 0 (manual select) or Override is true, ProgOper is true and ProgSelector is not equal to 0	Out = In[ProgSelector] SelectedIn = ProgSelector
SelectorMode = 1 (high select) and Override is false	Out = maximum of In[InsUsed] SelectedIn = index to the maximum input value
SelectorMode = 2 (low select) and Override is false	Out = minimum of In[InsUsed] SelectedIn = index to the minimum input value
SelectorMode = 3 (median select) and Override is false	Out = median of In[InsUsed] SelectedIn = index to the median input value
SelectorMode = 4 (average select) and Override is false	Out = average of In[InsUsed] SelectedIn = 0

For SelectorMode 1 through 4, a bad health indication for any of the inputs causes that bad input to be disregarded in the selection. For example, if SelectorMode = 1 (high select) and if In6 had the highest value but had bad health, then the next highest input with good health is moved into the output.

For high or low select mode, if two inputs are equal and are high or low, the instruction outputs the first found input. For median select mode, the median value always represents a value selected from the available inputs. If more than one value could be the median, the instruction outputs the first found input.

Switch Between Program Control and Operator Control

The following diagram shows how the ESEL instruction changes between Program control and Operator control.



(1) You can lock the instruction in Operator control mode by leaving ProgOperReq true.

(2) You can lock the instruction in Program control mode by leaving ProgProgReq true while ProgOperReq is false.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes
Instruction first run	The instruction is set to Operator control.
Instruction first scan	N/A
Postscan	EnableIn and EnableOut bits are cleared to false.

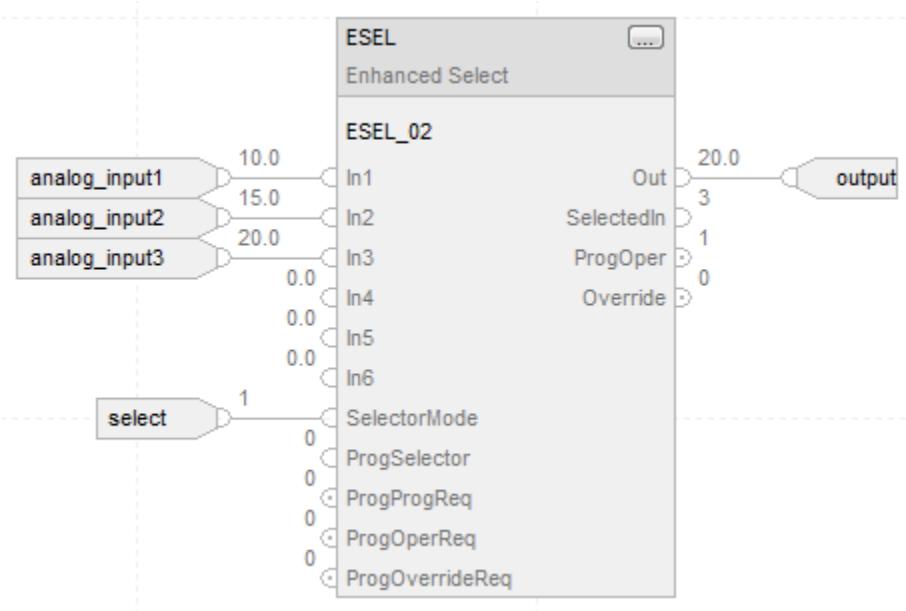
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

This ESEL instruction selects In1, In2, or In3, based on the SelectorMode. In this example, SelectorMode = 1, which means high select. The instruction determines which input value is the greatest and sets Out = greatest In.

Function Block



Structured Text

```
ESEL_01.In1 := analog_input1;
ESEL_01.In2 := analog_input2;
ESEL_01.In3 := analog_input3;
ESEL_01.SelectorMode := 1;
ESEL(ESEL_01);

selected_value := ESEL_01.Out;
```

See also

Function Block Attributes on page 1043

Common Attributes on page 1083

Structured Text Syntax on page 1057

Function Block Faceplate Controls on page 1095

High/Low Limit (HLL)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

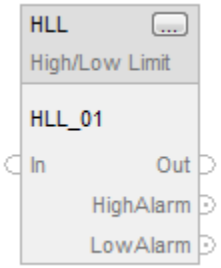
The HLL instruction limits an analog input between two values. You can select high/low, high, or low limits.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

HLL(HLL_tag);

Operands

Function Block

Operand	Type	Format	Description
HLL tag	HL_LIMIT	structure	HLL structure

HL_LIMIT Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0
HighLimit	REAL	The high limit for the Input. If SelectLimit = 0 and HighLimit ≤ LowLimit, the instruction sets the appropriate bit in Status and sets Out = LowLimit. Valid = HighLimit > LowLimit Default = 0.0
LowLimit	REAL	The low limit for the Input. If SelectLimit = 0 and LowLimit ≥ HighLimit, the instruction sets the appropriate bit in Status and sets Out = LowLimit. Valid = LowLimit < HighLimit Default = 0.0
SelectLimit	DINT	Select limit input. This input has three settings: 0 = Use both limits 1= Use high limit 2 = Use low limit If SelectLimit is invalid, the instruction assumes SelectLimit = 0 and sets the appropriate bit in Status. Valid = 0 to 2 Default = 0

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if Out overflows.
Out	REAL	The calculated output of the algorithm.
HighAlarm	BOOL	The high alarm indicator. Set to true when In ≥ HighLimit. The HighAlarm is disabled when SelectLimit is set to 2.
LowAlarm	BOOL	The low alarm indicator. Set to true when In ≤ LowLimit. The LowAlarm is disabled when SelectLimit is set to 1.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.

LimitsInv (Status.1)	BOOL	HighLimit ≤ LowLimit.
SelectLimitInv (Status.2)	BOOL	The value of SelectLimit is not a 0, 1, or 2.

Structured Text

Operand	Type	Format	Description
HLL tag	HL_LIMIT	structure	HLL structure

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Description

The HLL instruction determines the value of the Out using these rules:

Selection	Condition	Action
SelectLimit = 0 (use high and low limits)	In < HighLimit and In > LowLimit	Out = In
	In ≥ HighLimit	Out = HighLimit
	In ≤ LowLimit	Out = LowLimit
	HighLimit ≤ LowLimit	Out = LowLimit
SelectLimit = 1 (use high limit only)	In < HighLimit	Out = In
	In≥HighLimit	Out = HighLimit
SelectLimit = 2 (use low limit only)	In > LowLimit	Out = In
	In ≤ LowLimit	Out = LowLimit

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.

Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	EnableIn and EnableOut bits are cleared to false.

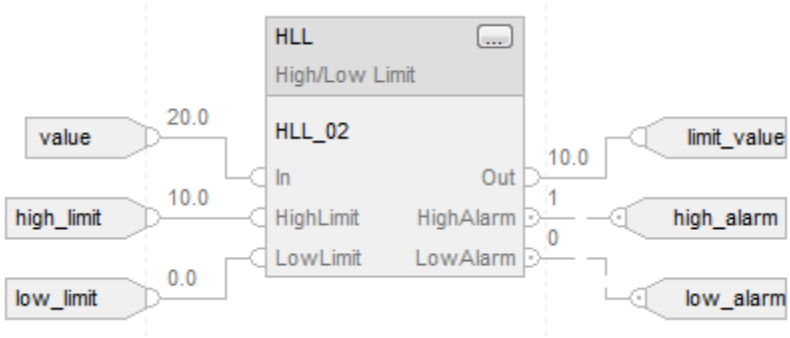
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

This HLL instruction limits In between two values and sets HighAlarm or LowAlarm, if needed when In is outside the limits. The instruction sets Out = limited value of In.

Function Block



Structured Text

HLL_01.In := value;

HLL_01.HighLimit := high_limit;

HLL_01.LowLimit := low_limit;

HLL(HLL_01);

limited_value := HLL_01.Out;

high_alarm := HLL_01.HighAlarm;

low_alarm := HLL_01.LowAlarm;

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Multiplexer (MUX)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

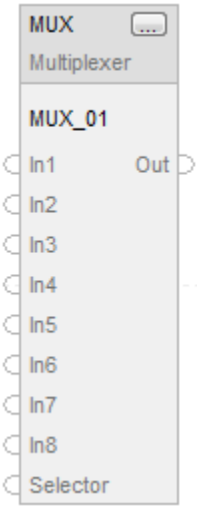
The MUX instruction selects one of eight inputs based on the selector input.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram.

Function Block



Structured Text

This instruction is not available in ladder diagram.

Operands

Function Block

Operand	Type	Format	Description
Block tag	MULTIPLEXER	Structure	MUX structure

MUX Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
In1	REAL	The first analog signal input to the instruction. Valid = any float Default = 0.0
In2	REAL	The second analog signal input to the instruction. Valid = any float Default = 0.0
In3	REAL	The third analog signal input to the instruction. Valid = any float Default = 0.0
In4	REAL	The fourth analog signal input to the instruction. Valid = any float Default = 0.0
In5	REAL	The fifth analog signal input to the instruction. Valid = any float Default = 0.0
In6	REAL	The sixth analog signal input to the instruction. Valid = any float Default = 0.0
In7	REAL	The seventh analog signal input to the instruction. Valid = any float Default = 0.0
In8	REAL	The eighth analog signal input to the instruction. Valid = any float Default = 0.0

Input Parameter	Data Type	Description
Selector	DINT	The selector input to the instruction. This input determines which of the inputs (1-8) is moved into Out. If this value is invalid (which includes 0), the instruction sets the appropriate bit in Status and holds Out at its current value. Valid = 1 to 8 Default = 0

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if the instruction is enabled. Cleared on overflow.
Out	REAL	The selected output of the algorithm. Math status flags are set for this output.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
SelectorInv (Status.1)	BOOL	Invalid Selector value.

Description

Based on the Selector value, the MUX instruction sets Out equal to one of eight inputs.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

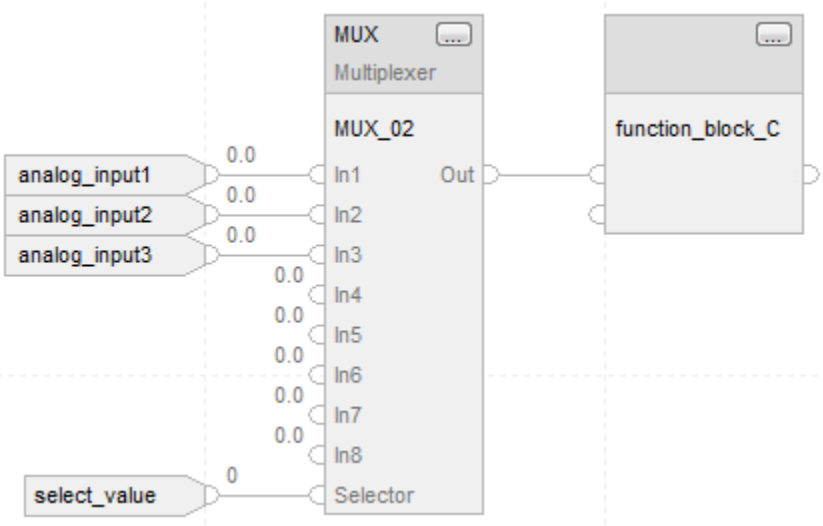
Function Block

Condition	Action

Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	Set internal value of Out to zero.
Postscan	EnableIn and EnableOut bits are cleared to false.

Example

Function Block



This MUX instruction selects In1, In2, or In3, In4, In5, In6, In7, or In8 based on the Selector. The instruction sets Out = In_n, which becomes an input parameter for function_block_C. For example, if select_value = 2, the instruction sets Out = analog_input2.

See also

Common Attributes on page 1083

Rate Limiter (RLIM)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

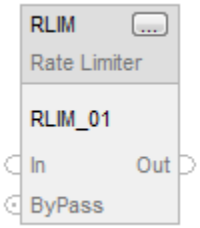
The RLIM instruction limits the amount of change of a signal over time.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

RLIM(RLIM_tag);

Operands

Function Block

Operand	Type	Format	Description
RLIM tag	RATE_LIMITER	structure	RLIM structure

RATE_LIMITER Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0
IncRate	REAL	Maximum output increment rate in per-second units. If invalid, the instruction sets IncRate = 0.0 and sets the appropriate bit in Status. Valid = any float ≥ 0.0 Default = 0.0

Input Parameter	Data Type	Description
DecRate	REAL	Maximum output decrement rate in per-second units. If invalid, the instruction sets DecRate = 0.0 and sets the appropriate bit in Status. Valid = any float ≥ 0.0 Default = 0.0
ByPass	BOOL	Request to bypass the algorithm. When true, Out = In. Default is false.
TimingMode	DINT	Selects timing execution mode. 0 = Period mode 1 = oversample mode 2 = real time sampling mode Valid = 0 to 2 Default = 0
OversampleDT	REAL	Execution time for oversample mode. Valid = 0 to 4194.303 seconds Default = 0
RTSTime	DINT	Module update period for real time sampling mode Valid = 1 to 32,767ms Default = 1
RTSTimeStamp	DINT	Module time stamp value for real time sampling mode. Valid = 0 to 32,767ms Default = 0

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if Out overflows.
Out	REAL	The calculated output of the algorithm.
DeltaT	REAL	Elapsed time between updates. This is the elapsed time in seconds used by the control algorithm to calculate the process output.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
IncRateInv (Status.1)	BOOL	IncRate < 0. The instruction uses 0.
DecRateInv (Status.2)	BOOL	DecRate < 0. The instruction uses 0.
TimingModeInv (Status.27)	BOOL	Invalid TimingMode value. For more information about timing modes, see Function Block Attributes.

Output Parameter	Data Type	Description
RTSMissed (Status.28)	BOOL	Only used in real time sampling mode. Set to true when ABS(DeltaT – RTSTime) > 1 millisecond.
RTSTimeInv (Status.29)	BOOL	Invalid RTSTime value.
RTSTimeStampInv (Status.30)	BOOL	Invalid RTSTimeStamp value.
DeltaTInv (Status.31)	BOOL	Set to true in oversample mode if either DeltaT <= 0 or DeltaT > 4194.303.

Structured Text

Operand	Type	Format	Description
RLIM tag	RATE_LIMITER	structure	RLIM structure

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Description

The RLIM instruction provides separate increment and decrement rates in per-second units. The ByPass input lets you stop rate limiting and pass the signal directly to the output.

Condition	Action
ByPass is true	$Out_n = In_n$ $Out_{n-1} = In_n$
ByPass is false and DeltaT > 0	$Slope = \frac{In_n - Out_{n-1}}{DeltaT}$ If Slope ≤ –DecRate then YSlope = –DecRate If –DecRate ≤ Slope ≤ IncRate then YSlope = Slope If IncRate ≤ Slope then YSlope = IncRate $Out_n = Out_{n-1}$ + DeltaT x YSlope $Out_{n-1} = Out_n$ where DeltaT is in seconds

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true The instruction executes.
Instruction first run	N/A
Instruction first scan	Initialize Out with the value of In.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

Function Block

The RLIM instruction limits In by IncRate. If analog_input1 changes at a rate greater than the IncRate value, the instruction limits In. The instruction sets Out = rate limited value of In.

Structured Text

```
RLIM_01.In := analog_input1;
RLIM_01.IncRate := value;
RLIM(RLIM_01);

rate_limited := RLIM_01.Out;
```

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Select (SEL)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380,

CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

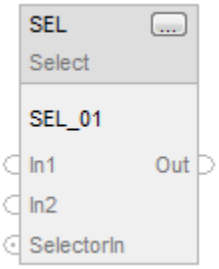
The SEL instruction uses a digital input to select one of two inputs.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

This instruction is not available in structured text.

Operands

Function Block

Operand	Type	Format	Description
SEL tag	SELECT	structure	SEL structure

SELECT Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.

In1	REAL	The first analog signal input to the instruction. Valid = any float Default = 0.0
In2	REAL	The second analog signal input to the instruction. Valid = any float Default = 0.0
SelectorIn	BOOL	The input that selects between In1 and In2. Default is cleared.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared on overflow.
Out	REAL	The calculated output of the algorithm.

Description

The SEL instruction operates as follows:

Condition	Action
SelectorIn is set	Out = In2
SelectorIn is cleared	Out = In1

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

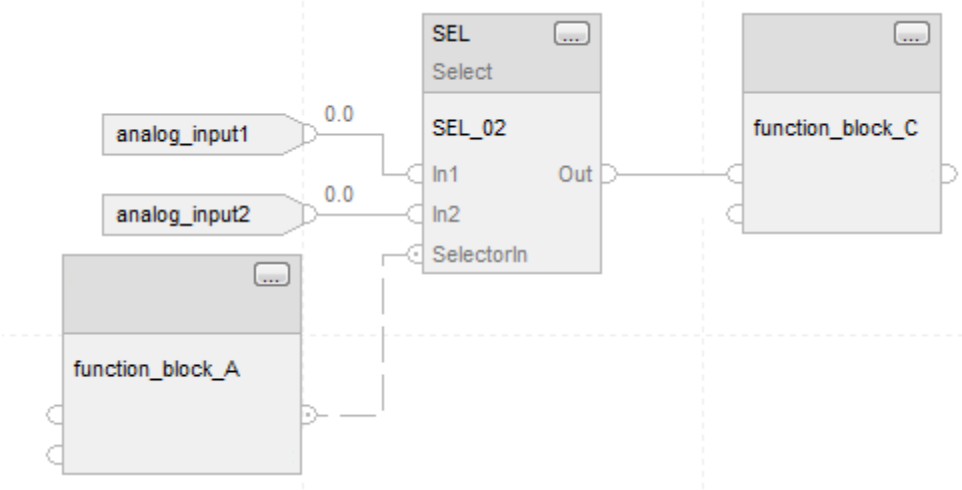
Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.

Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	Out n-1 is set to 0.
Postscan	N/A

Example

The SEL instruction selects In1 or In2 based on SelectorIn. If SelectorIn is set, the instruction sets Out = In2. If SelectorIn is cleared, the instruction sets Out = In1. Out becomes an input parameter for function_block_C.

Function Block



See also

Common Attributes on page 1083

Selected Negate (SNEG)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

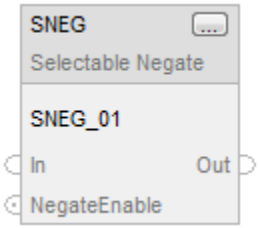
The SNEG instruction uses a digital input to select between the input value and the negative of the input value.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

SNEG(SNEG_tag);

Operands

Function Block

Operand	Type	Format	Description
SNEG tag	SELECTABLE_NEGATE	Structure	SNEG structure

SNEG Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true. Structured Text: No effect. The instruction executes.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0

Input Parameter	Data Type	Description
NegateEnable	BOOL	Negate enable. When NegateEnable is true, the instruction sets Out to the negative value of In. Default is true.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if Out overflows.

Structured Text

Operand	Type	Format	Description
SNEG tag	SELECTABLE_NEGATE	Structure	SNEG structure

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Description

The SNEG instruction operates as follows:

Condition	Action
NegateEnable is true	Out = - In
NegateEnable is false	Out = In

Affects Math Status Flags

No

Major/Minor faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	EnableIn and EnableOut bits are cleared to false.

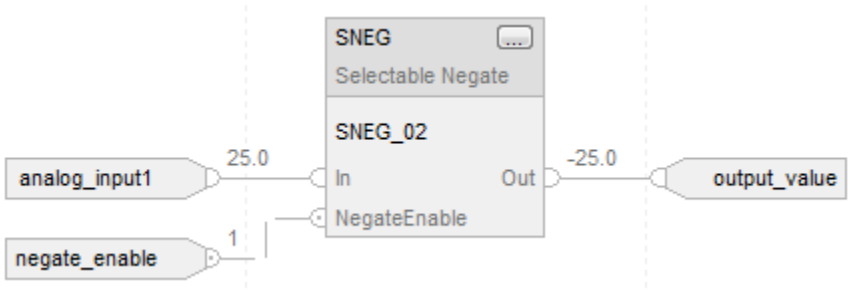
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

The negate_enable input determines whether to negate In or not. The instruction sets Out = In if NegateEnable is false. The instruction sets Out = -In if NegateEnable is true.

Function Block



Structured Text

```
SNEG_01.In := analog_input1;
SNEG_01.NegateEnable := negate_enable;
SNEG(SNEG_01);

output_value := SNEG_01.Out;
```

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Selected Summer (SSUM)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

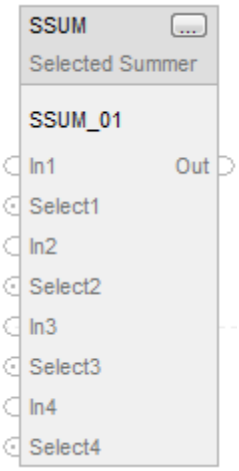
The SSUM instruction uses Boolean inputs to select real inputs to be algebraically summed.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

SSUM(SSUM_tag);

Operands

Function Block

Operand	Type	Format	Description
SSUM tag	SELECTED_SUMMER	Structure	SSUM structure

SELECTABLE_SUMMER Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
In1	REAL	The first input to be summed. Valid = any float Default = 0.0
Gain1	REAL	Gain for the first input. Valid = any float Default = 1.0
Select1	BOOL	Selector signal for the first input. Default is false.
In2	REAL	The second input to be summed. Valid = any float Default = 0.0
Gain2	REAL	Gain for the second input. Valid = any float Default = 1.0
Select2	BOOL	Selector signal for the second input. Default is false.
In3	REAL	The third input to be summed. Valid = any float Default = 0.0
Gain3	REAL	Gain for the third input. Valid = any float Default = 1.0
Select3	BOOL	Selector signal for the third input. Default is false.
In4	REAL	The fourth input to be summed. Valid = any float Default = 0.0
Gain4	REAL	Gain for the fourth input. Valid = any float Default = 1.0
Select4	BOOL	Selector signal for the fourth input. Default is false.
In5	REAL	The fifth input to be summed. Valid = any float Default = 0.0

Input Parameter	Data Type	Description
Gain5	REAL	Gain for the fifth input. Valid = any float Default = 1.0
Select5	BOOL	Selector signal for the fifth input. Default is false.
In6	REAL	The sixth input to be summed. Valid = any float Default = 0.0
Gain6	REAL	Gain for the sixth input. Valid = any float Default = 1.0
Select6	BOOL	Selector signal for the sixth input. Default is false.
In7	REAL	The seventh input to be summed. Valid = any float Default = 0.0
Gain7	REAL	Gain for the seventh input. Valid = any float Default = 1.0
Select7	BOOL	Selector signal for the seventh input. Default is false.
In8	REAL	The eighth input to be summed. Valid = any float Default = 0.0
Gain8	REAL	Gain for the eighth input. Valid = any float Default = 1.0
Select8	BOOL	Selector signal for the eighth input. Default is false.
Bias	REAL	Bias signal input. The instruction adds the Bias to the sum of the inputs. Valid = any float Default = 0.0

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if Out overflows.
Out	REAL	The calculated output of the algorithm.

Structured Text

Operand	Type	Format	Description
SSUM tag	SELECTED_SUMMER	Structure	SSUM structure

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Description

The SSUM instruction operates as follows:

Condition	Action
No In is selected	Out = Bias
One or more In are selected	For all n where Selectn is true Out = $\sum (In_n \times Gain_n) + Bias$

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	EnableIn and EnableOut bits are cleared to false.

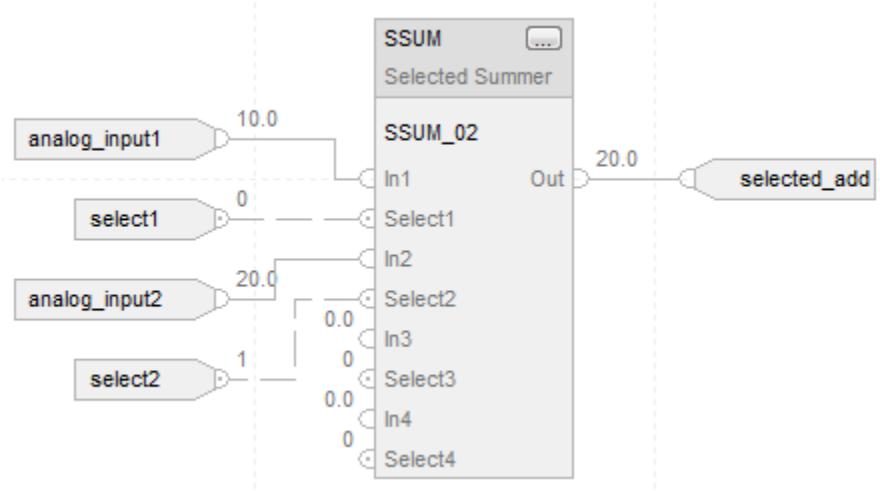
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

The values of select1 and select 2 determine whether to select analog_input1 and analog_input2, respectively. The instruction then adds the selected inputs and places the result in Out.

Function Block



Structured Text

```
SSUM_01.In1 := analog_input1;
SSUM_01.Select1 := select1;
SSUM_01.In2 := analog_input2;
SSUM_01.Select2 := select2;
SSUM(SSUM_01);

selected_add := SSUM_01.Out;
```

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Statistical Instructions

Statistical Instructions

The Statistical instructions include these instructions:

Available Instructions

Ladder Diagram

Not available

Function Block and Structured Text

MAVE	MAXC	MINC	MSTD

If you want to	Use this instruction
Calculate a time average value.	MAVE
Find the maximum signal in time.	MAXC
Find the minimum signal in time.	MINC
Calculate a moving standard deviation.	MSTD

See also

Filter Instructions on page 875

Logical and Move Instructions on page 955

Drives Instructions on page 823

Select/Limit Instructions on page 903

Process Control Instructions on page 17

Moving Average (MAVE)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

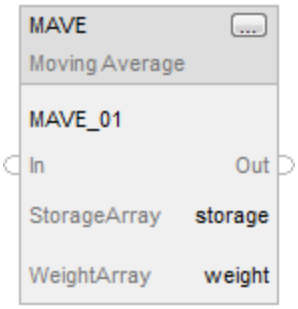
The MAVE instruction calculates a time average value for the In signal. This instruction optionally supports user-specified weights.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

MAVE(MAVE_tag,storage,weight);

Operands

Function Block

Operand	Type	Format	Description
MAVE tag	MOVING_AVERAGE	structure	MAVE structure
storage	REAL	array	Holds the moving average samples. This array must be at least as large as NumberOfSamples.

weight	REAL	array	(optional) Used for weighted averages. This array must be at least as large as NumberOfSamples. Element [0] is used for the newest sample; element [n] is used for the oldest sample.

MOVING_AVERAGE Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0
InFault	BOOL	Bad health indicator for the input. If In is read from an analog input, then InFault is normally controlled by fault status on the analog input. When set, InFault indicates the input signal has an error, the instruction sets the appropriate bit in Status, and the instruction holds Out at its current value. When InFault transitions from set to cleared, the instruction initializes the averaging algorithm and continues executing. Default is cleared.
Initialize	BOOL	Initialize input to the instruction. When set, the instruction holds Out = In, except when InFault is set, in which case, the instruction holds Out at its current value. When Initialize transitions from set to cleared, the instruction initializes the averaging algorithm and continues executing. Default is cleared.
SampleEnable	BOOL	Enable for taking a sample of In. When set, the instruction enters the value of In into the storage array and calculates a new Out value. When SampleEnable is cleared and Initialize is cleared, the instruction holds Out at its current value. Default is set.
NumberOfSamples	DINT	The number of samples to be used in the calculation. If this value is invalid, the instruction sets the appropriate bit in Status and holds Out at its current value. When NumberOfSamples becomes valid again, the instruction initializes the averaging algorithm and continues executing. Valid = 1 to (minimum size of StorageArray or WeightArray, if used) Default = 1

UseWeights	BOOL	Averaging scheme input to the instruction. When set, the instruction uses the weighted method to calculate the Out. When cleared, the instruction uses the uniform method to calculate Out. Default is cleared.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if Out overflows.
Out	REAL	The calculated output of the algorithm.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
InFaulted (Status.1)	BOOL	In health is bad (InFault is set).
NumberOfSamplInv (Status.2)	BOOL	NumberOfSamples invalid or not compatible with array size.

Structured Text

Operand	Type	Format	Description
MAVE tag	MOVING_AVERAGE	structure	MAVE structure
storage	REAL	array	Holds the moving average samples. This array must be at least as large as NumberOfSamples.
weight	REAL	array	(optional) Used for weighted averages. This array must be at least as large as NumberOfSamples. Element [0] is used for the newest sample; element [n] is used for the oldest sample.

MOVING_AVERAGE Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0

Input Parameter	Data Type	Description
InFault	BOOL	Bad health indicator for the input. If In is read from an analog input, then InFault is normally controlled by fault status on the analog input. When set, InFault indicates the input signal has an error, the instruction sets the appropriate bit in Status, and the instruction holds Out at its current value. When InFault transitions from set to cleared, the instruction initializes the averaging algorithm and continues executing. Default is cleared.
Initialize	BOOL	Initialize input to the instruction. When set, the instruction holds Out = In, except when InFault is set, in which case, the instruction holds Out at its current value. When Initialize transitions from set to cleared, the instruction initializes the averaging algorithm and continues executing. Default is cleared.
SampleEnable	BOOL	Enable for taking a sample of In. When set, the instruction enters the value of In into the storage array and calculates a new Out value. When SampleEnable is cleared and Initialize is cleared, the instruction holds Out at its current value. Default is set.
NumberOfSamples	DINT	The number of samples to be used in the calculation. If this value is invalid, the instruction sets the appropriate bit in Status and holds Out at its current value. When NumberOfSamples becomes valid again, the instruction initializes the averaging algorithm and continues executing. Valid = 1 to (minimum size of StorageArray or WeightArray, if used) Default = 1
UseWeights	BOOL	Averaging scheme input to the instruction. When set, the instruction uses the weighted method to calculate the Out. When cleared, the instruction uses the uniform method to calculate Out. Default is cleared.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if Out overflows.
Out	REAL	The calculated output of the algorithm.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
InFaulted (Status.1)	BOOL	In health is bad (InFault is set).
NumberOfSampInv (Status.2)	BOOL	NumberOfSamples invalid or not compatible with array size.

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Description

The MAVE instruction calculates a weighted or non-weighted moving average of the input signal. The NumberOfSamples specifies the length of the moving average span. At every scan of the block when Sample Enable is set, the instruction moves the value of In into the storage array and discards the oldest value. Each Inn has a user-configured Weightn, which is used if UseWeights is set.

Condition	Action
Weighted averaging method UseWeights is set.	$Out = \sum_{n=1}^{NumberOfSamples} Weight_n \times In_n$
Uniform averaging method UseWeights is cleared.	$Out = \frac{\sum_{n=1}^{NumberOfSamples} In_n}{NumberOfSamples}$

The instruction will not place an invalid In value (NAN or ± INF) into the storage array. When In is invalid, the instruction sets Out = In and logs an overflow minor fault, if this reporting is enabled. When In becomes valid, the instruction initializes the averaging algorithm and continues executing.

You can make runtime changes to the NumberOfSamples parameter. If you increase the number, the instruction incrementally averages new data from the current sample size to the new sample size. If you decrease the number, the instruction recalculates the average from the beginning of the sample array to the new NumberOfSamples value.

Initializing the Averaging Algorithm

Certain conditions, such as instruction first scan and instruction first run, require the instruction to initialize the moving average algorithm. When this occurs, the instruction considers the sample StorageArray empty and incrementally averages samples from 1 to the NumberOfSamples value. For example:

NumberOfSamples = 3, UseWeights is set

Scan 1: Out = In_n*$Weight_1$

Scan 2: Out = (In_n*$Weight_1$)+(In_{n-1}*$Weight_2$)

Scan 3: Out = (In_n*$Weight_1$)+(In_{n-1}*$Weight_2$) +(In_{n-2}*$Weight_3$)

NumberOfSamples = 3, UseWeights is cleared

Scan 1: Out = In_n/1

Scan 2: Out = (In_n+In_{n-1})/2

Scan 3: Out = (In_n+In_{n-1}+In_{n-2})/NumberOfSamples

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	Request re-initialization of Storage array the next time the instruction executes.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	Initialize Out to zero.
Postscan	EnableIn and EnableOut bits are cleared to false.

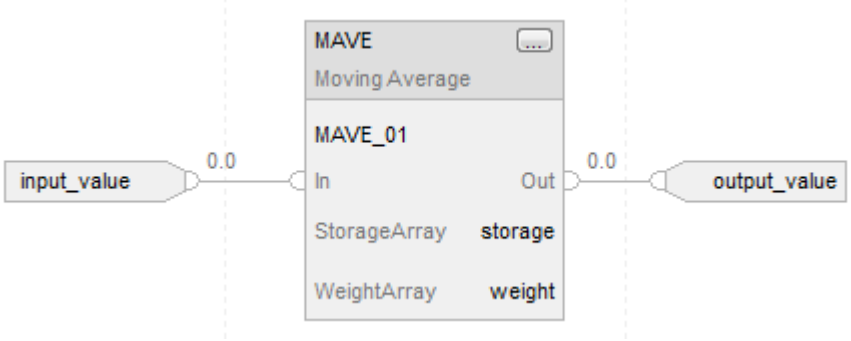
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

Each scan, the instruction places input_value in array storage. The instruction calculates the average of the values in array storage, optionally using the weight values in array weight, and places the result in Out.

Function Block



Structured Text

```
MAVE_01.In := input_value;
MAVE(MAVE_01,storage,weight);
output_value := MAVE_01.Out;
```

See also

Function Block Attributes on page 1043

Common Attributes on page 1083

Structured Text Syntax on page 1057

Maximum Capture (MAXC)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

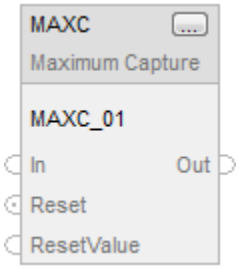
The MAXC instruction retains the maximum value of the input over time and allows the user to re-establish a maximum as needed.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

MAXC(MAXC_tag);

Operands

Function Block

Operand	Type	Format	Description
MAXC tag	MAXIMUM_CAPTURE	Structure	MAXC structure

MAXIMUM_CAPTURE Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0
Reset	BOOL	Request to reset control algorithm. The instruction sets Out = ResetValue as long as Reset is set. Default is cleared.
ResetValue	REAL	The reset value for the instruction. The instruction sets Out = ResetValue as long as Reset is set. Valid = any float Default = 0.0

Output Parameter	Data Type	Description
EnableOut	BOOL	Enable output.
Out	REAL	The calculated output of the algorithm.

Structured Text

Operand	Type	Format	Description
MAXC tag	MAXIMUM_CAPTURE	Structure	MAXC structure

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

Description

The MAXC instruction executes this algorithm:

Condition	Action
Reset is set	LastMaximum = Reset value Out = LastMaximum
Reset is cleared	If In > LastMaximum then update LastMaximum. Out = LastMaximum.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	Set request to initialize the Maximum value with the current input.

Postscan	EnableIn and EnableOut bits are cleared to false.

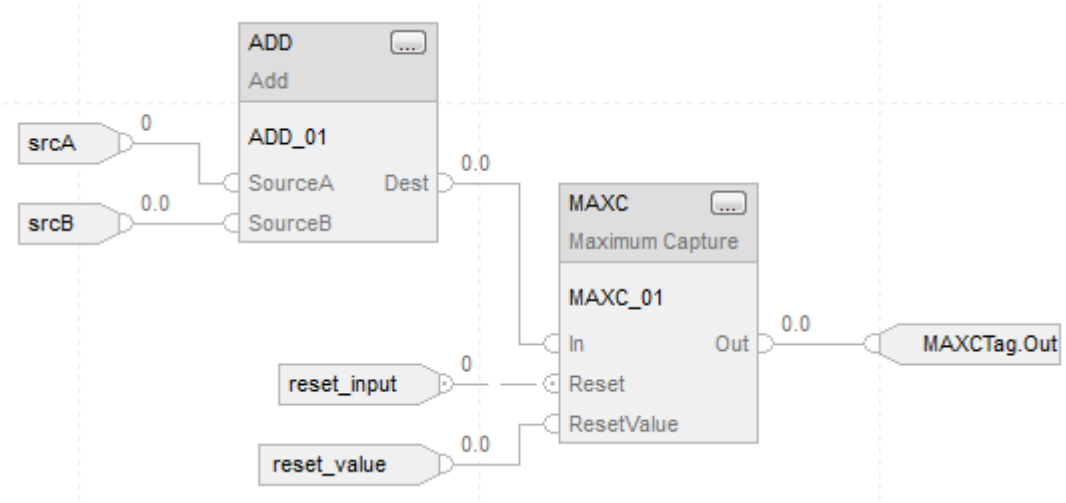
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Examples

If Reset is set, the instruction sets Out=ResetValue. If Reset is cleared, the instruction sets Out=In when In> LastMaximum. Otherwise, the instruction sets Out= LastMaximum.

Function Block



Structured Text

MAXCTag.In := input_value;

MAXCTag.Reset := reset_input;

MAXCTag.ResetValue := reset_value;

MAXC(MAXCTag);

result := MAXCTag.Out;

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Minimum Capture (MINC)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

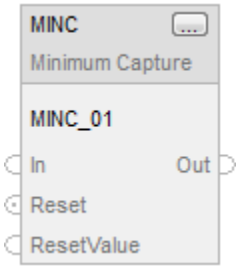
The MINC instruction retains the minimum value of the input over time and allows the user to re-establish a minimum as needed.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram.

Function Block



Structured Text

MINC(MINC_tag);

Operands

Function Block

Operand	Type	Format	Description
MINC tag	MINIMUM_CAPTURE	structure	MINC structure

MINIMUM_CAPTURE Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0
Reset	BOOL	Request to reset control algorithm. The instruction sets Out = ResetValue as long as Reset is set. Default is cleared.
ResetValue	REAL	The reset value for the instruction. The instruction sets Out = ResetValue as long as Reset is set. Valid = any float Default = 0.0

Output Parameter	Data Type	Description
EnableOut	BOOL	Enable output.
Out	REAL	The calculated output of the algorithm.

Structured Text

Operand	Type	Format	Description
MINC tag	MINIMUM _CAPTURE	structure	MINC structure

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

Description

The MINC instruction executes this algorithm:

Condition	Action
Reset is set	LastMinimum = ResetValue Out = ResetValue
Reset is cleared	If In < LastMinimum then update LastMinimum. Out = LastMinimum.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	Set request to initialize the Maximum value with the current input.
Postscan	EnableIn and EnableOut bits are cleared to false.

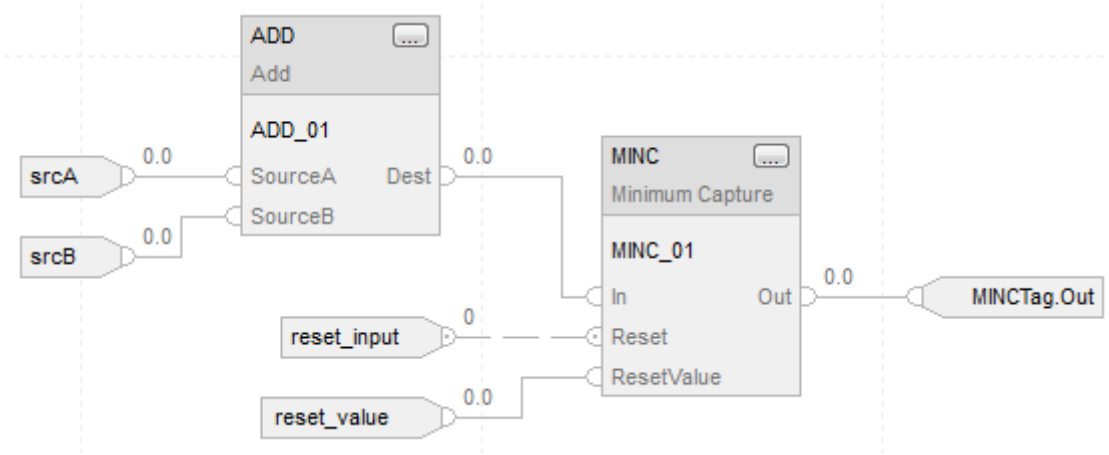
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Examples

If Reset is set, the instruction set Out=ResetValue. If Reset is cleared, the instruction set Out=In when In < LastMinimum. Otherwise, the instruction sets Out= LastMinimum.

Function Block



Structured Text

MINCTag.In := input_value;

MINCTag.Reset := reset_input;

MINCTag.ResetValue := reset_value;

MINC(MINCTag);

result := MINCTag.Out;

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Moving Standard Deviation (MSTD)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

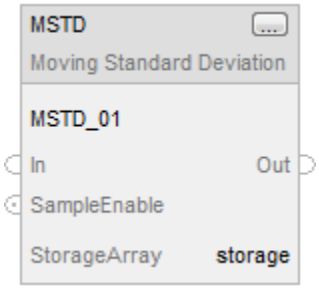
The MSTD instruction calculates a moving standard deviation and average for the In signal.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

MSTD(MSTD_tag, StorageArray);

Operands

Function Block

Operand	Type	Format	Description
block tag	MOVING_STD_DEV	structure	MSTD structure
StorageArray	REAL	array	Holds the In samples. This array must be at least as large as NumberOfSamples.

MOVING_STD_DEV Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
In	REAL	The analog signal input to the instruction. Valid = any float Default = 0.0

Input Parameter	Data Type	Description
InFault	BOOL	Bad health indicator for the input. If In is read from an analog input, then InFault is normally controlled by fault status on the analog input. When set, InFault indicates the input signal has an error, the instruction sets the appropriate bit in Status, and the instruction holds Out and Average at their current values. When InFault transitions from set to cleared, the instruction initializes the averaging algorithm and continues executing. Default is cleared.
Initialize	BOOL	Initialize input to the instruction. When set, the instruction sets Out = 0.0 and Average = In, except when InFault is set, in which case, the instruction holds both Out and Average at their current values. When Initialize transitions from set to cleared, the instruction initializes the standard deviation algorithm and continues executing. Default is cleared.
SampleEnable	BOOL	Enable for taking a sample of In. When set, the instruction enters the value of In into the storage array and calculates a new Out and Average value. When SampleEnable is cleared and Initialize is cleared, the instruction holds Out and Average at their current values. Default is cleared.
NumberOfSamples	DINT	The number of samples to be used in the calculation. If this value is invalid, the instruction sets the appropriate bit in Status and the instruction holds Out and Average at their current values. When NumberOfSamples becomes valid again, the instruction initializes the standard deviation algorithm and continues executing. Valid = 1 to size of the storage array Default = 1

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled. Cleared to false if Out overflows
Out	REAL	The calculated output of the algorithm.
Average	REAL	The calculated average of the algorithm.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
InFaulted (Status.1)	BOOL	In health is bad. InFault is set.

NumberOfSampInv (Status.2)	BOOL	NumberOfSamples invalid or not compatible with array size.

Structured Text

Operand	Type	Format	Description
block tag	MOVING_STD_DEV	structure	MSTD structure
StorageArray	REAL	array	Holds the In samples. This array must be at least as large as NumberOfSamples.

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Description

The MSTD instruction supports any input queue length. Each scan, if SampleEnable is set, the instruction enters the value of In into a storage array. When the storage array is full, each new value of In causes the oldest entry to be deleted.

The MSTD instructions uses these equations for the outputs:

Condition	Action
Average	$Average = \frac{\sum_{n=1}^{NumberOfSamples} In_n}{NumberOfSamples}$
Out	$Out = \sqrt{\frac{\sum_{n=1}^{NumberOfSamples} (In_n - Average)^2}{NumberOfSamples}}$

The instruction will not place an invalid In value (NAN or ± INF) into the storage array. When In is invalid, the instruction sets Out = In, sets Average = In, and logs an overflow minor fault, if this reporting is enabled. When In becomes valid, the instruction initializes the standard deviation algorithm and continues executing.

You can make runtime changes to the NumberOfSamples parameter. If you increase the number, the instruction incrementally processes new data from the current sample size to the new sample size. If you decrease the number, the instruction re-calculates the standard deviation from the beginning of the sample array to the new NumberOfSamples value.

Initializing the Standard Deviation Algorithm

Certain conditions, such as instruction first scan and instruction first run, require the instruction to initialize the standard deviation algorithm. When this occurs, the instruction considers the StorageArray empty and

incrementally processes samples from 1 to the NumberOfSamples value. For example:

NumberOfSamples = 3

Scan 1: Average = $In_n/1$

Out = Square root $(((In_n\text{-Average})^2)/1)$

Scan 2: Average = $(In_n+In_{n-1})/2$

Out = Square root $(((In_n\text{-Average})^2+(In_{n-1}\text{-Average})^2)/2)$

Scan 3: Average = $(In_n+In_{n-1}+In_{n-2})$/NumberOfSamples

Out = Square root $(((In_n\text{-Average})^2+(In_{n-1}\text{-Average})^2+(In_{n-2}\text{-Average})^2)$/NumberOfSamples)

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	Initialize the previous Output and Average.
Instruction first scan	Initialize Out to zero. Initialize Average to Input Initialize the instruction algorithm.
Postscan	EnableIn and EnableOut bits are cleared to false.

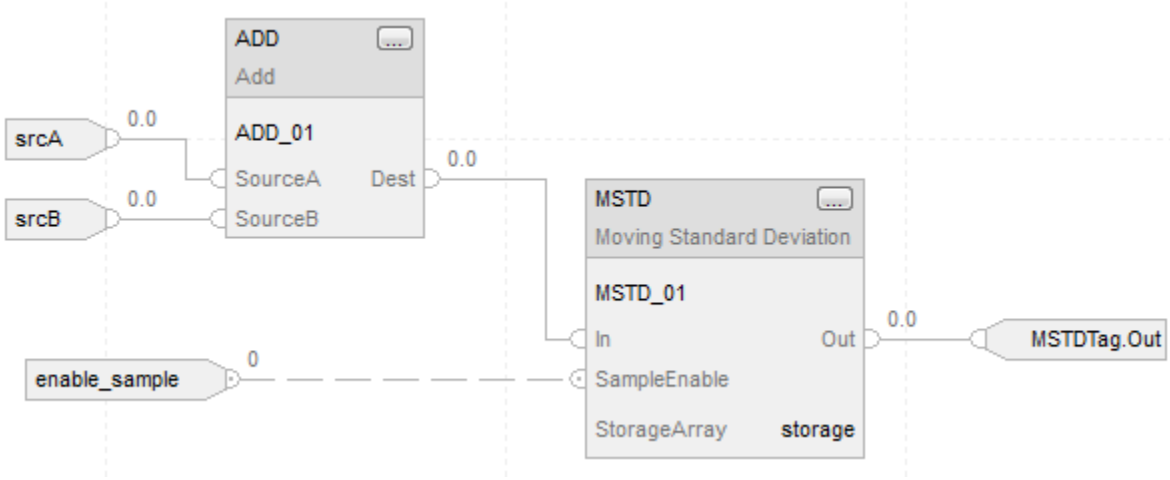
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

Each scan that SampleEnable is set, the instruction places the value of In into array storage, calculates the standard deviation of the values in array storage, and places the result in Out. Out becomes an input parameter for function_block_C.

Function Block



Structured Text

```
MSTD_01.In := input_value;
MSTD_01.SampleEnable := enable_sample;
MSTD(MSTD_01,storage);
deviation := MSTD_01.Out;
```

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Logical and Move

Logical and Move Instructions

These instructions are used to perform logical operations on and move output data.

Available Instructions

Ladder Diagram and Structured Text

DFF	JKFF	RESD	SETD

Function Block

Not available

Structured Text

DFF	JKFF	RESD	SETD

See also

Filter Instructions on page 875

Drives Instructions on page 823

Process Control Instructions on page 17

Select/Limit Instructions on page 903

Statistical Instructions on page 935

D Flip-Flop (DFF)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

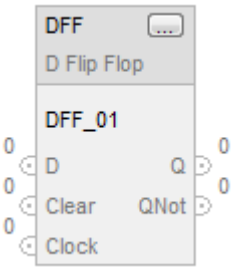
The DFF instruction sets the Q output to the state of the D input on a cleared to set transition of the Clock input. The QNot output is set to the opposite state of the Q output.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

DFF(DFF_tag);

Operands

Function Block

Operand	Type	Format	Description
DFF tag	FLIP_FLOP_D	structure	DFF structure

FLIP_FLOP_D Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
D	BOOL	The input to the instruction. Default is cleared.

Input Parameter	Data Type	Description
Clear	BOOL	Clear input to the instruction. If set, the instruction clears Q and sets QNot.
Clock	BOOL	Clock input to the instruction. Default is cleared.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled.
Q	BOOL	The output of the instruction.
QNot	BOOL	The complement of the Q output.

Structured Text

Operand	Type	Format	Description
DFF tag	FLIP_FLOP_D	structure	DFF structure

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

Description

When Clear is set, the instructions clears Q and sets QNot. Otherwise, if Clock is set and Clockn-1 is cleared, the instruction sets Q=D and sets QNot = NOT (D).

The instruction sets Clockn-1 = Clock state every scan.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	Clockn-1 is set to 1. Qn-1 is cleared to 0.
Instruction first scan	Previous input Clock state is set True. Previous output Q state is set False.
Postscan	EnableIn and EnableOut bits are cleared to false.

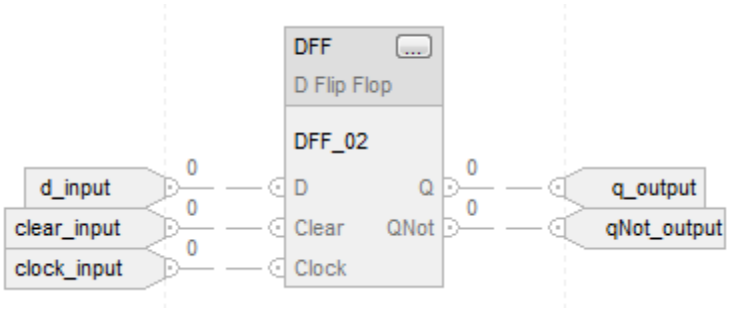
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

When Clock goes from cleared to set, the DFF instruction sets Q = D. When Clear is set, Q is cleared. The DFF instruction sets QNot to the opposite state of Q.

Function Block



Structured Text

```
DFF_03.D := d_input;
DFF_03.Clear := clear_input;
```

```
DFF_03.Clock := clock_input;
DFF(DFF_03);
q_output := DFF_03.Q;
qNot_output := DFF_03.QNot;
```

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

JK Flip-Flop (JKFF)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

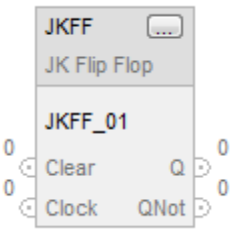
The JKFF instruction complements the Q and QNot outputs when the Clock input transitions from cleared to set.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

JKFF(JKFF_tag);

Operands

Function Block

Operand	Type	Format	Description
JKFF tag	FLIP_FLOP_JK	Structure	JKFF structure

FLIP_FLOP_JK Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
Clear	BOOL	Clear input to the instruction. If set, the instruction clears Q and sets QNot.
Clock	BOOL	Clock input to the instruction. Default is cleared.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled.
Q	BOOL	The output of the instruction.
QNot	BOOL	The complement of the Q output.

Structured Text

Operand	Type	Format	Description
JKFF tag	FLIP_FLOP_JK	Structure	JKFF structure

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Description

When Clear is set, the instructions clears Q and sets QNot. Otherwise, if Clock is set and Clockn-1 is cleared, the instruction toggles Q and QNot.

The instruction sets Clockn-1 = Clock state every scan.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	Clockn-1 is set to 1. Qn-1 is cleared to 0.
Instruction first scan	Previous input Clock states is set to True. Previous output Q state is False.
Postscan	EnableIn and EnableOut bits are cleared to false.

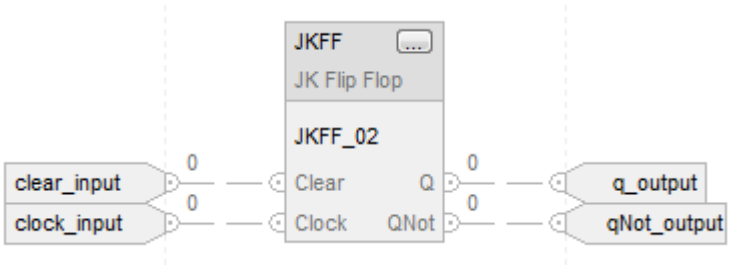
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Examples

When Clock goes from cleared to set, the JKFF instruction toggles Q. If Clear is set, Q is always cleared. The JKFF instruction sets QNot to the opposite state of Q.

Function Block



Structured Text

```
JKFF_01.Clear := clear_input;
JKFF_01.Clock := clock_input;
JKFF(JKFF_01);
q_output := JKFF_01.Q;
qNot_output := JKFF_01.QNot;
```

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Reset Dominant (RESD)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

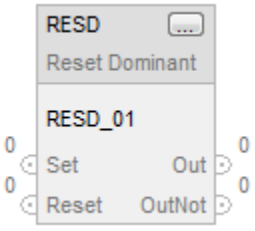
The RESD instruction uses Set and Reset inputs to control latched outputs. The Reset input has precedence over the Set input.

Available Languages

Ladder Diagram

This instruction is not available for ladder diagram.

Function Block



Structured Text

```
RESD(RESD_tag);
```

Operands

Function Block

Operand	Type	Format	Description
RESD tag	DOMINANT_RESET	structure	RESD structure

DOMINANT_RESET Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
Set	BOOL	Set input to the instruction. Default is cleared.
Clear	BOOL	Reset input to the instruction. Default is cleared.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled.
Out	BOOL	The output of the instruction.
OutNot	BOOL	The inverted output of the instruction.

Structured text

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

Description

The Reset Dominant instruction uses the Set and Reset input parameters to control latched output parameters Out and OutNot. The Reset input has precedence over the Set input.

Out will be latched true whenever the Set input parameter is set true. Setting the Reset parameter to true will override the current state of Out, setting Out to false.

When Reset returns to false, Out will be latched to the current state of the Set input parameter. OutNot will be set to the opposite state of Out.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for fault related attributes.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	Previous Out is set to False.
Instruction first scan	N/A
Postscan	EnableIn and EnableOut bits are cleared to false.

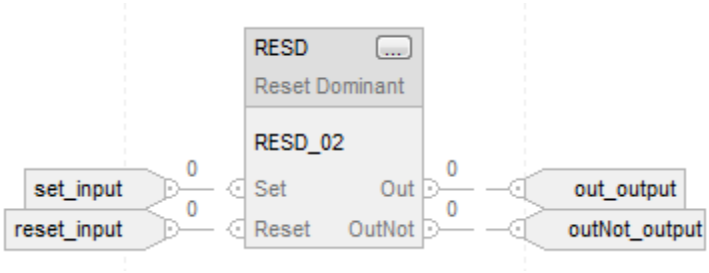
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

When Set is true and Reset is false, Out is set true. When Reset is true, Out is cleared. The Reset input has precedence over the Set input. The RESD instruction sets OutNot to the opposite state of Out.

Function Block



Structured Text

```
RESD_01.Set := set_input;
RESD_01.Reset := reset_input;
RESD(RESD_01);
out_output := RESD_01.Out;
outNot_output := RESD_01.OutNot;
```

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Set Dominant (SETD)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

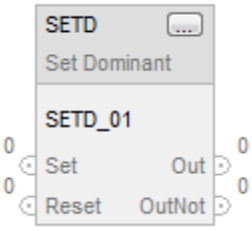
The SETD instruction uses Set and Reset inputs to control latched outputs. The Set input has precedence over the Reset input.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram logic.

Function Block



Structured Text

SETD(SETD_tag);

Operands

Function Block

Operand	Type	Format	Description
SETD tag	DOMINANT_SET	structure	SETD structure

Structured text

Operand	Type	Format	Description
SETD tag	DOMINANT_SET	structure	SETD structure

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

DOMINANT_SET Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
Set	BOOL	Set input to the instruction. Default is cleared.
Clear	BOOL	Reset input to the instruction. Default is cleared.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled.
Out	BOOL	The output of the instruction.

OutNot	BOOL	The inverted output of the instruction.

Description

The Set Dominant instruction uses the Set and Reset input parameters to control latched output parameters Out and OutNot. The Set input has precedence over the Reset input.

Out will be latched true whenever the Set input parameter is set true. Setting the Reset parameter to true will set Out to false only if the Set input is false. OutNot will be set to the opposite state of Out.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	Previous Out is set to True.
Instruction first scan	N/A
Postscan	EnableIn and EnableOut bits are cleared to false

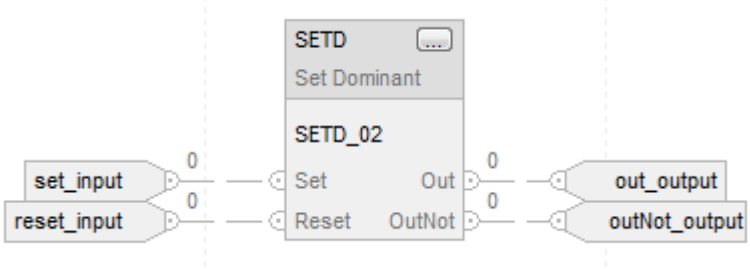
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

When Set is true, Out is set true. When Set is false and Reset is true, Out is cleared. The Set input has precedence over the Reset input. The SETD instruction sets OutNot to the opposite state of Out.

Function Block



Structured Text

```
SETD_01.Set := set_input;
SETD_01.Reset := reset_input;
SETD(SETD_01);
out_output := SETD_01.Out;
outNot_output := SETD_01.OutNot;
```

See also

Common Attributes on page 1083

Structured Text Syntax on page 1057

Equipment Phase Instructions

Equipment Phase Instructions

The Equipment Phase instructions include:

Available Instructions

Ladder Diagram and Structured Text

PSC	PCMD	POVR	PFL	PCLF	PXRQ	PRNP	PPD	PATT	PDET

Function Block

These instructions are not available in function block.

If want to:	Use this instruction:
Signal an equipment phase that the state routine is complete to indicate that it should go to the next state	PSC
Change the state or substate of an equipment phase	PCMD
Give a Hold, Stop, or Abort command to an equipment phase, regardless of ownership	POVR
Signal a failure for an equipment phase	PFL
Clear the failure code of an equipment phase	PCLF
Initiate communication with FactoryTalk Batch software.	PXRQ
Clear the NewInputParameters bit of an equipment phase	PRNP
Set up breakpoints within the logic of an equipment phase	PPD
Take ownership of an equipment phase to either: • Prevent another program or FactoryTalk Batch software from commanding an equipment phase • Make sure another program or FactoryTalk Batch software does not already own an equipment phase	PATT
Relinquish ownership of an equipment phase	PDET

See also

Equipment Sequence Diagram instructions on page 1031

Attach to Equipment Phase (PATT)

This instruction applies to the CompactLogix 5370 and CompactLogix 5380, ControlLogix 5570 and ControlLogix 5580, Compact GuardLogix 5370 and Compact GuardLogix 5380, and GuardLogix 5570 and GuardLogix 5580 controllers.

Use the PATT instruction to take ownership of an equipment phase to either:

- Prevent another program or FactoryTalk Batch software from commanding an equipment phase.
- Make sure another program or FactoryTalk Batch software does not already own an equipment phase.

The PATT instruction lets a program take ownership of an equipment phase.

- Ownership is optional. As long as an equipment phase has no owners, any sequencer (program in the controller, FactoryTalk Batch software) can command an equipment phase.
- FactoryTalk Batch software always takes ownership of an equipment phase.
- Once a sequencer owns an equipment phase, no other sequencer can command the equipment phase.

Available Languages

Ladder Diagram

Function Block

This instruction is not available in function block.

Structured Text

PATT(Phase_Name,Result);

Operands

Ladder Diagram

Operand	Type	Format	Description
Phase Name	PHASE	Name of the equipment phase	Equipment phase to own.
Result	DINT	immediate tag	To let the instruction return a code for its success or failure, enter a DINT tag in which to store the result code. Otherwise, enter 0.

Structured Text

The operands are the same as those for the Ladder Diagram PATT instruction.

Guidelines for using the PATT Instruction

Guideline	Details	
Consider ownership if have multiple sequencers that use a common equipment phase.	Ownership makes sure that a program can command all the equipment phases it needs and locks out any other sequencers.	
	If using:	**Then:**
	FactoryTalk Batch software to also run sequences within this controller	Before executing the sequence (process), take ownership of all the equipment phases that the sequence uses.
	Multiple programs to command the same equipment phase	
	None of the above	No need to own the equipment phases.
Remember that Logix Designer overrides the controller.	Regardless of whether a program or FactoryTalk Batch software owns an equipment phase, the option exists to use Logix Designer to override ownership and command the equipment phase to a different state.	
	This:	**Overrides this:**
	Logix Designer	Controller (internal sequencer), FactoryTalk Batch software (external sequencer)
	Controller (internal sequencer)	None
	FactoryTalk Batch software (external sequencer)	None

Guideline	Details
Use the Result operand to validate ownership.	Use the Result operand to get a code that shows the success or failure of the PATT instruction. To interpret the result code, see the *PATT Result Codes* section below.
Avoid or plan for a result code = 24582.	On each execution, the PATT instruction tries to take ownership of the equipment phase. Once a program owns an equipment phase, another execution of the PATT instruction produces a result code = 24582. When using a PATT instruction, either: • Limit its execution to a single scan to avoid the 24582 result code. • Include a result code = 24582 in conditions for ownership.
When the sequence is done, relinquish ownership.	To relinquish ownership, use a Detach from Equipment Phase (PDET) instruction.

PATT Result Codes

If assigning a tag to store the result of a PATT instruction, the instruction returns one of these codes when it executes:

Code (Dec)	Description
0	The command was successful.
24579	Logix Designer or HMI already owns equipment phase. The caller is attached to the equipment phase, but it is not the current commanding owner. • This program now also owns the equipment phase. • Since the Logix Designer or HMI is higher priority than a program, the program cannot command the equipment phase. • High priority HMI ownership is specific only to CompactLogix 5370 and ControlLogix 5570 controllers.
24582	The program already owns the equipment phase.
24593	One of these already owns the equipment phase: • External sequencer (FactoryTalk Batch software) • Another program in the controller
24594	The equipment phase is unscheduled, inhibited, or in a task that is inhibited.

Affects Math Status Flags

No

Major/Minor Faults

None. See *Index Through Arrays* for operand-related faults.

Execution

For Structured Text, EnableIn is always true during normal scan. Therefore, if the instruction is in the control path activated by the logic, it will execute. All Conditions below the thick solid line can only occur during Normal Scan mode.

Condition/State	Action Taken
Prescan	No action taken.
Postscan	No action taken.
EnableIn is false	No action taken.
EnableIn is true	The instruction executes as described above.

Example

Ladder Diagram

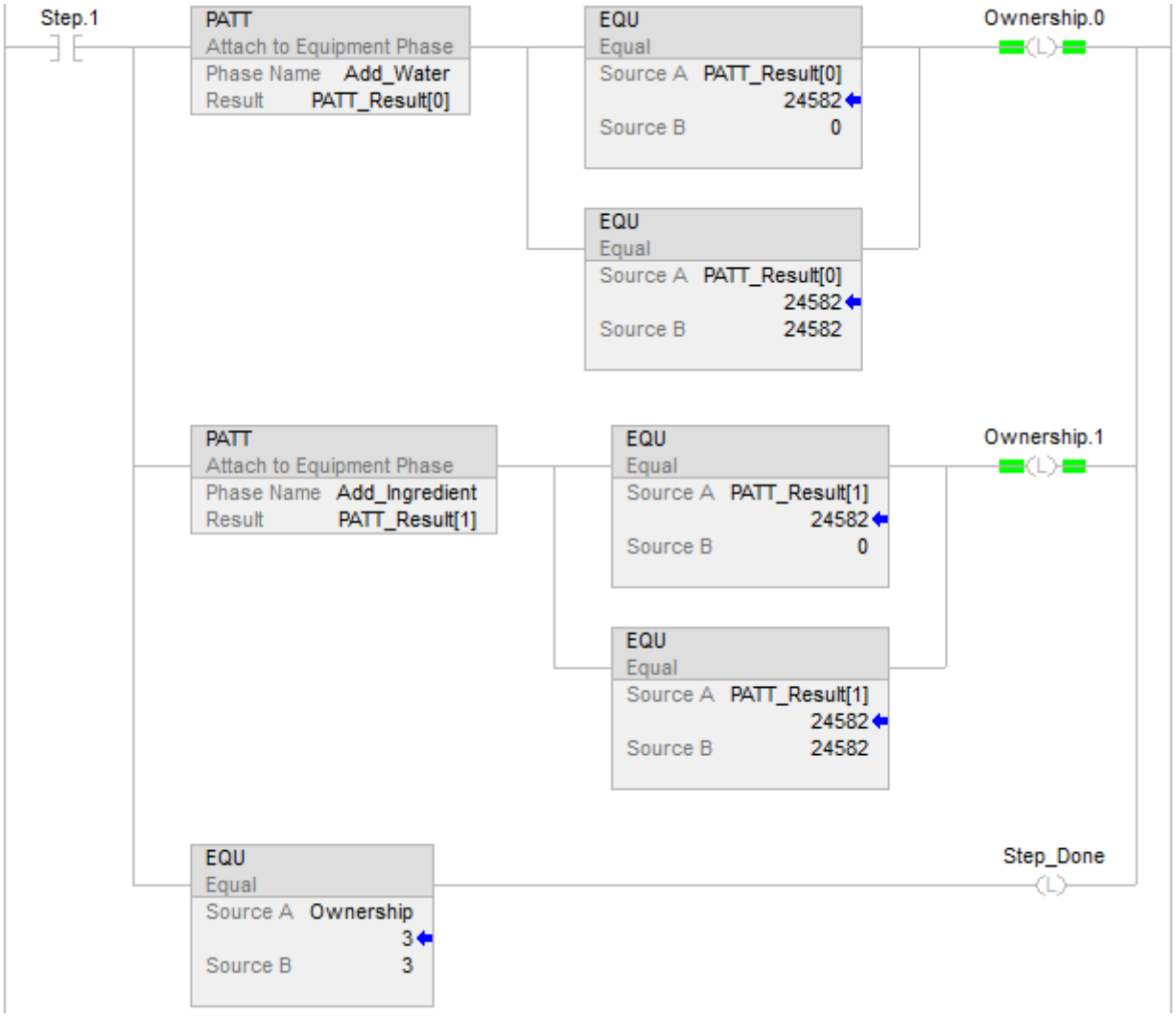
If *Step.1* = 1 (first step in the sequence) then

Each PATT instruction tries to take ownership of an equipment phase.

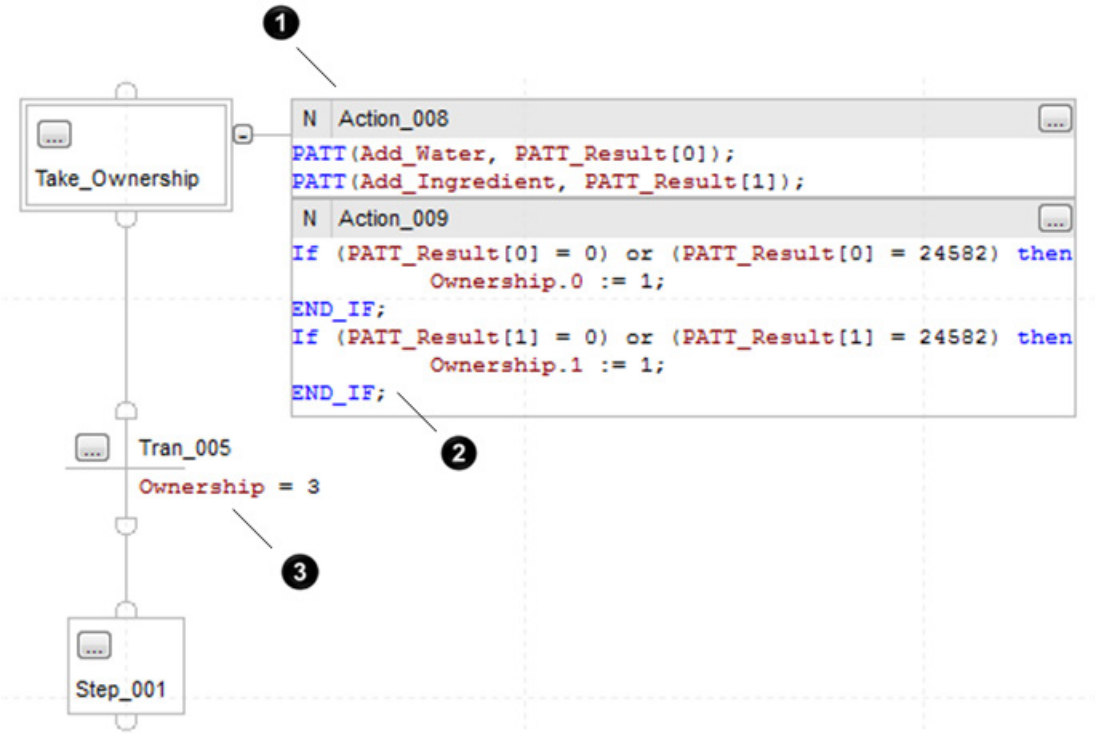
If the Result of a PATT instruction = 0 or 24582 (the program owns the equipment phase), then
A bit within the *Ownership* tag = 1. (In the *Ownership* tag, each equipment phase is assigned a bit.)

If *Ownership* = 3 (The program owns both equipment phases as shown by bits 0 and 1), then

Done = 1. (This signals the sequence to go to the next step.)



Structured Text



Number	Description
1	At the first step in the sequence, the Take_Ownership action tries to take ownership of two equipment phases that the sequence uses.
2	Action_009 checks that the program owns the equipment phases. If the Result of each PATT instruction = 0 or 24282 (the program owns the equipment phase), then a bit within the Ownership tag = 1. (In the Ownership tag, each equipment phase is assigned a bit.)
3	If the Ownership = 3 (The program owns both equipment phases as shown by bits 0 and 1.), then the SFC goes to the next step.

See also

Equipment Phase Instructions on page 969

Index Through Arrays on page 1094

Detach from Equipment Phase (PDET)

This instruction applies to the CompactLogix 5370 and CompactLogix 5380, ControlLogix 5570 and ControlLogix 5580, Compact GuardLogix 5370 and Compact GuardLogix 5380, and GuardLogix 5570 and GuardLogix 5580 controllers.

Use the PDET instruction to relinquish ownership of an equipment phase.

After a program executes a PDET instruction, the program *no longer* owns the equipment phase. This frees the equipment phase for ownership by another program or by FactoryTalk Batch software. Use the PDET instruction only if the program previously took ownership of an equipment phase via an Attach to Equipment Phase (PATT) instruction.

Available Languages

Ladder Diagram

Function Block

This instruction is not available in function block.

Structured Text

```
PDET(Phase_Name);
```

Operands

Ladder Diagram

Operand	Type	Format	Description
Phase Name	PHASE	Name of the equipment phase	Equipment phase no longer to own.

Structured Text

The operands are the same as those for the Ladder Diagram PDET instruction.

Affects Math Status Flags

No

Major/Minor Faults

None. See *Index Through Arrays* for operand-related faults.

Execution

For Structured Text, EnableIn is always true during normal scan. Therefore, if the instruction is in the control path activated by the logic, it will execute.

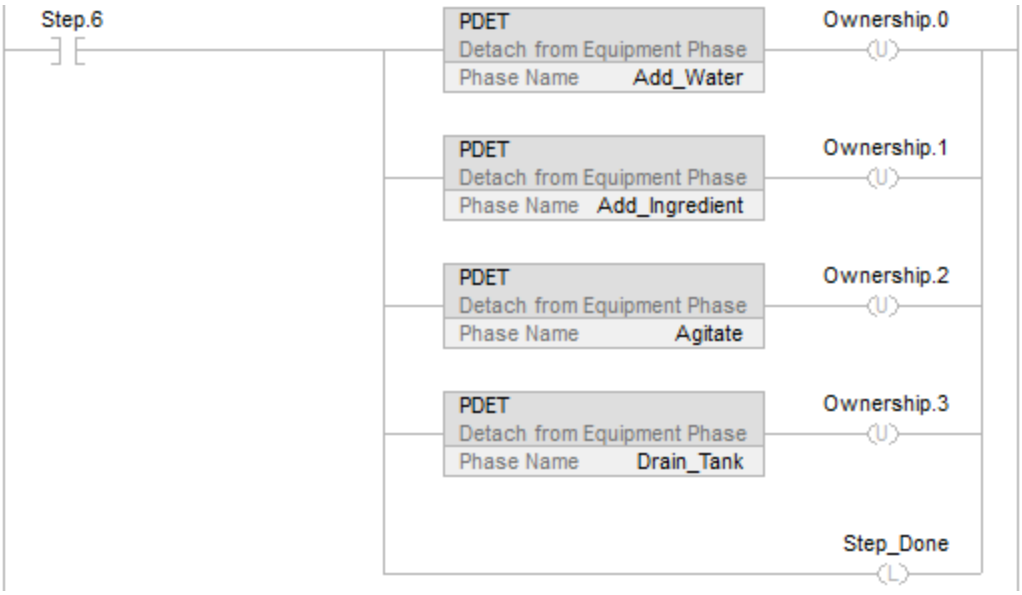
Condition/State	Action Taken
Prescan	No action taken.
Postscan	No action taken.
EnableIn is false	No action taken.
EnableIn is true	The instruction executes.

Example

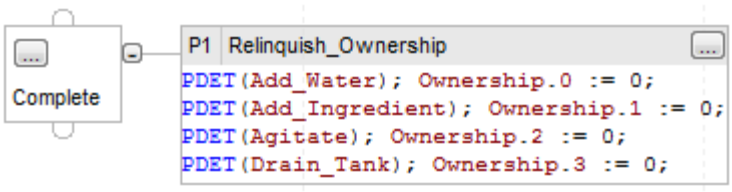
Ladder Diagram

If *Step*.6 = 1 (step 6 in the sequence) then

Each PDET instruction relinquishes ownership of the phases that the sequence owned.
Each *Ownership* bit = 0. (In the *Ownership* tag, each equipment phase is assigned a bit.)
Done = 1. (This signals the sequence to go to the next step.)



Structured Text



When the sequence executes, the Relinquish_Ownership action:

- Relinquishes ownership of the equipment phase.

- Clears the ownership flags (bits that the SFC set when it took ownership of the equipment phases).

Using an action Qualifier of type P1 limits the action execution to the first scan of that step.

See also

Equipment Phase Instructions on page 969

Index Through Arrays on page 1094

Equipment Phase Clear Failure (PCLF)

This instruction applies to the CompactLogix 5370 and CompactLogix 5380, ControlLogix 5570 and ControlLogix 5580, Compact GuardLogix 5370 and Compact GuardLogix 5380, and GuardLogix 5570 and GuardLogix 5580 controllers.

Use the PCLF instruction to clear the failure code of an equipment phase.

The PCLF instruction clears the failure code for an equipment phase.

- Use only a PCLF instruction to clear the failure code of an equipment phase.
- A CLR instruction, MOV instruction, or assignment (:=) *does not* change the failure code of an equipment phase.

Make sure the equipment phase *does not* have other owners when using the PCLF instruction. The PCLF instruction *will not* clear the failure code if Logix Designer, HMI, FactoryTalk Batch software, or another program owns the equipment phase.

- High priority HMI ownership is specific only to the CompactLogix 5370 and ControlLogix 5570 controllers.

Available Languages

Ladder Diagram

Function Block

This instruction is not available in function block.

Structured Text

PCLF(Phase_Name);

Operands

Ladder Diagram

Operand	Type	Format	Description
Phase Name	PHASE	Name of the equipment phase	Equipment phase whose failure code to clear.

Structured Text

The operands are the same as those for the Ladder Diagram PCLF instruction.

Affects Math Status Flags

No

Major/Minor Faults

None. See *Index Through Arrays* below for operand-related faults.

Execution

For Structured Text, EnableIn is always true during normal scan. Therefore, if the instruction is in the control path activated by the logic, it executes.

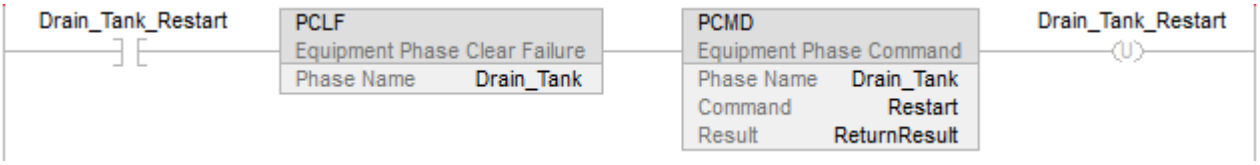
Condition/State	Action Taken
Prescan	No action taken.
Postscan	No action taken.
EnableIn is false	No action taken.
EnableIn is true	The instruction executes as described above.

Example

Ladder Diagram

If *Drain_Tank_Restart* = 1 (restart the *Drain_Tank* equipment phase) then

Clear the failure code of the *Drain_Tank* equipment phase
Change the state of the *Drain_Tank* equipment phase to restarting via the restart command.
Drain_Tank_Restart = 0;



Structured Text

(*If *Drain_Tank_Restart* = on, then

Clear the failure code for the *Drain_Tank* equipment phase.
Restart the *Drain_Tank* equipment phase.
Turn off *Drain_Tank_Restart*.*)

If *Drain_Tank_Restart* Then

PCLF(*Drain_Tank*);
PCMD(*Drain_Tank*,Restart,0);
Drain_Tank_Restart := 0;

End_If;

See also

Equipment Phase Instructions on page 969

Index Through Arrays on page 1094

Equipment Phase Command (PCMD)

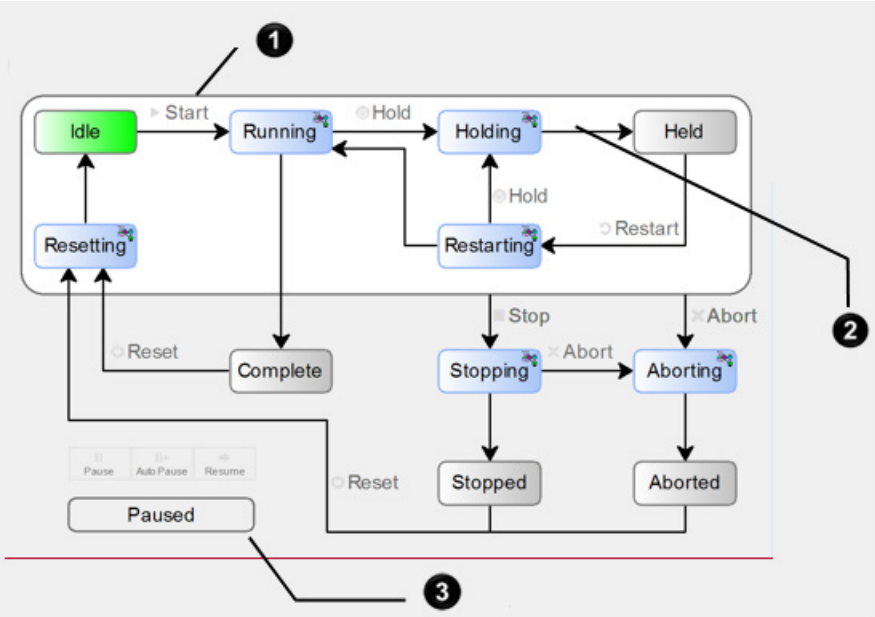
This instruction applies to the CompactLogix 5370 and CompactLogix 5380, ControlLogix 5570 and ControlLogix 5580, Compact GuardLogix 5370 and Compact GuardLogix 5380, and GuardLogix 5570 and GuardLogix 5580 controllers.

The PCMD instruction transitions an equipment phase to the next state or sub-state.

Use the PCMD instruction to change the state or sub-state of an equipment phase.

In the running state routine, use the PSC instructions to transition the equipment phase to the complete state. For more information about Paused functionality, please see the PPD phase instruction.

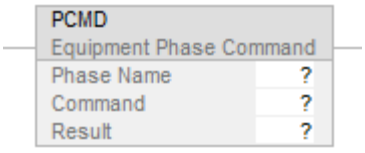
Tip: The PPD instruction is necessary for using pause functionality.



Number	Description
1	Command Some states need a command to go to the next state. If the equipment phase is in the idle state, a start command transitions the equipment phase to the running state. Once in the running state, the equipment phase executes its running state routine.
2	Other states use a *Phase State Complete (PSC)* instruction to go to the next state. If the equipment phase is in the holding state, a PSC instruction transitions the equipment phase to the held state. Once in the held state, the equipment phase needs a restart command to go to the restarting state.
3	Sub-state Use Auto Pause, Pause, and Resume to test and debug a state routine. The sub-states require the *Equipment Phase Paused (PPD)* instruction to create breakpoints in the logic. Use the auto pause, pause, and resume commands to step through the breakpoints.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

PCMD (PhaseName,Command,Result);

Operands

Ladder Diagram

Operand	Type	Format	Description
Phase Name	PHASE	Name of the equipment phase	Equipment phase to change to a different state.
Command	command	Command enumeration value	Command to send to the equipment phase to change its state. For available commands, refer to the illustration above.
Result	DINT	immediate tag	To let the instruction return a code for its success or failure, enter a DINT tag in which to store the result code. Otherwise, enter 0.

Structured Text

The operands are the same as those for the Ladder Diagram PCMD instruction.

Guidelines for using the PCMD Instruction

<table>
<tr><th>Guideline</th><th colspan="2">Details</th></tr>
<tr><td>Limit execution of a PCMD instruction to a single scan.</td><td colspan="2">Limit the execution of the PCMD instruction to a single scan. Each command applies to only a specific state or states. Once the equipment phase changes state, the command is no longer valid. To limit execution, use methods such as:
• Execute the PCMD instruction within a P1 Pulse (Rising Edge) or P0 Pulse (Falling Edge) action.
• Place a one shot instruction before the PCMD instruction.
• Execute the PCMD instruction and then advance to the next step.</td></tr>
<tr><td rowspan="3">Determine if need ownership of the equipment phase.</td><td colspan="2">As an option, a program can own an equipment phase. This prevents another program or FactoryTalk Batch software from also commanding the equipment phase.</td></tr>
<tr><td>If using:</td><td>Then:</td></tr>
<tr><td>FactoryTalk Batch software to also run procedures (recipes) within this controller</td><td>Also run procedures (recipes) within this controller. Before using a PCMD instruction, use an Attach to Equipment Phase (PATT) instruction to take ownership of the equipment phase.</td></tr>
</table>

Guideline	Details	
	Multiple programs to command the same equipment phase	Use an Attach to Equipment Phase (PATT) instruction to take ownership of the equipment phase.
	None of the above	There is no need to own the equipment phase.
Using a POVR instruction instead of a PCMD instruction.	1. Giving the hold, stop, or abort command? • No - Use the PCMD instruction. • Yes - Go to step 2. 2. Must the command work even if the equipment phase is under manual control via Logix Designer? • Yes - Use the POVR instruction instead. • No - Go to step 3. 3. Must the command work even if FactoryTalk Batch software or another program owns the equipment phase? • Yes - Use the POVR instruction instead. • No - Use the PCMD instruction. For example, suppose the equipment checks for jammed material. And if there is a jam, the equipment should always abort. In that case, use the POVR instruction. This way, the equipment aborts even if under manual control via Logix Designer.	
Determine if need a return code.	Use the Result operand to get a code that shows the success or failure of the PCMD instruction.	
	If:	**Then in the Result operand, enter a:**
	Anticipate ownership conflicts or other possible errors	DINT tag, in which to store a code for the result of the execution of the instruction.
	Do *not* anticipate ownership conflicts or other possible errors	0
	Want to interpret the result code, refer to the *Result Codes* table below.	

PCMD Result Codes

If assigning a tag to store the result of a PCMD instruction, the instruction returns one of these codes when it executes:

Code (Dec)	Description
0	Successful command.
24577	Valid command.
24578	Invalid command for the current state of the equipment phase. For example, if the equipment phase is in the running state, then a start command is not valid.
24579	Cannot command the equipment phase. One of these already owns the equipment phase. • Logix Designer • HMI • external sequencer (for example, FactoryTalk Batch software) • another program in the controller
24594	Unscheduled or inhibited equipment phase or in an inhibited task.

Tip: High priority HMI ownership is specific only to CompactLogix 5370 and ControlLogix 5570 controllers.

Affects Math Status Flags

No

Major/Minor Faults

None. See *Index Through Arrays* for operand-related faults.

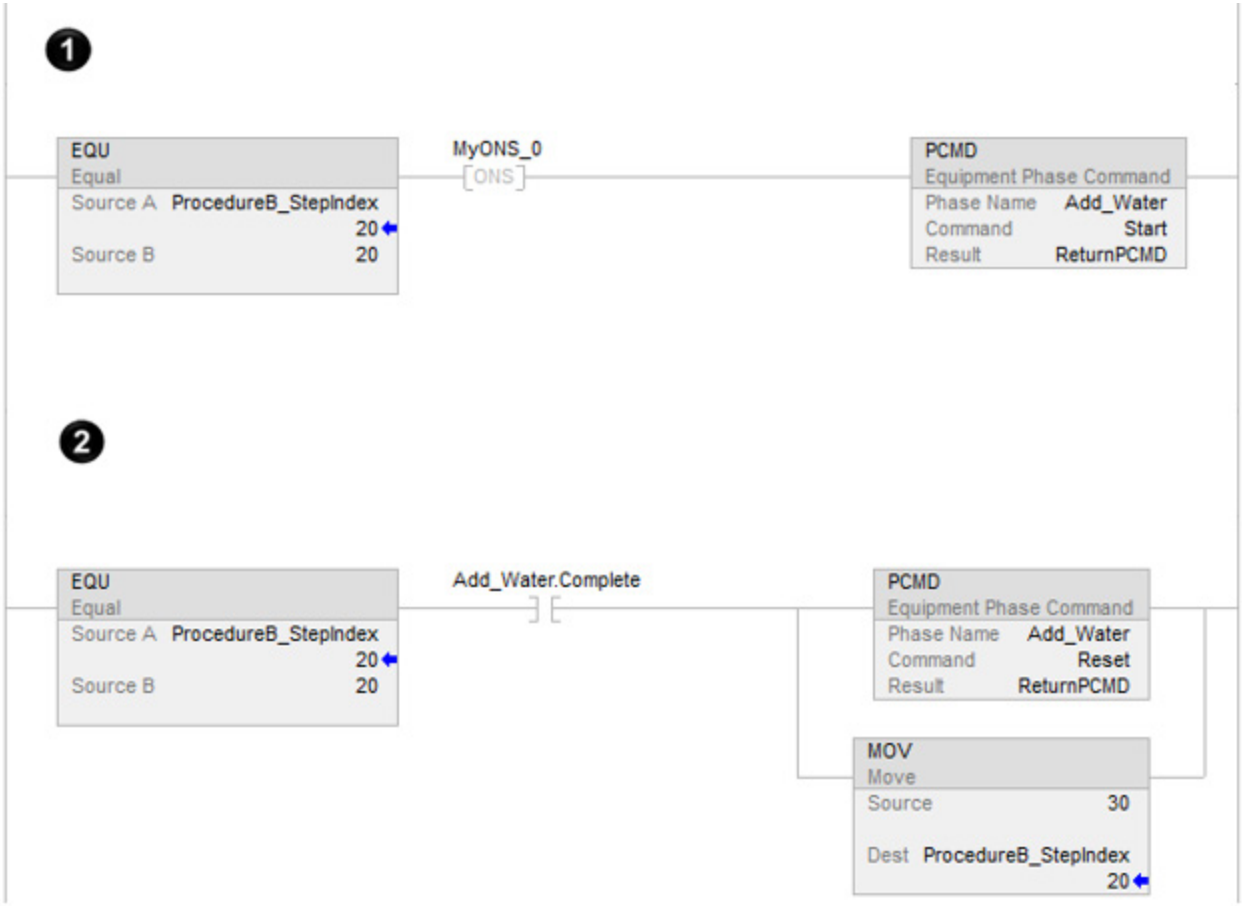
Execution

For Structured Text, EnableIn is always true during normal scan. Therefore, if the instruction is in the control path activated by the logic, it will execute. All conditions below the think solid line can only occur during Normal Scan mode.

Condition/State	Action Taken
Prescan	No action taken.
Postscan	No action taken.
EnableIn is false	No action taken.
EnableIn is true	The instruction executes as described above.

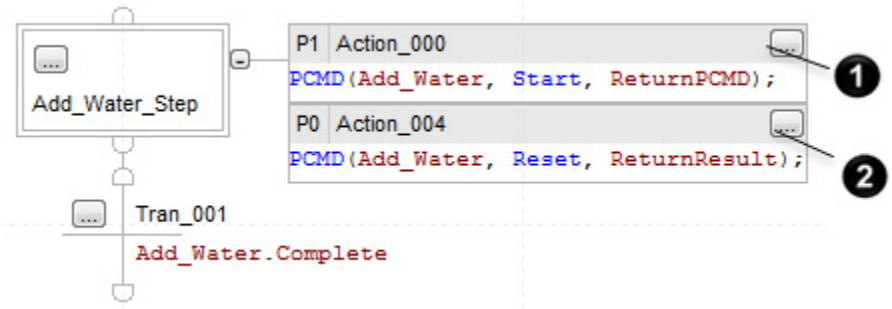
Example 1

Ladder Diagram



Number	Description
1	If ProcedureB_Stepindex = 20 (the routine is at step 20) And this is the transition to step 20 (the ONS instruction signals that the EQU instruction went from false to true.) Then Change the state of the Add_Water equipment phase to running via the start command.
2	If ProcedureB_Stepindex = 20 (the routine is at step 20) And the Add_Water equipment phase is complete (Add_Water.Complete = 1) Then Change the state of the Add_Water equipment phase to resetting via the reset command. Advance to step 30.

Structured Text



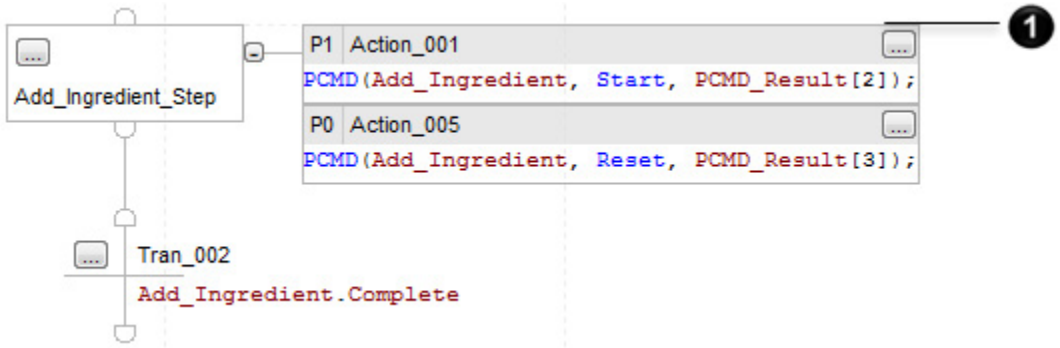
Number	Description
1	When the SFC enters *Add_Water_Step*, change *ADD_Water* equipment phase to running via the start command, The P1 qualifier limits this to the first scan of the step.
2	Before the SFC leaves *Add_Water_Step* (*Add_Water_Complete* = 1), change *Add_Water* equipment phase to resetting via the reset command. The P0 qualifier limits this to the last scan of the step.

Example 2

Ladder Diagram

Number	Description
1	If ProcedureB_Stepindex = 30 (the routine is at step 30) And this is the transition to step 30 (the ONS instruction signals that the EQU instruction went from false to true.) Then Change the state of the Add_Water equipment phase to running via the start command. Verify that the command was successful and store the result code in PCMD_Result[1] [DINT Tag].

Structured Text



Number	Description
1	When the SFC enters *Add_Ingredient_Step* • Change *Add_Ingredient* equipment phase to running via the start commend. • Verify that the command was successful and store the result code in *PCDM_Result[2]* (DINT tag). The P1 qualifier limits this to the first scan of the step.

See also

Equipment Phase Instructions on page 969

Index Through Arrays on page 1094

Equipment Phase Paused (PPD)

Equipment Phase External Request (PXRQ)

This instruction applies to the CompactLogix 5370 and CompactLogix 5380, ControlLogix 5570 and ControlLogix 5580, Compact GuardLogix 5370 and Compact GuardLogix 5380, and GuardLogix 5570 and GuardLogix 5580 controllers.

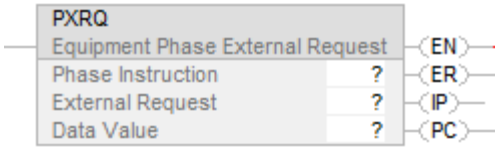
Use the PXRQ instruction to initiate communication with FactoryTalk® Batch software.

IMPORTANT When the PXRQ instruction is used in an Equipment Sequence, only the Download All (1000) request and the Upload All (2000) request are supported. All other PXRQ instruction requests are ignored.

The PXRQ instruction sends a request to FactoryTalk Batch software.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

PXRQ (Phase_Instruction, External_Request, Data_Value);

Operands

Ladder Diagram

Operand	Type	Format	Description
Phase Instruction	PHASE_INSTRUCTION	tag	Tag that controls the operation.
External Request	request	enumeration value	Type of request.
Data Value	DINT	array tag	Parameters of the request.

Structured Text

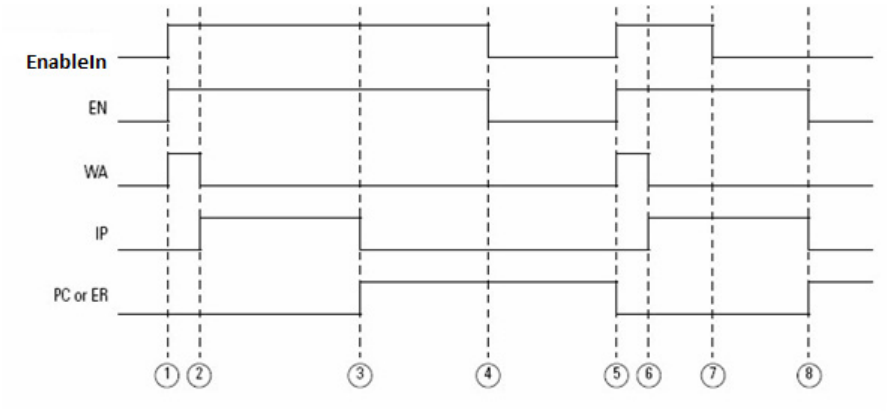
The operands are the same as those for the Ladder Diagram PXRQ instruction.

PHASE_INSTRUCTION Data Type

If want to:	Then check or set this member:	Data type	Notes
Determine if a false-to-true transition caused the instruction to execute	EN	BOOL	Refer to the timing diagram below.
Determine if the request failed	ER	BOOL	Refer to the timing diagram below. To diagnose the error, refer to the ERR and EXERR values.
Determine if the FactoryTalk Batch software completed its processing of the request	PC	BOOL	Refer to the timing diagram below.
Determine if the FactoryTalk Batch software is processing the request	IP	BOOL	Refer to the timing diagram below.
Determine if the instruction sent the request but FactoryTalk Batch software has not yet acknowledged it	WA	BOOL	Refer to the timing diagram below. WA also = 0 if: • The connection times out • A network error occurs • ABORT = 1

If want to:	Then check or set this member:	Data type	Notes
Cancel the request	ABORT	BOOL	To abort (cancel) the request, set the ABORT bit = 1. When the controller aborts the instruction: • ER = 1 • ERR shows the result of the abort
• Diagnose the cause of an error	ERR	INT	If ER = 1, the error code gives diagnostic information. To interpret the error code, see *PXRQ Error Codes*.
• Write logic to respond to specific errors	EXERR	INT	If ER = 1, the extended error code gives additional diagnostic information for some errors. To interpret the extended error code, see *PXRQ Error Codes*.
Use one member for the various status bits of the tag	STATUS	DINT	**For this member:** / **Use this bit:** EN / 31 ER / 28 PC / 27 IP / 26 WA / 25 ABORT / 24

Timing Diagram



Guidelines for using the PXRQ Instruction

Guideline	Details
Make sure to use an array for the Data Values operand.	The Data Values operand requires a DINT array, even if the array contains only 1 element (that is, the data type is DINT[1]).
In Ladder Diagram, condition the instruction to execute on a transition.	This is a transitional instruction. Each time the instruction executes, toggle the EnableIn from false to true

Guideline	Details

In Structured Text, use a construct to condition the execution of the instruction.	When programming a PXRQ instruction in structured text, consider:
	• In structured text, instructions execute *each time* they are scanned.
	• The PXRQ instruction updates its status bits *only* when it is scanned.
	• To keep the instruction from repeatedly executing but ensure that the status bits update, enclose the instruction in a construct that:
	  • Initiates the execution of the instruction *only* on a transition (change in conditions).
	  • Remains true until either PC = 1 or ER = 1

Configure the PXRQ Instruction

If want to:	Then configure the PXRQ instruction as follows:		
	External Request	**Data Value Array Element**	**Value**
Download all input parameters	Download Input Parameters	DINT[0]	0
Download a single input parameter	Download Input Parameters	DINT[0]	parameter ID
Download a range of input parameters	Download Input Parameters	DINT[0]	parameter ID of the first parameter
		DINT[1]	Number of parameters to download
Download the input parameters configured for automatic download on start or transfer of control	Download Input Parameters Subset	DINT[0]	start = 1 transfer of control = 2
Download all output parameters	Download Output Parameter Limits	DINT[0]	0
Download a single output parameter	Download Output Parameter Limits	DINT[0]	parameter ID
Upload all reports	Upload Output Parameters	DINT[0]	0
Upload a single report	Upload Output Parameters	DINT[0]	report ID
Upload a range of reports	Upload Output Parameters	DINT[0]	report ID of the first report
		DINT[1]	Number of reports to download
Upload the output parameters configured for automatic upload on terminal state or transfer of control	Upload Output Parameters Subset	DINT[0]	terminal = 1 transfer of control = 2
Send a message to an operator	Send Message to Operator	DINT[0]	message ID
Clear a message from an operator	Clear Message to Operator	DINT[0]	0
Acquire a resource	Acquire Resources	DINT[0]	equipment ID
Acquire multiple resources	Acquire Resources	DINT[0]	equipment ID
		DINT[1]	equipment ID
			...
Release a single resource	Release Resources	DINT[0]	equipment ID

If want to:	Then configure the PXRQ instruction as follows:		
	External Request	**Data Value Array Element**	**Value**
Release multiple resources	Release Resources	DINT[0]	equipment ID
		DINT[1]	equipment ID
			...
Release all resources	Release Resources	DINT[0]	0
Send a message (and optional data) to another phase	Send Message to Linked Phase	DINT[0]	message ID
		DINT[1]	Number of phases to receive message
		DINT[2]	Message Value
		DINT[3]	Message Value
			...
Send a message (and optional data) to another phase and wait for the phase to receive the message	Send Message to Linked Phase and Wait	DINT[0]	message ID
		DINT[1]	Number of phases to receive message
		DINT[2]	Message Value
		DINT[3]	Message Value
			...
Wait to receive a message from another phase	Receive Message From Linked Phase	DINT[0]	message ID
		DINT[1]	Message Value
		DINT[2]	Message Value
			...
Cancel a message to another phase	Cancel Message to Linked Phase	DINT[0]	message ID
Cancel all messages to another phase	Cancel Message to Linked Phase	DINT[0]	0
Download customer's batch ID	Download Batch Data	DINT[0]	1
		DINT[1]	parameter ID in which to store the value
Download unique batch ID	Download Batch Data	DINT[0]	2
		DINT[1]	parameter ID in which to store the value
Download phase ID	Download Batch Data	DINT[0]	3
		DINT[1]	parameter ID in which to store the value
	Download Batch Data	DINT[0]	4

If want to:	Then configure the PXRQ instruction as follows:		
	External Request	**Data Value Array Element**	**Value**
Download recipe control versus manual phase control		DINT[1]	parameter ID in which to store the value
Download current mode of the phase	Download Batch Data	DINT[0]	5
		DINT[1]	parameter ID in which to store the value
Download the low limit of an input parameter	Download Batch Data	DINT[0]	6 The input parameter tag stores the low limit.
Download the high limit of an input parameter	Download Batch Data	DINT[0]	7 The input parameter tag stores the high limit.
Download data about the container currently in use	Download Material Manager Data Container In Use	DINT[0]	1
		DINT[1]	Attribute ID (specify the single attribute).
		DINT[2]	Phase parameter ID (specify the parameter tag to download the value to)
Download data about the current material inside the container currently in use	Download Material Manager Data Container In Use	DINT[0]	2
		DINT[1]	Attribute ID (specify the single attribute).
		DINT[2]	Phase parameter ID (specify the parameter tag to which to download the value)
Download data about the current lot inside the container currently in use	Download Material Manager Data Container In Use	DINT[0]	3
		DINT[1]	Attribute ID (specify the single attribute).
		DINT[2]	Phase parameter ID (specify the parameter tag to which to download the value)
Upload data about the container currently in use	Upload Material Manager Data Container In Use	DINT[0]	1
		DINT[1]	Attribute ID (specify the single attribute).
		DINT[2]	Phase parameter ID (specify the parameter tag from which to upload the value)
Upload data about the current material inside the container currently in use	Upload Material Manager Data Container In Use	DINT[0]	2
		DINT[1]	Attribute ID (specify the single attribute).
		DINT[2]	Phase parameter ID (specify the parameter tag from which to upload the value)
Upload data about the current lot inside the container currently in use	Upload Material Manager Data Container In Use	DINT[0]	3
		DINT[1]	Attribute ID (specify the single attribute).

If want to:	Then configure the PXRQ instruction as follows:		
	External Request	**Data Value Array Element**	**Value**
		DINT[2]	Phase parameter ID (specify the parameter)
Download the current binding's container priority	Download Container Binding Priority	DINT[0]	parameter ID in which to store the value
Upload a new container priority for the current binding	Upload Container Binding Priority	DINT[0]	parameter ID that has value
Download information regarding the availability of sufficient material	Download Sufficient Material	DINT[0]	parameter ID in which to store the value In the result value: • 0 = insufficient material • 1 = sufficient material
Generate a signature	Generate E Signature	DINT[0]	ID of the signature template
		DINT[1]	
Download material attribute	Download Material Manager Database Data	DINT[0]	0
		DINT[1]	Phase parameter ID (specify the parameter tag to which to download the value)
		DINT[2]	Material controller ID
		DINT[3]	Attribute ID (specify the single attribute)
Download lot attribute	Download Material Manager Database Data	DINT[0]	1
		DINT[1]	Phase parameter ID (specify the parameter tag to which to download the value)
		DINT[2]	Material controller ID
		DINT[3]	Attribute ID (specify the single attribute)
Download container attribute	Download Material Manager Database Data	DINT[0]	3
		DINT[1]	Phase parameter ID (specify the parameter tag to which to download the value)
		DINT[2]	Material controller ID
		DINT[3]	Attribute ID (specify the single attribute)
Download container priority assignment	Download Material Manager Database Data	DINT[0]	4
		DINT[1]	Phase parameter ID (specify the parameter tag to which to download the value)
		DINT[2]	Material controller ID
		DINT[3]	Attribute ID (specify the single attribute)
		DINT[4]	

If want to:	Then configure the PXRQ instruction as follows:		
	External Request	**Data Value Array Element**	**Value**
Upload material attribute	Upload Material Manager Database Data	DINT[0]	5
		DINT[1]	Phase report ID (specify the phase report tag from which to upload)
		DINT[2]	Material controller ID
		DINT[3]	Attribute ID (specify the single attribute)
Upload lot attribute	Upload Material Manager Database Data	DINT[0]	6
		DINT[1]	Phase report ID (specify the phase report tag from which to upload)
		DINT[2]	Material controller ID
		DINT[3]	Attribute ID (specify the single attribute)
Upload container attribute	Upload Material Manager Database Data	DINT[0]	8
		DINT[1]	Phase report ID (specify the phase report tag from which to upload)
		DINT[2]	Material controller ID
		DINT[3]	Attribute ID (specify the single attribute)
Download container priority assignment	Download Material Manager Database Data	DINT[0]	9
		DINT[1]	Phase parameter ID (specify the parameter tag to which to download the value)
		DINT[2]	Material controller ID
		DINT[3]	Attribute ID (specify the single attribute)
		DINT[4]	

PXRQ Error Codes

ERR (hex)	EXERR (hex)	Description	Recommended Action
00	0000	The PXRQ instruction was aborted before it sent the request to FactoryTalk Batch software.	None
01	0000	The PXRQ instruction was aborted after it sent the request to FactoryTalk Batch software	None
02	0000	Two or more PXRQ instructions executed at the same time using the same request type	Limit execution to one PXRQ instruction at a time.

ERR (hex)	EXERR (hex)	Description	Recommended Action
03	0110	Communication error. The request was not delivered because there is no subscriber to the phase	Check that FactoryTalk Batch software is connected and running.
	0210	Communication error. The request was not delivered because there is no connection to the Notify object.	
	0410	Communication error. Delivery failed.	
	1010	Communication error. The request was not delivered because FactoryTalk Batch software does not subscribe to receive the external request.	
	1020	FactoryTalk Batch software is not attached to the phase.	
04	0002	The FactoryTalk Batch software encountered an error while processing the request.	Check the connection and communication path to FactoryTalk Batch software.
	0003	The PXRQ instruction contains an invalid value.	
	0004	FactoryTalk Batch software is not in the proper state to process the request.	
	0005	Two or more PXRQ instructions executed at the same time using different request types	
	0006	Error storing to parameter tags at end of request processing.	
05	0000	FactoryTalk Batch software received the request but passed back an invalid cookie.	Check the connection and communication path to FactoryTalk Batch software.
06	0000	PXRQ instruction sent an invalid parameter to FactoryTalk Batch software.	Check the connection and communication path to FactoryTalk Batch software.
07	0000	Communication lost while PXRQ is waiting for response from the external sequencer.	Check the connection and communication path to FactoryTalk Batch software.

Affects Math Status Flags

No

Major/Minor Faults

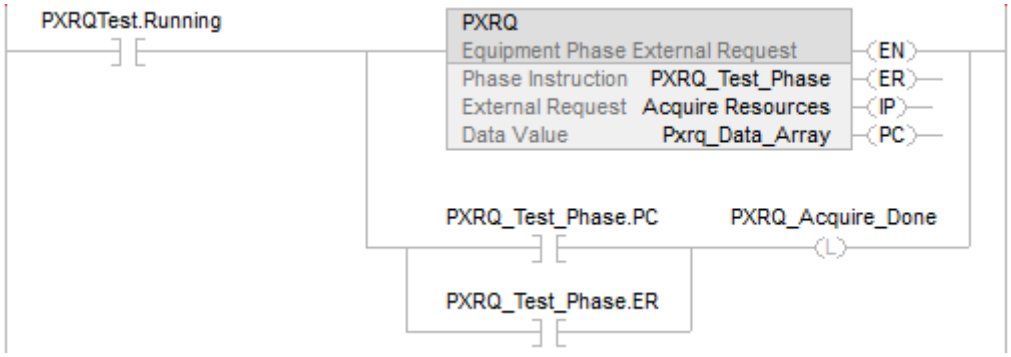
None. See *Index Through Arrays* for operand-related faults.

Execution

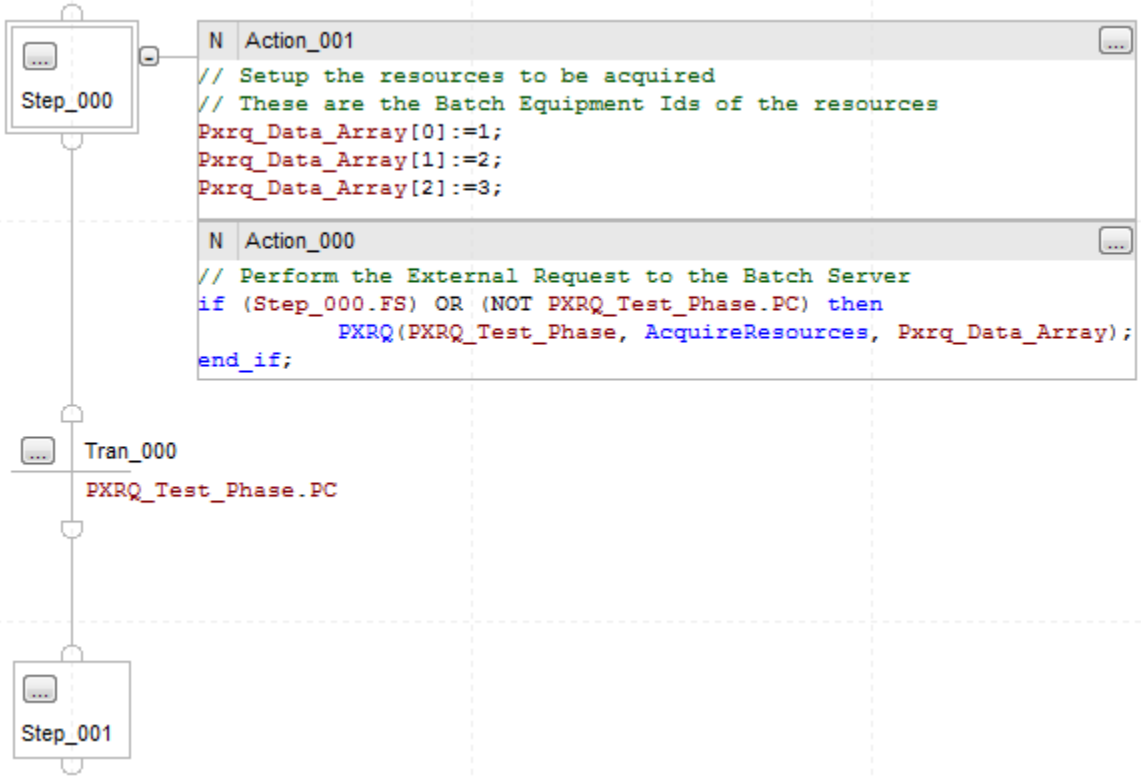
Condition/State	Action Taken
Prescan	No action taken.
Postscan	No action taken.
EnableIn is false	No action taken.
EnableIn is true	The instruction executes.

Example

Ladder Diagram



Structured Text



See also

Equipment Phase Instructions on page 969

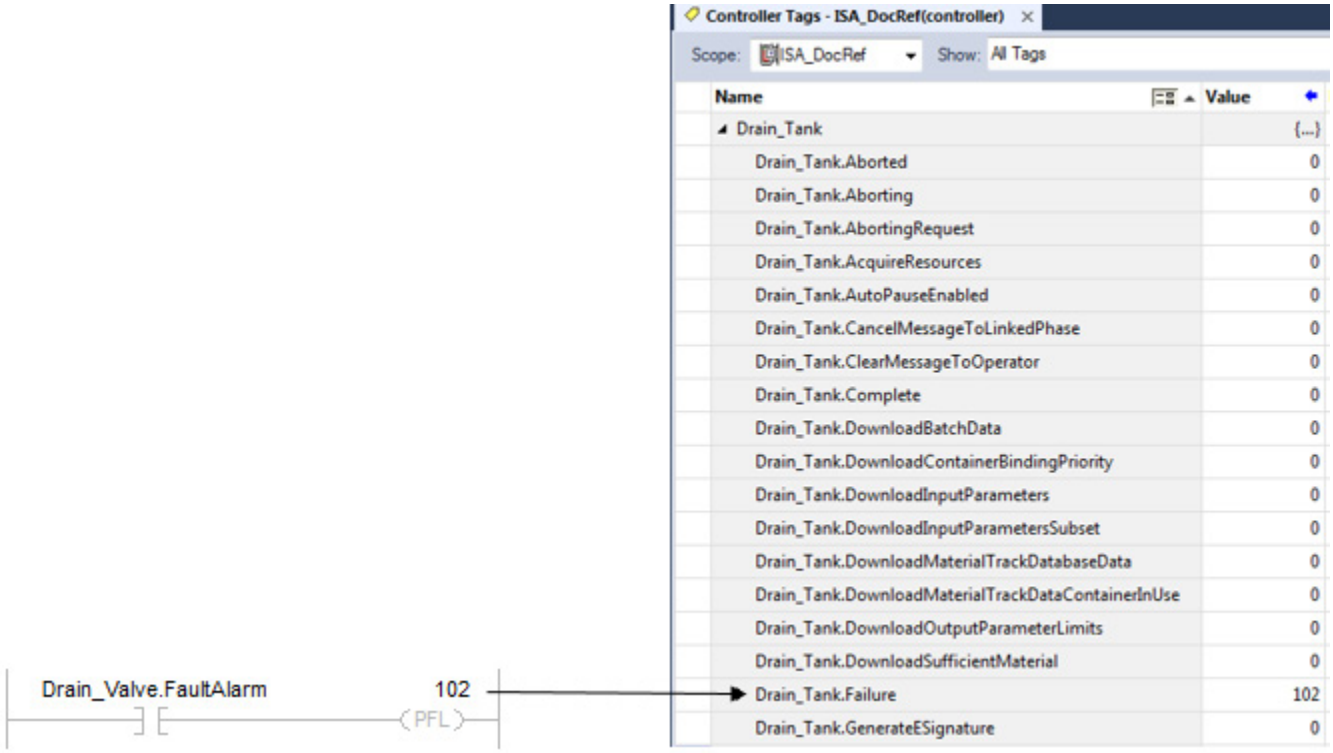
Index Through Arrays on page 1094

Equipment Phase Failure (PFL)

This instruction applies to the CompactLogix 5370 and CompactLogix 5380, ControlLogix 5570 and ControlLogix 5580, Compact GuardLogix 5370 and Compact GuardLogix 5380, and GuardLogix 5570 and GuardLogix 5580 controllers.

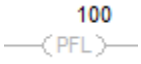
Use the PFL instruction as an optional method to signal a failure for an equipment phase.

The PFL instruction sets the value of the failure code for an equipment phase. Use the instruction to signal a specific failure for an equipment phase, such as a specific device has faulted. The PFL instruction sets the failure code only to a value greater than its current value.



Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

PFL (Failure_Code);

Operands

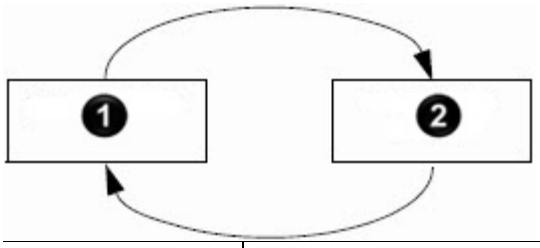
Ladder Diagram

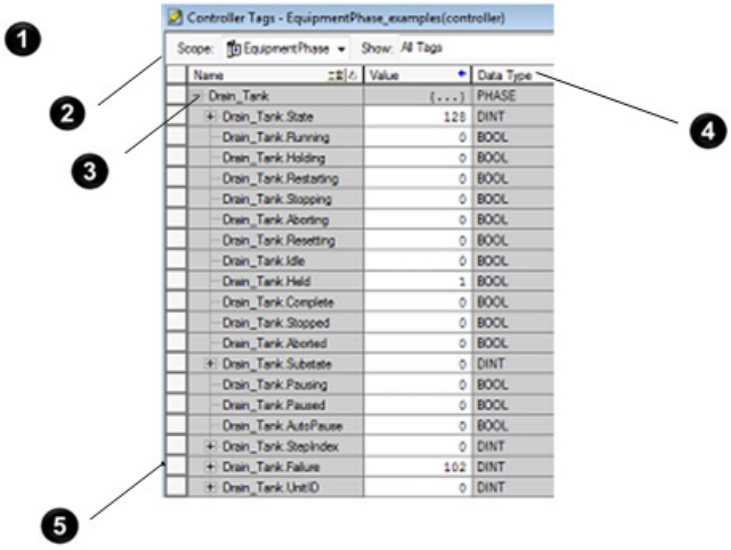
Operand	Type	Format	Description
Failure_Code	DINT	immediate tag	Value to set the failure code for the equipment phase. If a negative failure code given, it evaluates as 0.

Structured Text

The operands are the same as those for the Ladder Diagram PFL instruction.

Guidelines for using the PFL Instruction

Guideline	Details
Put the PFL instruction in the equipment phase.	The PFL instruction sets the failure code for the equipment phase in which the instruction is put. There is *no* operand to identify a specific equipment phase. Typically, put the PFL instruction in a prestate routine for the equipment phase. • The controller always scans the prestate routine, even when an equipment phase is in the idle state. • The controller scans the prestate routine before *each* scan of a state.  **Number** / **Description** 1 — Prestate routine 2 — Current state routine Use the progress routine to continuously monitor the health of an equipment phase as it progresses through its states.

<table>
<tr><th>Guideline</th><th>Details</th></tr>
<tr><td>Prioritize failure codes.</td><td>The PFL instruction sets the failure code only to a value greater than its current value.
For example, if a PFL instruction sets the failure code = 102, another PFL instruction can only set the failure code > 102.
Make sure to assign higher values to exceptions that require higher priority in their handling. Otherwise, a lower priority exception may overwrite a more critical exception.</td></tr>
<tr><td>To take action when a failure occurs, monitor the Failure member of the PHASE tag.</td><td>The PFL instruction writes its value to the Failure member of the PHASE tag for the equipment phase. 

<table>
<tr><th>Number</th><th>Description</th></tr>
<tr><td>1</td><td>When creating an equipment phase, the Logix Designer application creates a tag for the status of the equipment phase.</td></tr>
<tr><td>2</td><td>controller scope</td></tr>
<tr><td>3</td><td>Name = phase_name</td></tr>
<tr><td>4</td><td>PHASE data type</td></tr>
<tr><td>5</td><td>The PFL instruction writes its value to the failure member for the equipment phase.</td></tr>
</table></td></tr>
<tr><td>To clear the failure code, use a PCLF instruction.</td><td>Use a PCLF instruction to clear the failure code of an equipment phase. Instructions such as a CLR or MOV will not change the failure code.</td></tr>
</table>

Affects Math Status Flags

No.

Major/Minor Faults

A major fault will occur if:	Fault type	Fault code
Instruction is called from outside an Equipment Phase program.	4	91

See *Index Through Arrays* below for array-index faults.

Execution

For Structured Text, EnableIn is always true during normal scan. Therefore, if the instruction is in the control path activated by the logic, it will execute. All conditions below the thick solid line can only occur during Normal Scan mode.

Condition/State	Action Taken
Prescan	No action taken.
Postscan	No action taken.
EnableIn is false	No action taken.
EnableIn is true	The instruction executes as described above.

Example

Ladder Diagram

In the prestate routine of an equipment phase...

If the *Drain_Valve.FaultAlarm* = 1 (The valve did not go to the commander state.) then

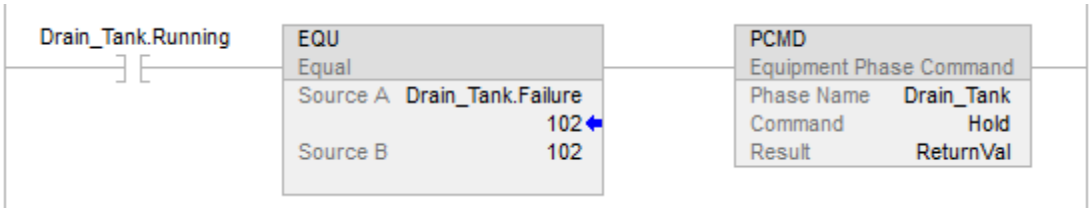
Failure code for the equipment phase = 102.

If *Drain_Tank.Running* = 1 (The *Drain_Tank* equipment phase is in the running state.)

And *Drain_Tank.Failure* = 102 (failure code for the equipment phase)

Then

Change the state of the *Drain_Tank* equipment phase to holding via the hold command.



Structured Text

In the prestate routine of an equipment phase...

```
(*If the drain valve does not go to the commanded state, then set the
failure code of this equipment phase = 102.*)
If Drain_Valve.FaultAlarm Then
    PFL(102);
End_If;

(*If the Drain_Tank equipment phase = running and its failure code = 102,
issue the hold command and send the equipment phase to the holding state.*)
If Drain_Tank.Running And (Drain_Tank.Failure = 102) Then
    PCMD(Drain_Tank,hold,0);
End_IF;
```

See also

Equipment Phase Instructions on page 969

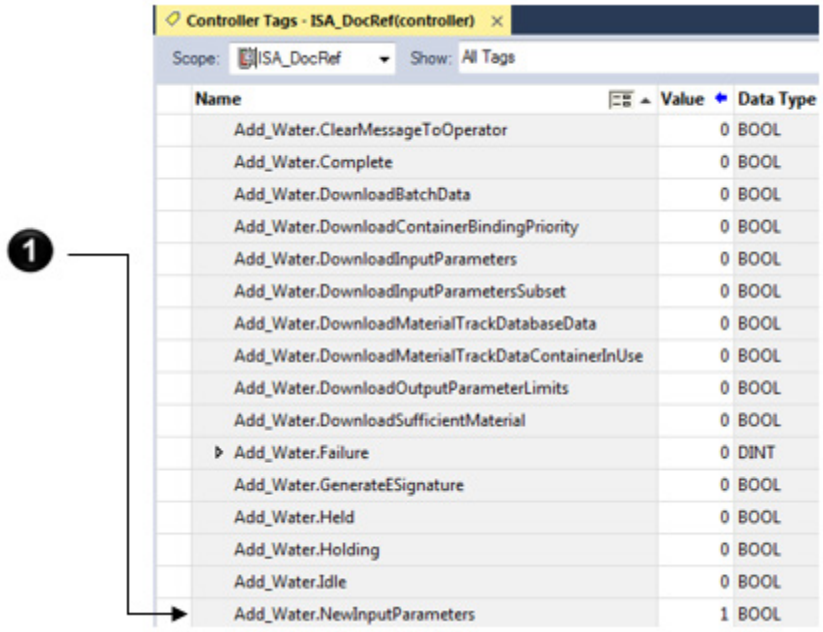
Index Through Arrays on page 1094

Equipment Phase New Parameters (PRNP)

This instruction applies to the CompactLogix 5370 and CompactLogix 5380, ControlLogix 5570 and ControlLogix 5580, Compact GuardLogix 5370 and Compact GuardLogix 5380, and GuardLogix 5570 and GuardLogix 5580 controllers.

Use the PRNP instruction to clear the NewInputParameters bit of an equipment phase.

The PRNP instruction clears the NewInputParameters bit of the equipment phase.



Number	Description
1	When FactoryTalk Batch software has new parameters for an equipment phase, it sets the NewInputParameters bit for the phase. After downloading the parameters, use the PRNP instruction to clear the bit.

Available Languages

Ladder Diagram

Function Block

This instruction is not available in function block.

Structured Text

PRNP ( );

Operands

Ladder Diagram

None

Structured Text

None

Enter the parentheses ( ) after the instruction mnemonic, even though there are no operands.

Affects Math Status Flags

No

Major/Minor Faults

None. See *Index Through Arrays* for operand-related faults.

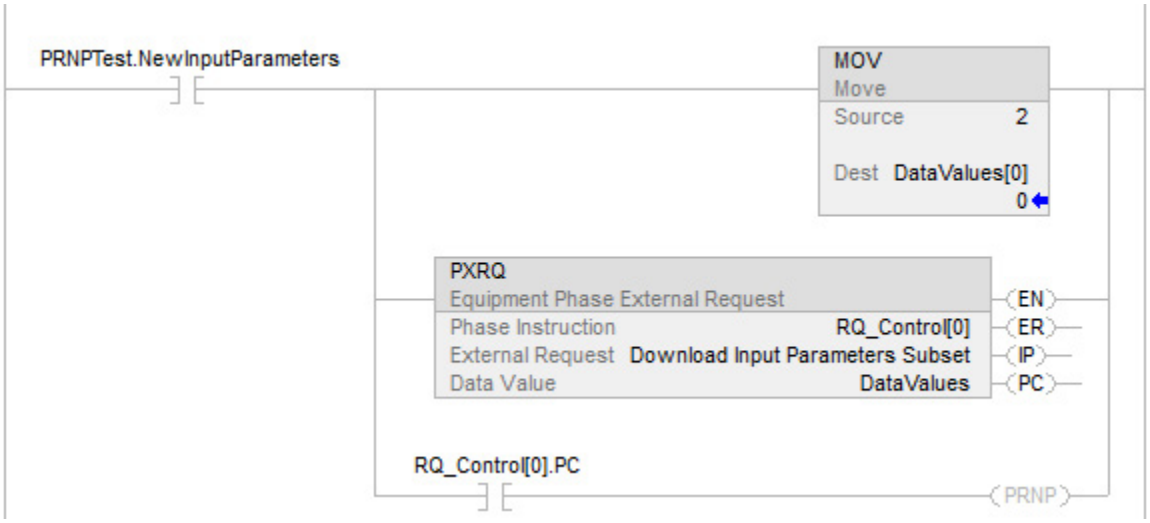
Execution

Condition/State	Action Taken
Prescan	No action taken.
Postscan	No action taken.

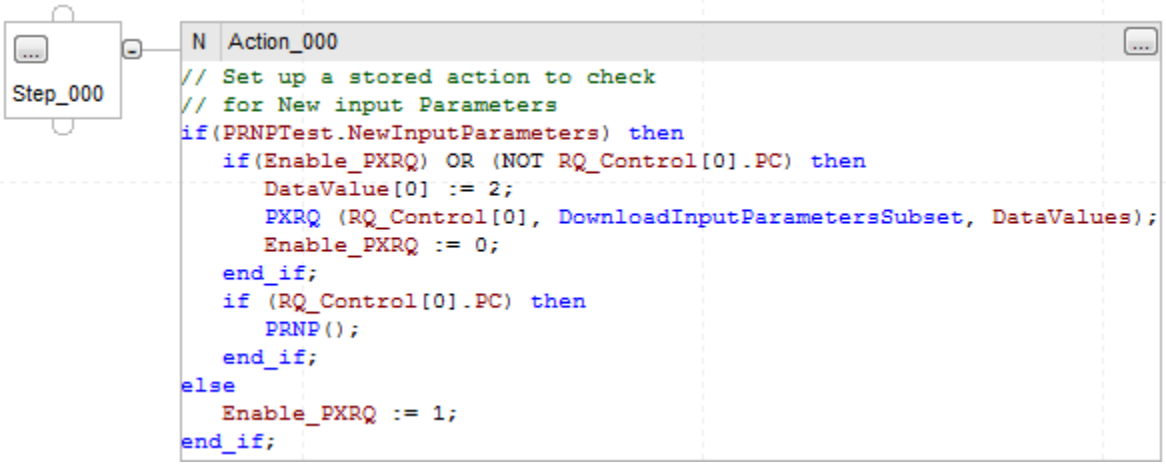
EnableIn is false	No action taken.
EnableIn is true	The instruction executes.

Example

Ladder Diagram



Structured Text



If PRNPTest.NewInputParameters = 1 (FactoryTalk Batch software has new input parameter for the equipment phase), then

If Enable_PXRQ = 1 (Let the PXRQ instruction execute.)
Or RQ_Control[0].PC = 0 (The PXRQ instruction is in process.), then

Datavalues[0] = this set the PXRQ instruction for transfer of control.
Send the Download Input Parameters Subset request to FactoryTalk Batch software.

Send DataValues[0] = 2, the instruction is set for transfer of control.
Enable_PXRQ = 0 (Do no let the PXRQ instruction restart after the request completes)

If RQ_Control[0].PC = 1 (The request is complete.), then

ThisPhase.NewInputParameters = 0 via the PRNP instruction

Otherwise

Enable_PXRQ = 1 (Let the PXRQ instruction execute the next time new input parameters are available.)

See also

Equipment Phase Instructions on page 969

Index Through Arrays on page 1094

Equipment Phase Override Command (POVR)

This instruction applies to the CompactLogix 5370 and CompactLogix 5380, ControlLogix 5570 and ControlLogix 5580, Compact GuardLogix 5370 and Compact GuardLogix 5380, and GuardLogix 5570 and GuardLogix 5580 controllers.

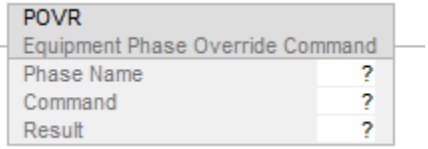
Use the POVR instruction to give a Hold, Stop, or Abort command to an equipment phase, regardless of ownership.

The POVR instruction:

- Gives the Hold, Stop, or Abort command to an equipment phase.
- Overrides all owners of the equipment phase. The command works even if Logix Designer software, HMI, FactoryTalk Batch software, or another program already owns the equipment phase. This instruction does not change the ownership of the equipment phase.
- High priority HMI ownership is specific only to CompactLogix 5370 and ControlLogix 5570 controllers.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

POVR (PhaseName, Command, Result);

Operands

Ladder Diagram

Operand	Type	Format	Description
Phase Name	phase	Name of the equipment phase	Equipment phase to change to a different state
Command	command	Name of the command	One of these commands for the equipment phase: • Hold • Stop • Abort
Result	DINT	immediate tag	To let the instruction return a code for its success or failure, enter a DINT tag in which to store the result code. Otherwise, enter 0.

Structured Text

The operands are the same as those for the Ladder Diagram POVR instruction.

Guidelines for using the POVR Instruction

Guideline	Details
If want to override other owners.	Want the equipment to hold, stop, or abort even if have manual control of the equipment phase via Logix Designer software? • Yes - Use the POVR instruction • No - Use the PCMD instruction This also applies to HMI, FactoryTalk Batch software or other programs. Use the POVR only to hold, stop, or abort, regardless of ownership. For example, suppose the equipment checks for jammed material. If there is a jam, always abort the equipment. In that case, use the POVR instruction. This way, the equipment aborts even under manual control via Logix Designer software.

Guideline	Details
Limit execution of a POVR instruction to a single scan.	Limit the execution of the POVR instruction to a single scan. Each command applies to only a specific state or states. Once the equipment phase changes state, the command is *no longer* valid. To limit execution, use methods such as: • Execute the POVR instruction within a P1 Pulse (Rising Edge) or P0 Pulse (Falling Edge) action. • Place a one shot instruction before the POVR instruction. • Execute the POVR instruction and then advance to the next step.

POVR Result Codes

If assigning a tag to store the result of a POVR instruction, the instruction returns one of these codes when it executes:

Code (Dec)	Description
0	Successful command.
24577	Invalid command.
24578	Invalid command for the current state of the equipment phase. For example, if the equipment phase is in the stopping state, then a hold command is not valid.
24594	Unscheduled or inhibited equipment phase or in an inhibited task.

Affects Math Status Flags

No

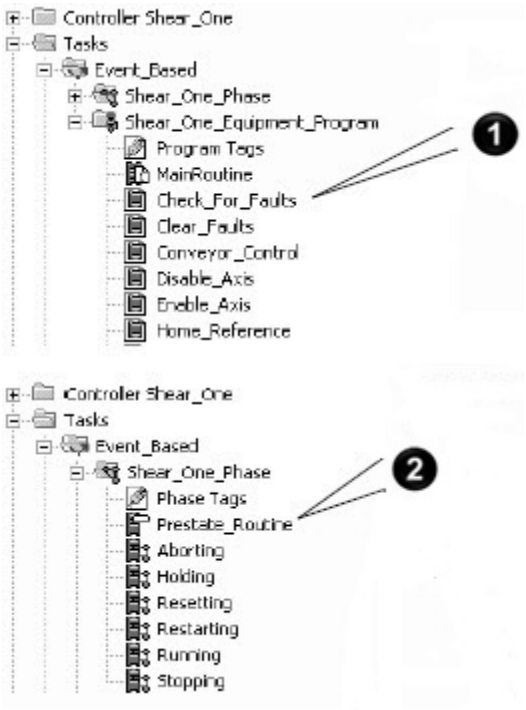
Major/Minor Faults

None. See *Index Through Arrays* for operand-related faults.

Execution

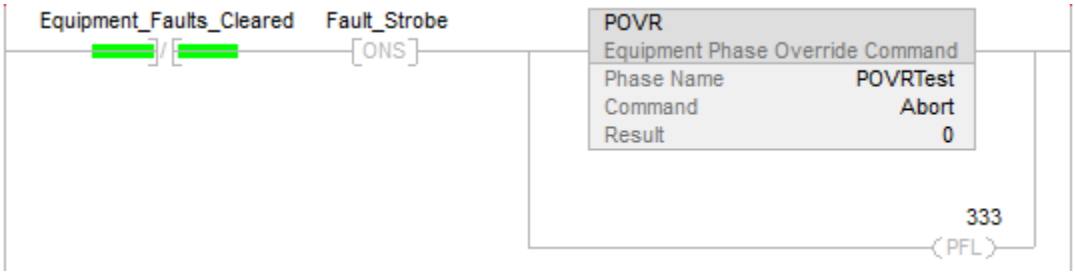
Condition/State	Action Taken
Prescan	No action taken.
Postscan	No action taken.
EnableIn is false	No action taken.
EnableIn is true	The instruction executes.

Example

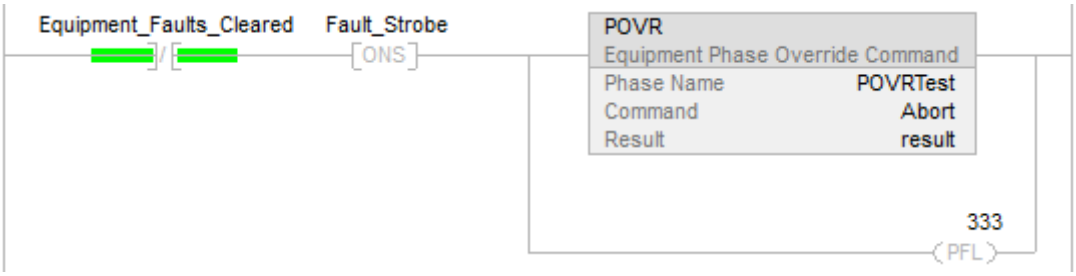


Number	Description
1	The equipment program watches for the these faults: • Faulted axis • Jammed material If there is a fault, then *Local_Interface.Equipment_Faults_Cleared* = 0. This tag is an alias for the controller-scoped tag *Shear_1*.
2	The prestate routine of the equipment phase watches for the equipment program to signal a fault. • If *Interface_To_Equipment.Equipment_Faults_Cleared*=0 then there is a fault. • Both *Interface_To_Equipment* and *Local_Interface* as aliases for *Shear_1*, so they have the same values. If there is a fault, then Give the *Shear_One_Phase* equipment phase the abort command. The POVR instruction makes sure the command works, even if someone has manual control of the equipment phase through Logix Designer software. The PFL instruction sets the failure code for *Shear_One_Phase = 333*. The *Fault_Strobe* keeps these actions to a single scan.

Ladder Diagram



Example 2



Structured Text

If NOT Equipment_Faults_Cleared And NOT Fault_Strobe then

```
POVR(POVRTest,Abort, 0);
PFL(333);
```

end_if;

Fault_Strobe := NOT Equipment_Faults_Cleared;

See also

Equipment Phase Instructions on page 969

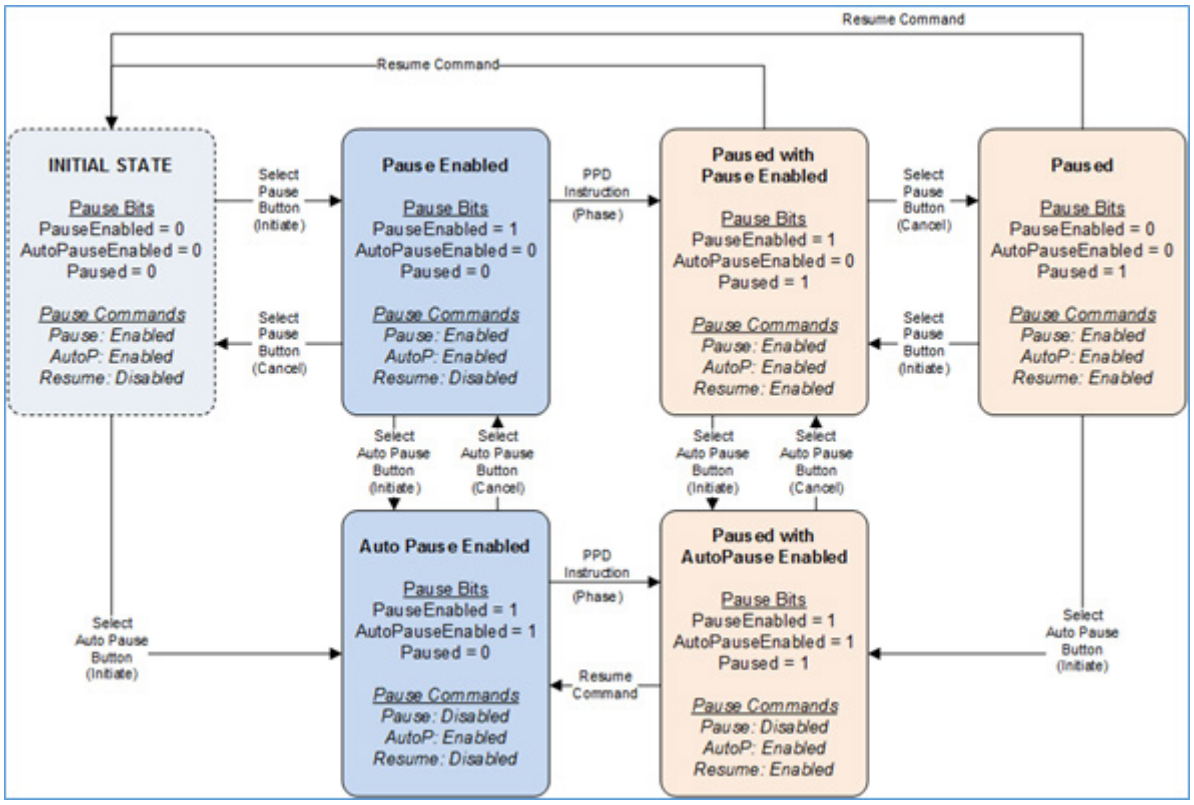
Index Through Arrays on page 1094

Equipment Phase Paused (PPD)

This instruction applies to the CompactLogix 5370 and CompactLogix 5380, ControlLogix 5570 and ControlLogix 5580, Compact GuardLogix 5370 and Compact GuardLogix 5380, and GuardLogix 5570 and GuardLogix 5580 controllers.

Use the PPD instruction to set up breakpoints within the logic of an equipment phase.

Pausing an equipment phase requires configuring break points by coding a PPD instruction in a phase's state routine logic. The phase state routine pauses when the PPD instruction executes and the phase receives a command to pause at its next opportunity.



Three operator commands relate to the pause functionality:

- **Pause**: The Pause command enables or disables pausing execution of the phase when the next PPD instruction executes. The Pause command toggles the PauseEnabled bit ON (1) or OFF (0).
- **AutoPause:** The AutoPause command enables or disables automatically enabling pausing the phase after processing a Resume command. The AutoPause command toggles the AutoPauseEnabled bit ON (1) or OFF (0).
- **Resume:** The Resume command directs the firmware to resume executing the phase state routine logic. Resume sets the PauseEnabled and Paused bits OFF (0).

The Pause substate uses these three bits:

- **PauseEnabled**: The PauseEnabled bit maintains the status of processing a Pause command. This is bit 0 of the Pause substate.

  When Paused is ON (1), execution of a PPD instruction pauses execution of the state routine's logic. This bit updates when it receives a Pause command (toggling the bit's value). Additionally, a Resume command sets PauseEnabled OFF (0), if the AutoPauseEnabled bit is ON (1).
- **AutoPauseEnabled**: The AutoPauseEnabled bit maintains the status of automatically enabling pausing immediately after a Resume command. This is bit 2 of the Pause substate.

This bit updates when it receives an AutoPause command (toggling the bit's value). When AutoPauseEnabled is ON (1) and the phase is Paused, the Resume command leaves PauseEnabled ON (1).

- **Paused**: The Paused bit maintains the pause-state of the phase, Paused (1) or not-Paused (0). This is bit 1 of the Pause substate. The Paused bit also disables the rest of the rung (RLL), it does not terminate or suspend the execution of the routine.

  This bit is only set by the phase's firmware. When the PauseEnabled bit is ON, execution of a PPD instruction causes the Paused bit to be set to Paused (1) and firmware pauses execution of the phase state routine suspends. A Resume command sets the Paused bit to not-Paused (0) and the phase executes its logic.

Available Languages

Ladder Diagram

Function Block

This instruction is not available in function block.

Structured Text

PPD( );

Operands

Ladder Diagram

None

Structured Text

None

Enter the parentheses ( ) after the instruction mnemonic, even though there are no operands.

Guidelines for using the PPD Instruction

Guideline	Details	
Organize logic as a series of steps.	PPD instructions (breakpoints) are easiest to use if the logic moves through defined steps, such as a state machine or SFC. • A breakpoint *only* signals that specific conditions are met. It *does not* stop the execution of the equipment phase. • To have logic actually break (pause) at a breakpoint, organize the logic so that it stays at the step at which the breakpoint occurred until it receives the resume command.	
Do not use a PPD instruction as a temporary end of the routine.	Even when an equipment phase pauses, it continues to execute all its logic. • When a PPD instruction executes, it only sets the Paused bit for the equipment phase. • If programing the PPD instruction in RLL, it disables only the rest of the logic on its rung. It *does not* terminate or suspend the execution of the routine. • Think of the PPD instruction as a condition that applies or is ignored based on the auto pause and pause commands.	
Use PPD instruction to pause at the same breakpoint over several scans.	When the PauseEnabled bit is TRUE, an equipment phase goes to Paused at the first PPD instruction whose conditions are true. If the PPD instruction executes over several scans, the equipment phase may continually pause at the same breakpoint.	
Make sure only 1 PPD instruction at a time is true.	A PPD instruction *does not* have a control tag to remember whether it executed. • Anytime its conditions are true (and the equipment phase is in a substate with PauseEnabled True), the PPD instruction acts as a breakpoint (and pauses the phase by disabling the rest of the logic on the rung). • Limiting logic to one possible breakpoint at a time ensures a pause at the required breakpoint.	
Choose the substate to use.	PPD instructions (breakpoints) work only when the equipment phase PauseEnabled bit is True.	
	To pause at:	**Give this command:**
	Each true breakpoint	Auto Pause
	First true breakpoint	Pause

Affects Math Status Flags

No

Major/Minor Faults

A major fault occurs if:	Fault type	Fault code
Instruction is called from outside an Equipment Phase program.	4	91

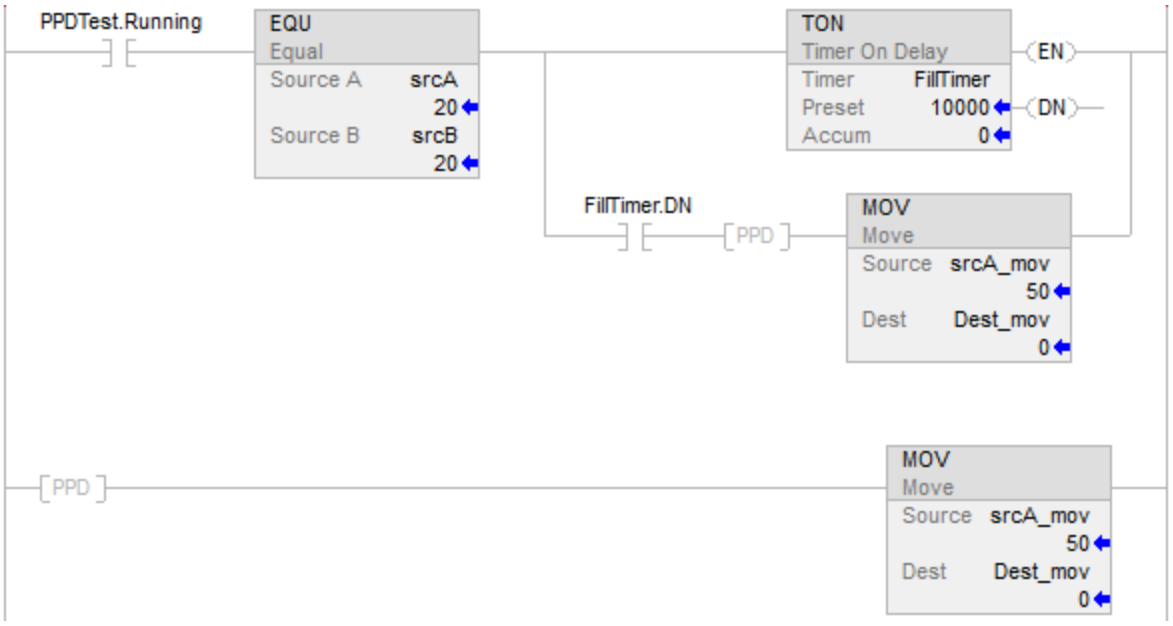
If an Add-On Instruction uses a PPD instruction, and a non-equipment phase program calls the Add-On Instruction, Logix Designer gives a warning. Check the Add-On Instruction for this instruction to disallow it. See *Index Through Arrays* for operand-related faults.

Execution

Condition/State	Action Taken
Prescan	No action taken.
Postscan	No action taken.
EnableIn is false	No action taken.
EnableIn is true	The instruction executes.

Example

Ladder Diagram



See also

Equipment Phase Instructions on page 969

Index Through Arrays on page 1094

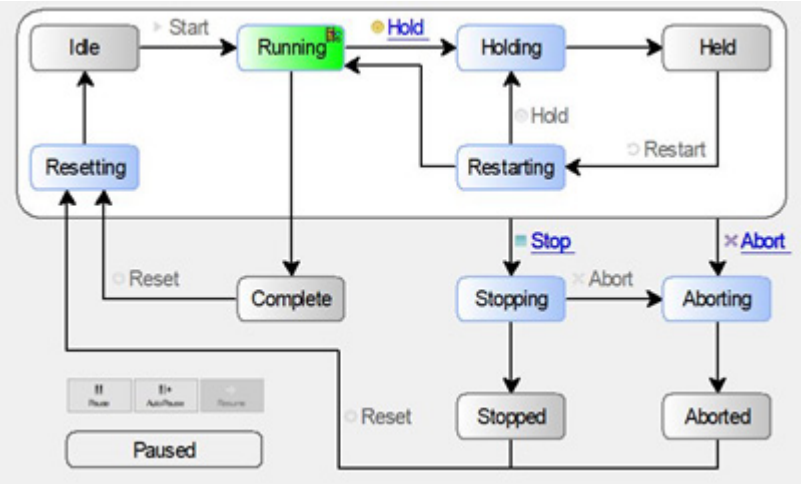
Equipment Phase Command on page 980

Phase State Complete (PSC)

This instruction applies to the CompactLogix 5370 and CompactLogix 5380, ControlLogix 5570 and ControlLogix 5580, Compact GuardLogix 5370 and Compact GuardLogix 5380, and GuardLogix 5570 and GuardLogix 5580 controllers.

Use the PSC instruction to signal an equipment phase that the state routine is complete to indicate that it should go to the next state.

The PSC instruction signals the completion of a phase state routine.



In the running state routine, use the PSC instruction to transition the equipment phase to the complete state.

Available Languages

Ladder Diagram

Function Block

This instruction is not available in function block.

Structured Text

PSC( );

Operands

Ladder Diagram

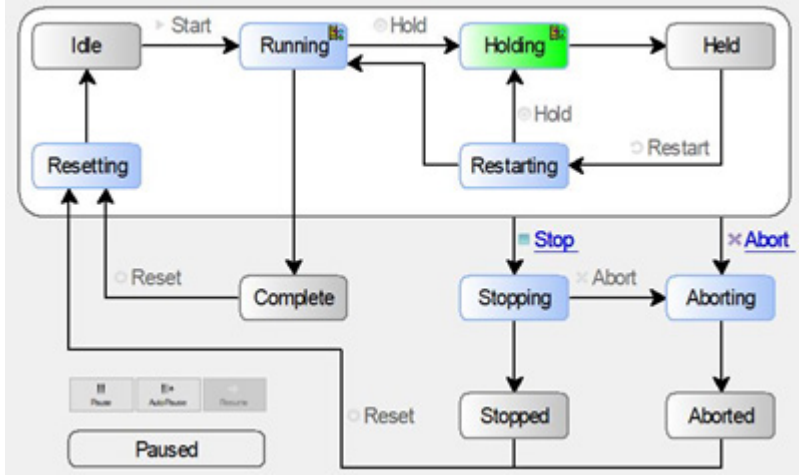
None

Structured Text

None

Enter the parentheses ( ) after the instruction mnemonic, even though there are no operands.

Guidelines for using the PSC Instruction

Guideline	Details
Use the PSC instruction in *each* phase state routine that is added to an equipment phase.	Without a PSC instruction, the equipment phase remains in the state and does *not* go to the next state. • Place the PSC instruction as the last step in the phase state routine. • When the state is done (complete), execute the PSC instruction.
	In the holding state routine, use the PSC instruction to let the equipment phase go to the Held state
Remember that the PSC instruction does *not* stop the current scan of a routine.	When the PSC instruction executes, the controller scans the rest of the routine and then transitions the equipment phase to the next state. The PSC instruction does not terminate the execution of the routine.
Do *not* use a PSC instruction in a prestate routine.	Use the PSC instruction only to signal the transition from one state to another.

Affects Math Status Flags

No

Major/Minor Faults

A major fault will occur if:	Fault type	Fault code
Instruction is called from outside an equipment phase program.	4	91

If an Add-On Instruction used a PSC instruction and a non-equipment phase program calls the Add-On Instruction, Logix Designer gives a warning. Check the Add-On Instruction for this instruction and disallow it. See *Index Through Arrays* for operand-related faults.

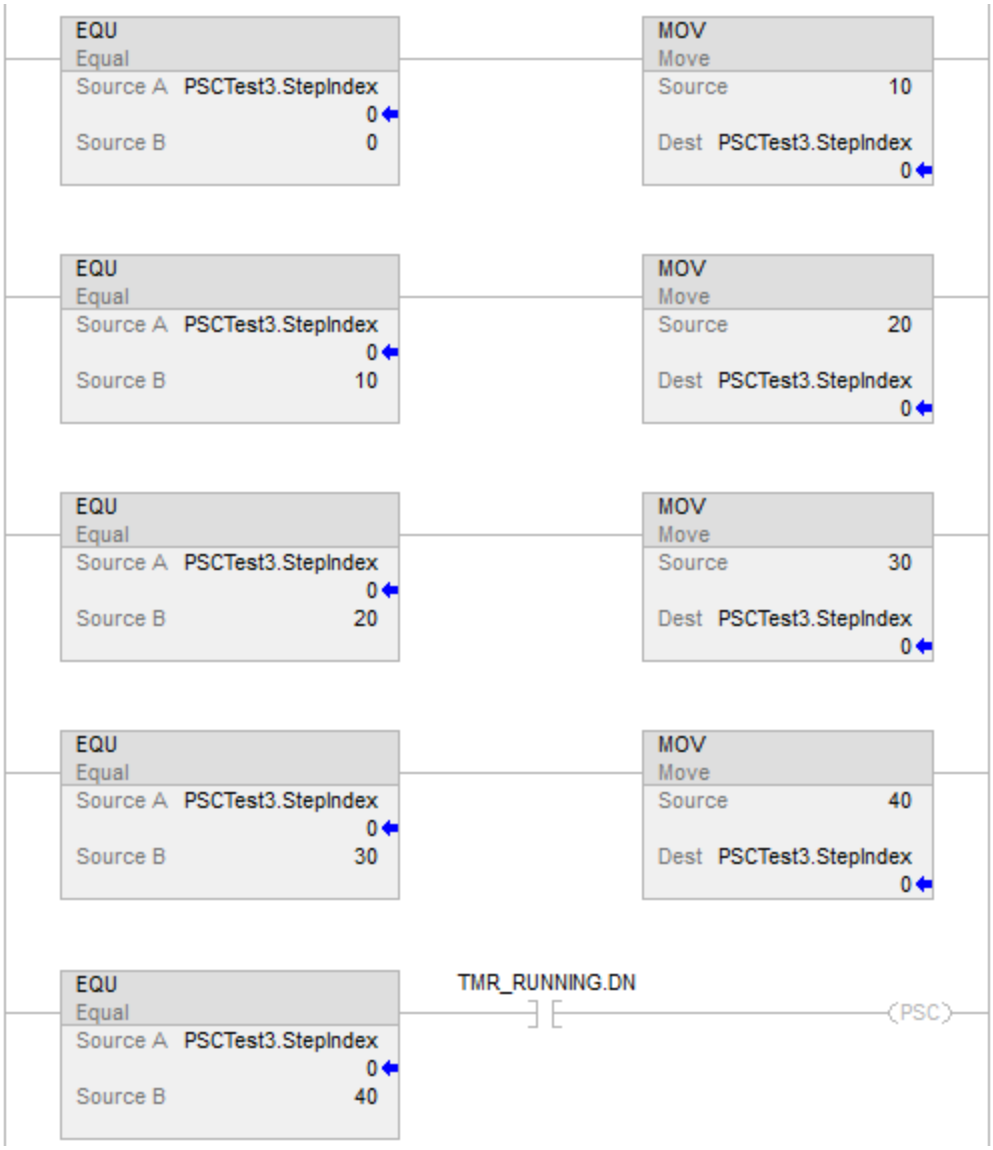
Execution

In structured text, instructions execute each time they are scanned. To limit the scan of an instruction, use a qualifier of an SFC action, a structured text construct, or both.

Condition/State	Action Taken
Prescan	No action taken.
Postscan	No action taken.
EnableIn is false	No action taken.
EnableIn is true	The instruction executes.

Examples

Ladder Diagram



Structured Text

If TagEnableRunning

And PSCTest.Running Then

PSC();

End_if;

See also

Equipment Phase Instructions on page 969

Index Through Arrays on page 1094

Equipment Phase Override Command (POVR) on page 1004

Equipment Sequence

Equipment Sequence instructions

The following table lists the command instructions for Equipment Sequences. The instructions are available for routines that use the Ladder Diagram and the Structured Text programming languages. They are not available for use in routines that use the Function Block and the Sequential Function Chart programming languages.

Instruction	Description	Structured Text format
Attach to Equipment Sequence (SATT)	Request that the parent program of the user routine be the owner of the Equipment Sequence.	SequenceName, Result
Detach from Equipment Sequence (SDET)	Release ownership of an Equipment Sequence.	SequenceName
Equipment Sequence Command (SCMD)	Send a command to the Equipment Sequence.	SequenceName, Command, Result
Equipment Sequence Clear Failure (SCLF)	Clear the failure code of an Equipment Sequence.	SequenceName
Equipment Sequence Override (SOVR)	Send a HOLD, STOP, or ABORT command to an Equipment Sequence, regardless of ownership.	SequenceName, Command, Result
Equipment Sequence Assign Sequence Identifier (SASI)	Assign an ID string to the Equipment Sequence.	SequenceName, Sequence ID, Result

See also

Equipment Sequence Diagram instructions on page 1031

SATT on page 1019, SDET on page 1021

SCMD on page 1027, SCLF on page 1025

SOVR on page 1031, SASI on page 1023

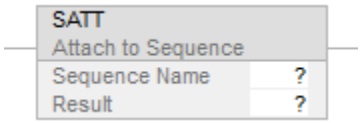
Attach to Equipment Sequence (SATT)

Use the Attach to Equipment Sequence (SATT) instruction to take ownership of an Equipment Sequence. An Equipment Sequence must be owned by a program before the program can command it. A tag must be assigned to store the result code of an SATT instruction.

The SATT instruction returns one of five result codes. Result code **0** indicates that the SATT instruction ran successfully. The other four codes indicate that the instruction did not run successfully and provide additional information about the reason for the instruction failure.

Available Languages

Ladder Diagram



Function Block

This language is not available for this instruction.

Structured Text:

SATT(SequenceName,Result)

Operands

The SATT instruction uses the following operands.

Operand	Type	Format	Description
Sequence Name	Sequence	name of the Equipment Sequence	Equipment Sequence that you want to change to own (attach) to command (for example, Make_Product_101).
Result	DINT	immediate tag	To let the instruction return a code for its success or failure, enter a DINT tag in which the result code is stored. Otherwise, enter **0**.

Affects Math Status Flags

No

Major/Minor Faults

The SATT instruction cannot trigger a fault, so there are no fault conditions for this instruction.

Execution

The following table describes the execution steps for SATT instructions.

Ladder Diagram

Condition	Action Taken
Prescan	The rung-condition-out is set to false.
Rung-condition-in is false	The rung-condition-out is set to false.
Rung-condition-in is true	• The instruction executes. • The rung-condition-out is set to true.
Scan of structured text	No action taken
Instruction execution	The instruction tries to take ownership of the specified Equipment Sequence.
Postscan	The rung-condition-out is set to false.

Structured Text

Condition	Action Taken
Prescan	No action taken.
Rung-condition-in is false	No action taken.
Rung-condition-in is true	No action taken.
Scan of structured text	In structured text, instructions execute each time they are scanned. To limit the scan of an instruction, use a qualifier of an SFC action or a structured text construct.
Instruction execution	The instruction tries to take ownership of the specified Equipment Sequence.
Postscan	No action taken.

See also

Guidelines for SATT instructions on page 1033

Result codes for SATT instructions on page 1036

SATT instruction examples on page 1039

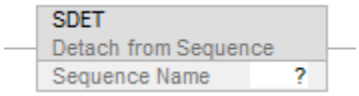
Equipment Sequence Instructions on page 1019

Detach from Equipment Sequence (SDET)

Use the Detach from Equipment Sequence (SDET) instruction to relinquish ownership of an Equipment Sequence. After a program executes an SDET instruction, the program no longer owns the Equipment Sequence. The Equipment Sequence is then available for ownership by another program or by FactoryTalk Batch software. Use the SDET instruction only if the program previously took ownership of an Equipment Sequence through an Attach to Equipment Sequence (SATT) instruction.

Available Languages

Ladder diagram



Function Block

This instruction is not available in Function Block.

Structured text

SDET(SequenceName)

Operands

The SDET instruction uses the following operand.

Operand	Type	Format	Description
Sequence Name	Sequence	name of the Equipment Sequence	Equipment Sequence for which you want to relinquish ownership.

Affects Math Status Flags

No

Major/Minor Faults

The SDET instruction cannot trigger a fault, so there are no fault conditions for this instruction.

Execution

Ladder Diagram

Condition	Action Taken
Prescan	The rung-condition-out is set to false.
Rung-condition-in is false	The rung-condition-out is set to false.

Condition	Action Taken
Rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.
Scan of structured text	No action taken
Instruction execution	The instruction tries to take ownership of the specified Equipment Sequence.
Postscan	The rung-condition-out is set to false.

Structured Text

Condition	Action Taken
Prescan	No action taken.
Rung-condition-in is false	No action taken.
Rung-condition-in is true	No action taken.
Scan of structured text	In structured text, instructions execute each time they are scanned. To limit the scan of an instruction, use a qualifier of an SFC action or a structured text construct that includes a condition, such as if, then, or else
Instruction execution	The instruction tries to take ownership of the specified Equipment Sequence.
Postscan	No action taken.

See also

SDET instruction examples on page 1041

Equipment Sequence Instructions on page 1019

Equipment Sequence Assign Sequence Identifier (SASI)

Use the Assign Sequence Identifier (SASI) instruction to assign a sequence ID to the Equipment Sequence. You can only set the sequence ID when the following prerequisites are met:

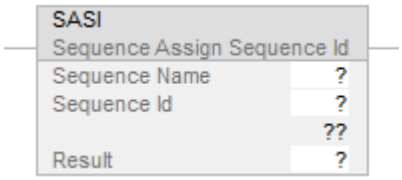
- The controller is online.
- The Equipment Sequence is in the IDLE state.
- You have taken ownership of the Equipment Sequence.

The sequence ID can be up to 82 characters in length, using the following printable ASCII characters:

a-z, A-Z, 0-9, !"#$%&'()+,-./:;<=>?@[\] ^_`{|}~ and space*

Available Languages

Ladder Diagram



Function Block

This instruction is not available in Function Block.

Structured Text

SASI(SequenceName,SequenceID,SequenceIDValue,Result)

Operands

The SASI instruction uses the following operands.

Operand	Type	Format	Description
Sequence Name	Sequence	name of the Equipment Sequence	Equipment Sequence to which you want to assign an identifier.
Sequence ID	STRING	tag	Enter a STRING tag in which the identifier is stored, or a quoted string containing up to 82 characters.
Result	DINT	immediate tag	To let the instruction return a code for its success or failure, enter a DINT tag in which the result code is stored. Otherwise, enter **0**.

Affects Math Status Flags

No

Major/Minor Faults

The SASI instruction cannot trigger a fault, so there are no fault conditions for this instruction.

Execution

Ladder Diagram

Condition	Action Taken
Prescan	The rung-condition-out is set to false.
Rung-condition-in is false	The rung-condition-out is set to false.
Rung-condition-in is true	• The instruction executes. • The rung-condition-out is set to true.
Scan of structured text	No action taken
Instruction execution	The instruction tries to take ownership of the specified Equipment Sequence.
Postscan	The rung-condition-out is set to false.

Structured Text

Condition	Action Taken
Prescan	No action taken.
Rung-condition-in is false	No action taken.
Rung-condition-in is true	No action taken.
Scan of structured text	In structured text, instructions execute each time they are scanned. To limit the scan of an instruction, use a qualifier of an SFC action or a structured text construct.
Instruction execution	The instruction tries to take ownership of the specified Equipment Sequence.
Postscan	No action taken.

See also

SASI instruction examples on page 1038

Equipment Sequence Instructions on page 1019

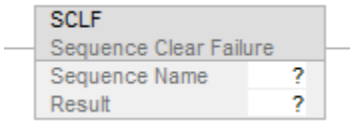
Equipment Sequence Clear Failure (SCLF)

Use the Equipment Sequence Clear Failure (SCLF) instruction to clear the failure code of an Equipment Sequence. Keep the following in mind when using the SCLF instruction.

- A CLR instruction, MOV instruction, or assignment does not change the failure code of an Equipment Sequence.
- The Equipment Sequence cannot have other owners when you use the SCLF instruction. The SCLF instruction does not clear the failure code if the Logix Designer application, FactoryTalk Batch software, or another program owns the Equipment Sequence.
- An Equipment Sequence refuses a RESUME command until it is cleared of failures.

Available Languages

Ladder diagram



Function Block

Not available

Structured text

SCLF(SequenceName)

Operands

The SCLF instruction uses the following operands.

Operand	Type	Format	Description
Sequence Name	Sequence	name of the Equipment Sequence	Equipment Sequence for which you want to clear a failure code.
Result	DINT	Immediate tag	For an instruction to return a success or failure code, enter a DINT tag where the result code is stored. Otherwise, enter **0**.

Affects Math Status Flags

No.

Major/Minor Faults

The SCLF instruction cannot trigger a fault, so there are no fault conditions for this instruction.

Execution

Ladder Diagram

Condition	Action Taken
Prescan	The rung-condition-out is set to false.
Rung-condition-in is false	The rung-condition-out is set to false.
Rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.
Scan of structured text	No action taken
Instruction execution	The instruction tries to take ownership of the specified Equipment Sequence.
Postscan	The rung-condition-out is set to false.

Structured Text

Condition	Action Taken
Prescan	No action taken.
Rung-condition-in is false	No action taken.
Rung-condition-in is true	No action taken.
Scan of structured text	In structured text, instructions execute each time they are scanned. To limit the scan of an instruction, use a qualifier of an SFC action or a structured text construct that includes a condition, such as if, then, or else
Instruction execution	The instruction tries to take ownership of the specified Equipment Sequence.
Postscan	No action taken.

See also

Result codes for SCLF instructions on page 1036

SDET instruction examples on page 1041

Equipment Sequence Instructions on page 1019

Equipment Sequence command (SCMD)

Use the Equipment Sequence command (SCMD) instruction to change the state of an Equipment Sequence. The SCMD instruction can send the following commands to an Equipment Sequence: START, RESTART, HOLD, STOP, ABORT, and RESET. The program must attach to an Equipment Sequence before the SCMD instruction can run. Use the SATT instruction to attach to an Equipment Sequence.

Like the SCMD instruction, the Equipment Sequence Override instruction (SOVR) also changes the state of an Equipment Sequence, but it changes the state regardless of ownership. If the SCMD instruction must execute

regardless of ownership, use an SOVR instruction instead of an SCMD instruction.

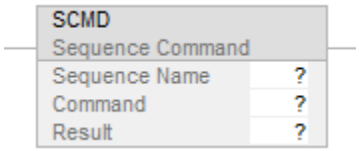
IMPORTANT The SOVR instruction is intended for emergencies only. Control Engineers should use caution when deciding to use it.

When a tag is assigned to store the result of an SCMD instruction, the instruction returns one of five result codes when it runs. Result code **0** indicates that the SCMD instruction ran successfully. The other four codes indicate that the instruction did not run successfully and provide additional information about the reason for the instruction failure.

Available Languages

The SCMD instruction is available in the following languages.

Ladder Diagram



Function Block

This instruction is not available in Function Block.

Structured Text

SCMD(SequenceName,Command,Result)

Supported Operands

The SCMD instruction uses the following operands.

Operand	Type	Format	Description
Sequence Name	Sequence	Name of the Equipment Sequence	Equipment Sequence to change to a different state. For example, Make_Product_101.
Command	Command	Name of the command	Command to send to the Equipment Sequence and change state. Send one of these commands: START, RESTART, HOLD, STOP, ABORT, or RESET.
Result	DINT	Immediate tag	For an instruction to return a success or failure code, enter a DINT tag where the result code is stored. Otherwise, enter **0**.

Valid command states for the SCMD instruction

Use the SCMD instruction transitions to command an Equipment Sequence to another state. The SCMD instruction commands may only be processed in certain states. The following table lists the states in which commands are valid.

Command	Valid in these states
START	Valid in the IDLE state.
RESTART	Valid in the HELD state.
HOLD	Valid in the RUNNING and RESTARTING states.
STOP	Valid in the RUNNING, HOLDING, RESTARTING, and HELD states.
ABORT	Valid in the RUNNING, HOLDING, RESTARTING, STOPPING, and HELD states.
RESET	Valid in the ABORTED, STOPPED, and COMPLETE states.

Arithmetic status flags and fault conditions

Arithmetic status flags are not affected by the SCMD instruction. The SCMD instruction cannot trigger a fault, so there are no fault conditions for this instruction.

Execution

Ladder Diagram

Condition	Action Taken
Prescan	The rung-condition-out is set to false.
Rung-condition-in is false	The rung-condition-out is set to false.
Rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.
Scan of structured text	No action taken
Instruction execution	The instruction tries to take ownership of the specified Equipment Sequence.
Postscan	The rung-condition-out is set to false.

Structured Text

Condition	Action Taken
Prescan	No action taken.
Rung-condition-in is false	No action taken.
Rung-condition-in is true	No action taken.
Scan of structured text	In structured text, instructions execute each time they are scanned. To limit the scan of an instruction, use a qualifier of an SFC action or a structured text construct.

Condition	Action Taken
Instruction execution	The instruction tries to take ownership of the specified Equipment Sequence.
Postscan	No action taken.

See also

SCMD instruction examples on page 1040

Use an SOVR instruction instead of an SCMD instruction on page 1042

Result codes for SCMD instructions on page 1037

Guidelines for SCMD instructions on page 1034

Equipment Sequence Diagram instructions

The following table describes the Equipment Sequence Diagram instructions.

Icon	Icon Name	Description
	Add Step and Transition Pair	Use **Add Step and Transition Pair** to add a step and transition pair. Although added as a pair, you can select and edit each element separately.
	Add Disconnected Step	Use the **Add Disconnected Step** to add a step without adding a transition.
	Add Disconnected Transition	Use **Add Disconnected Transition** to add a transition without adding a step.
	Add Simultaneous Divergence	Use **Add Simultaneous Divergence** to create a branch where all linked steps execute simultaneously.
	Add Selective Divergence	Use **Add Selective Divergence** to create a divergence for a selective branch. In a selective divergence, only one of multiple paths is executed--the path containing the transition that first evaluates as TRUE.
	Add Simultaneous Convergence	Use **Add Simultaneous Convergence** to merge simultaneous execution paths back together.
	Add Selective Convergence	Use **Add Selective Convergence** to merge selective divergent paths back into one execution path in the selective branch.

See also

Equipment Sequence instructions on page 1019

Equipment Sequence Override (SOVR)

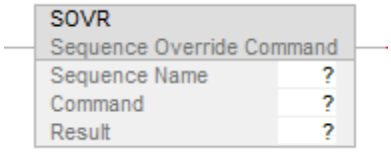
Use the Equipment Sequence Override (SOVR) instruction to send a HOLD, STOP, or ABORT command to an Equipment Sequence, regardless of ownership.

IMPORTANT The SOVR instruction is intended for emergencies only. Control Engineers should use caution when deciding to use it.

When a tag is assigned to store the result of an SOVR instruction, the instruction returns one of five result codes when it runs. Result code **0** indicates that the SOVR instruction ran successfully. The other four codes indicate that the instruction did not run successfully and provide additional information about the reason for the instruction failure.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in Function Block.

Structured Text

SOVR(SequenceName,Command,Result)

Operands

The SOVR instruction uses the following operands.

Operand	Type	Format	Description
Sequence Name	Sequence	Name of the Equipment Sequence	Equipment Sequence to change to a different state. For example, Make_Product_101.
Command	Command	Name of the command	Command to send to the Equipment Sequence and change state. Use one of these commands: • HOLD • STOP • ABORT
Result	DINT	Immediate tag	For an instruction to return a success or failure code, enter a DINT tag where the result code is stored. Otherwise, enter **0**.

Affects Math Status Flags

No.

Major/Minor Faults

The SOVR instruction cannot trigger a fault, so there are no fault conditions for this instruction.

Execution

Ladder Diagram

Condition	Action Taken
Prescan	The rung-condition-out is set to false.
Rung-condition-in is false	The rung-condition-out is set to false.
Rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.
Scan of structured text	No action taken
Instruction execution	The instruction commands the Equipment Sequence to the specified state.
Postscan	The rung-condition-out is set to false.

Structured Text

Condition	Action Taken
Prescan	No action taken.
Rung-condition-in is false	No action taken.
Rung-condition-in is true	No action taken.
Scan of structured text	In structured text, instructions execute each time they are scanned. To limit the scan of an instruction, use a qualifier of an SFC action or a structured text construct that includes a condition, such as if, then, or else.
Instruction execution	The instruction commands the Equipment Sequence to the specified state. The SOVR instruction sends one of the following commands: HOLD, STOP, or ABORT.
Postscan	No action taken.

See also

Guidelines for SOVR instructions on page 1035

Result codes for SOVR instructions on page 1038

SOVR instruction examples on page 1041

When should I use an SOVR instruction instead of an SCMD instruction? on page 1042

Guidelines for SATT instructions

Keep the following guidelines in mind when using the Attach to Equipment Sequence (SATT) instruction.

Guideline	Details
Remember that the Logix Designer application overrides ownership of the Equipment Sequence.	Ownership makes sure that a program can command the Equipment Sequence, and it locks out any other sequencers.

The Equipment Sequence must be owned by the program to command it.	Both Equipment Sequences and Equipment Phases must be *owned* to be commanded. The ownership commands are **Attach** (SATT) and **Detach** (SDET). Internal sequencers (programs), external sequencers (FactoryTalk Batch), and operators use an **Attach** instruction to command an Equipment Sequence.
When the sequence is done, relinquish ownership.	To relinquish ownership, use a Detach from Equipment Sequence (SDET) instruction.
Avoid making unnecessary command requests if the equipment sequence is generating sequence events.	Unnecessary command requests can flood the event processing buffers and cause you to miss significant events.
Use the **Result** code to verify ownership, and include steps that should take place if the attachment fails because the Equipment Sequence is owned by another program or by the operator.	Use the **Result** operand to get a code that shows the success or failure of the SATT instruction. On each execution, the SATT instruction tries to take ownership of the Equipment Sequence. When a program or operator owns an Equipment Sequence, another execution of the SATT instruction fails and produces result code 24582. When you use the SATT instruction, either: • Limit its execution to a single scan to avoid the 24582 result code. • Include the following in your conditions for ownership: result code = 24582.

See also

Attach to Equipment Sequence on page 1019

Result codes for SATT instructions on page 1036

Equipment Sequence Instructions on page 1019

Guidelines for SCMD instructions

Keep the following guidelines in mind when using the Equipment Sequence Command (SCMD) instruction. The SCMD instruction can send the following commands: START, RESTART, HOLD, STOP, ABORT, and RESET.

Guideline	Details
Limit execution of the SCMD instruction to a single scan.	Limit the execution of the SCMD instruction to a single scan. Each command applies to only a specific state or states. Once the Equipment Sequence changes state, the command is no longer valid. To limit execution, use methods such as: • Run the SCMD instruction within a P1 Pulse (Rising Edge) or P0 Pulse (Falling Edge) action. • Place a one-shot instruction before the SCMD instruction. • Run the SCMD instruction and then advance to the next step.
The Equipment Sequence must be owned by the program to command it.	Both Equipment Sequences and Equipment Phases must be *owned* to be commanded. The ownership commands are **Attach** (SATT) and **Detach** (SDET). Internal sequencers (programs), external sequencers (FactoryTalk Batch), and operators use an **Attach** instruction to command an Equipment Sequence.

See also

Equipment Sequence command (SCMD) on page 1027

Equipment Sequence Instructions on page 1019

When should I use an SOVR instruction instead of an SCMD instruction? on page 1042

Guidelines for SOVR instructions

Use the Equipment Sequence Override (SOVR) instruction to send a HOLD, STOP, or ABORT command to an Equipment Sequence, regardless of ownership.

IMPORTANT The SOVR instruction is intended for emergencies only. Control Engineers should use caution when deciding to use it.

Keep the following guidelines in mind when using the Equipment Sequence Override (SOVR) instruction.

Guideline	Details
Make sure you want to override other owners.	Under most circumstances, use the SCMD instruction to programmatically command an Equipment Sequence. However, use the SOVR instruction to command an Equipment Sequence under the following conditions: • When you are giving the HOLD, STOP, or ABORT command, and the command must always execute under all ownership circumstances. • If the HOLD, STOP, or ABORT command must execute even when you have manual control of the Equipment Sequence through the Logix Designer application or when another program, such as the FactoryTalk Batch software, owns the Equipment Sequence.
Limit execution of the SOVR instruction to a single scan.	Limit the execution of the SOVR instruction to a single scan. Each command applies to only a specific state or states. Once the Equipment Sequence changes state, the command is no longer valid. To limit execution, use methods such as: • Run the SOVR instruction within a P1 Pulse (Rising Edge) or P0 Pulse (Falling Edge) action. • Place a one-shot instruction before the SOVR instruction. • Run the SOVR instruction and then advance to the next step.
Avoid making unnecessary command requests if the Equipment Sequence is generating sequence events.	Unnecessary command requests can flood the event processing buffers and cause you to miss significant events.

See also

Equipment Sequence Override command on page 1031

Result codes for SOVR instructions on page 1038

When should I use an SOVR instruction instead of an SCMD instruction? on page 1042

Result codes for SATT instructions

When a tag is assigned to store the result of an Attach to Equipment Sequence (SATT) instruction, the instruction returns one of the following codes when it runs.

Code (Dec)	Description
0	The command was successful.
24579	The Equipment Sequence cannot be commanded for the following reason: The program successfully attached to the Equipment Sequence, but it cannot command the sequence because Logix Designer, a higher priority application, has overridden ownership.
24582	The program already owns the Equipment Sequence.
24593	One of the following already owns the equipment phase. • An external sequencer such as FactoryTalk Batch software. • Another program in the controller.
24594	The Equipment Sequence is unscheduled, inhibited, or in a task that is inhibited.

Use the **Result** operand to get a code that shows the success or failure of the SATT instruction. The **Result** operand should contain either **0** or a DINT tag, depending on whether ownership conflicts or other errors are likely to occur.

- If ownership conflicts or other errors are not likely, enter **0** in the **Result** operand.
- If ownership conflicts or other errors are likely, enter a DINT tag in the **Result** operand. The DINT tag stores a code for the result of the execution of the instruction.

See also

Attach to Equipment Sequence on page 1019

Equipment Sequence Instructions on page 1019

Result codes for SCLF instructions

When a tag is assigned to store the result of an Equipment Sequence Clear Failure (SCLF) instruction, the instruction returns one of the following codes when it executes.

Code (Dec)	Description
0	The command was successful.
48	The command was not executed because it was not possible at the time to generate an event to record the command. • If the command was an ABORT command, the ABORT command is still executed even if the event could not be generated. • This code only occurs if event generation has been enabled in the **Equipment Sequence Properties - Configuration** tab.
24578	The command is not valid for the current state of the Equipment Sequence. For example, if the Equipment Sequence is stopped, then a stop command is not valid.
24594	The Equipment Sequence is unscheduled, inhibited, or in a task that is inhibited.

Use the **Result** operand to get a code that shows the success or failure of the SCLF instruction. The **Result** operand should contain either **0** or a DINT tag, depending on whether ownership conflicts or other errors are likely to occur.

- If ownership conflicts or other errors are not likely, enter **0** in the **Result** operand.
- If ownership conflicts or other errors are likely, enter a DINT tag in the **Result** operand. The DINT tag stores a code for the result of the execution of the instruction.

See also

Equipment Sequence Clear Failure on page 1025

Equipment Sequence Instructions on page 1019

Result codes for SCMD instructions

When a tag is assigned to store the result of an Equipment Sequence command (SCMD) instruction, the instruction returns one of the following codes when it runs.

Code (Dec)	Description
0	The command was successful.
48	The command was not executed because it was not possible at the time to generate an event to record the command. • If the command was an ABORT command, the ABORT command is still executed even if the event could not be generated. • This code only occurs if event generation has been enabled in the **Equipment Sequence Properties - Configuration** tab.
24577	The command is not valid.
24578	The command is not valid for the current state of the Equipment Sequence. For example, if the Equipment Sequence is in the running state, then a start command is not valid.
24579	The Equipment Sequence cannot be commanded for the following reason: • The program successfully attached to the Equipment Sequence, but it cannot command the sequence because Logix Designer, a higher priority application, has overridden ownership.
24582	Attachment to the Equipment Sequence failed because the sequence was previously attached to one of the following users: • An external sequencer, such as FactoryTalk Batch software, has ownership. • Another program in the controller (an internal sequencer) has ownership. • An operator using the Sequence Manager ActiveX Controls has ownership.
24580	The caller of the instruction is attached, but is not the current owner of the Equipment Sequence. A higher priority owner, such as Logix Designer, is commanding the Equipment Sequence.
24594	The Equipment Sequence is unscheduled, inhibited, or in a task that is inhibited.
24604	An equal or higher priority command is being processed.
24631	Too many sequence parameter or step tags are defined per step, so events cannot be handled and the START command failed.

Use the **Result** operand to get a code that shows the success or failure of the SCMD instruction. The **Result** operand should contain either **0** or a DINT tag, depending on whether ownership conflicts or other errors are likely to occur.

- If ownership conflicts or other errors are not likely, enter **0** in the **Result** operand.

- If ownership conflicts or other errors are likely, enter a DINT tag in the **Result** operand. The DINT tag stores a code for the result of the execution of the instruction.

See also

Equipment Sequence command on page 1027

Equipment Sequence Instructions on page 1019

Result codes for SOVR instructions

When a tag is assigned to store the result of an Equipment Sequence Override (SOVR) instruction, the instruction returns one of the following codes when it executes.

Code (Dec)	Description
0	The command was successful.
48	The command was not executed because it was not possible at the time to generate an event to record the command. • If the command was an ABORT command, the ABORT command is still executed even if the event could not be generated. • This code only occurs if event generation has been enabled in the **Equipment Sequence Properties - Configuration** tab.
24577	The command is not valid.
24578	The command is not valid for the current state of the Equipment Sequence. For example, if the Equipment Sequence is stopped, then a stop command is not valid.
24594	The Equipment Sequence is unscheduled, inhibited, or in a task that is inhibited.

Use the **Result** operand to get a code that shows the success or failure of the SOVR instruction. The **Result** operand should contain either **0** or a DINT tag, depending on whether ownership conflicts or other errors are likely to occur.

- If ownership conflicts or other errors are not likely, enter **0** in the **Result** operand.
- If ownership conflicts or other errors are likely, enter a DINT tag in the **Result** operand. The DINT tag stores a code for the result of the execution of the instruction.

See also

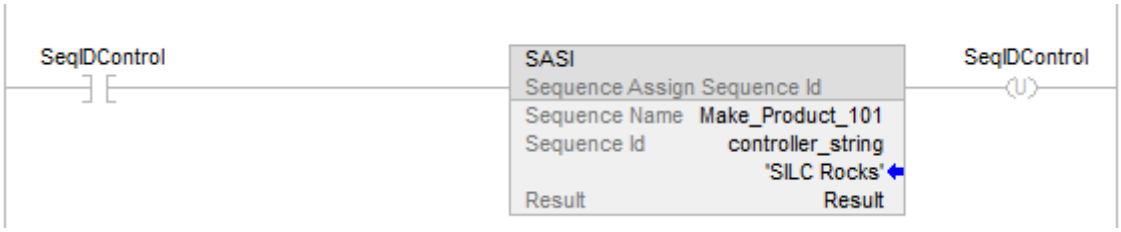
Equipment Sequence Override command on page 1031

Equipment Sequence Instructions on page 1019

SASI instruction examples

The following example shows the SASI instruction as it appears in a ladder diagram and in structured text.

Ladder Diagram Example



Tip: The Sequence ID parameter can be a STRING tag in which the identifier is stored, or a quoted string containing up to 82 characters.

Structured Text Example

```
if (IdControl) then
    SASI (Make_Product_101, NewId, Results);
end_if;
```

See also

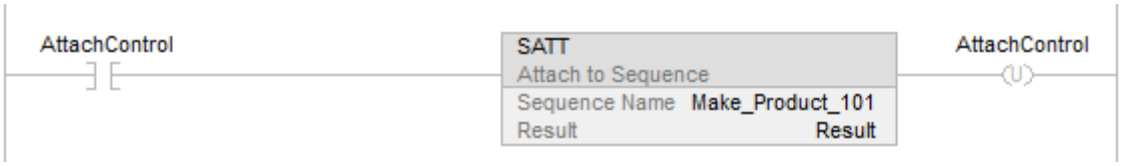
Equipment Sequence Assign Sequence Identifier on page 1023

Equipment Sequence Instructions on page 1019

SATT instruction examples

The following examples show the SATT instruction as it appears in a Ladder Diagram and in Structured Text.

Ladder Diagram



Structured Text

```
if (AttachControl) then
    SATT (Make_Product_101, Result);
end_if;
```

See also

Attach to Equipment Sequence on page 1019

Guidelines for SATT instructions on page 1033

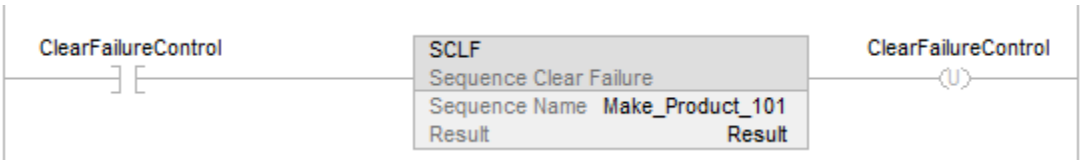
Result codes for SATT instructions on page 1036

Equipment Sequence Instructions on page 1019

SCLF instruction examples

The following examples show the SCLF instruction as it appears in a ladder diagram and in structured text.

Ladder Diagram Example



Structured Text Example

```
if (ClearFailureControl) then
    SCLF (Make_Product_101);
end_if;
```

See also

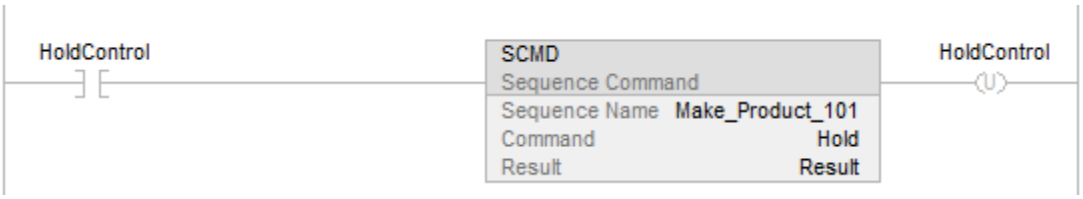
Equipment Sequence Clear Failure on page 1025

Equipment Sequence Instructions on page 1019

SCMD instruction examples

The following examples show the Equipment Sequence command (SCMD) instruction as it appears in a Ladder Diagram and in Structured Text.

Ladder Diagram



Structured Text

```
if (HoldControl) then
    SCMD (Make_Product_101), Hold, Result);
end_if;
```

See also

Equipment Sequence command on page 1027

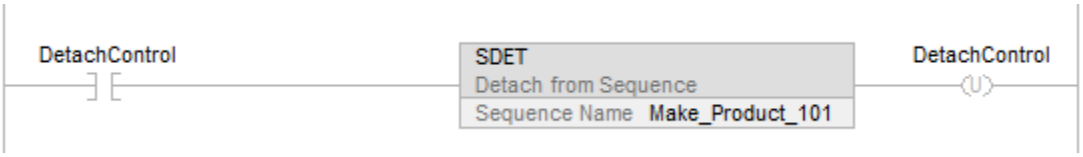
Guidelines for SCMD instructions on page 1034

Equipment Sequence Instructions on page 1019

SDET instruction examples

The following examples show the SDET instruction as it appears in a ladder diagram and in structured text.

Ladder Diagram



Structured Text

```
if (DetachControl) then
    SDET (Make_Product_101);
end_if;
```

See also

Detach from Equipment Sequence on page 1021

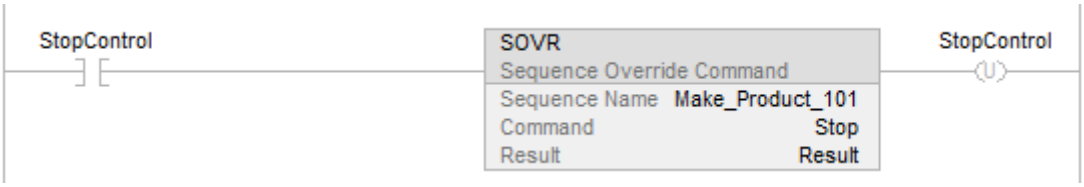
Attach to Equipment Sequence on page 1019

Equipment Sequence instructions on page 1019

SOVR instruction examples

The following examples show the SOVR instruction as it appears in a Ladder Diagram and in Structured Text.

Ladder Diagram



Structured Text

```
if (StopControl) then
    SOVR (Make_Product_101, Stop, Results);
```

```
end_if;
```

See also

Equipment Sequence command on page 1027

Guidelines for SCMD instructions on page 1034

Equipment Sequence Instructions on page 1019

When should I use an SOVR instruction instead of an SCMD instruction?

Under most circumstances, use the SCMD instruction to programmatically command an Equipment Sequence. However, use the SOVR instruction to command an Equipment Sequence under the following conditions:

- When you are giving the HOLD, STOP, or ABORT command, and the command must always execute under all ownership circumstances.
- If the HOLD, STOP, or ABORT command must execute even when you have manual control of the Equipment Sequence through the Logix Designer application or when another program, such as the FactoryTalk Batch software, owns the Equipment Sequence.

For example, suppose your equipment checks for jammed material. If there is a jam, you always want the equipment to abort. In that case, use the SOVR instruction. This way, the equipment aborts even if you have manual control through the Logix Designer application.

See also

Equipment Sequence command on page 1027

Equipment Sequence Override command on page 1031

Guidelines for SCMD instructions on page 1034

Equipment Sequence Instructions on page 1019

Function Block Attributes

Click a topic below for more information on issues that are unique to function block programming. Review this information to make sure you understand how your function block routines will operate.

See also

Choose the Function Block Elements on page 1043

Latching Data on page 1044

Order of Execution on page 1046

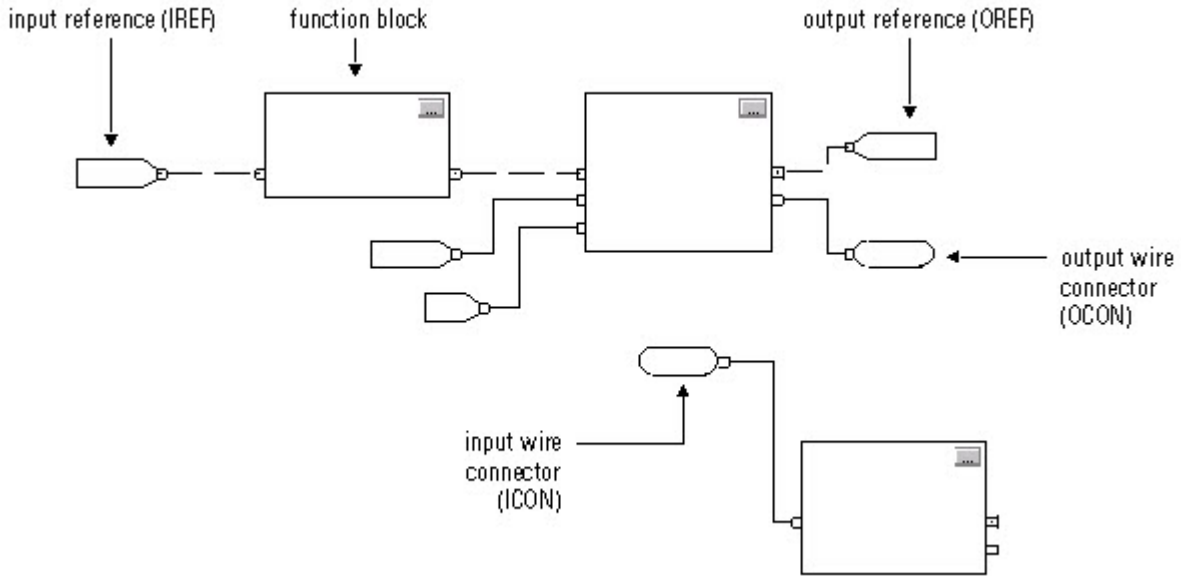
Function Block Responses to Overflow Conditions on page 1045

Timing Modes on page 1050

Program/Operator Control on page 1052

Choose the Function Block Elements

To control a device, use these elements:



Use the following table to help you choose your function block elements:

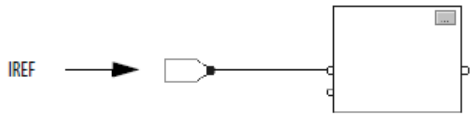
If you want to supply a value from an input device or tag	Then use a input reference (IREF)
Send a value to an output device or tag	Output reference (OREF)

If you want to supply a value from an input device or tag	Then use a input reference (IREF)
Perform an operation on an input value or values and produce an output value or values.	Function block
Transfer data between function blocks when they are: • Far apart on the same sheet • On different sheets within the same routine	Output wire connector (OCON) and an input wire connector (ICON)
Disperse data to several points in the routine	Single output wire connector (OCON) and multiple input wire connectors (ICON)

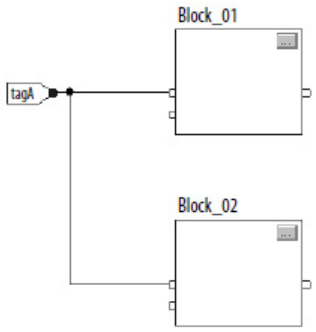
The function block moves the input references into the block structure. If necessary, the function block converts those input references to REAL values. The function block executes and moves the results into the output references. Again, if necessary, the function block converts those result values from REAL to the data types for the output references.

Latching Data

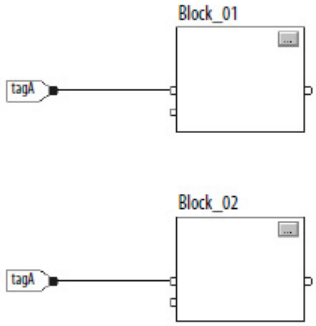
If you use an IREF to specify input data for a function block instruction, the data in that IREF is latched for the scan of the function block routine. The IREF latches data from program-scoped and controller-scoped tags. The controller updates all IREF data at the beginning of each scan.



In this example, the value of tagA is stored at the beginning of the routine's execution. The stored value is used when Block_01 executes. The same stored value is also used when Blcock_02 executes. If the value of tagA changes during execution of the routine, the stored value of tagA in the IREF does not change until the next execution of the routine.

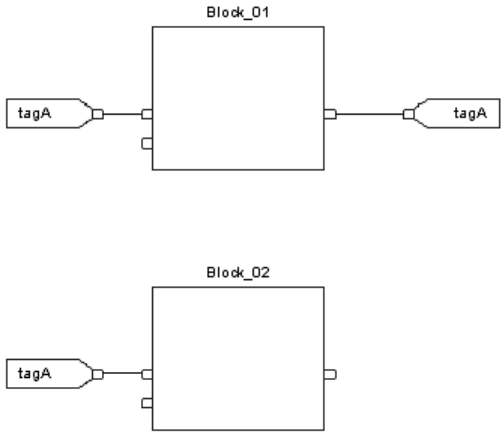


This example is the same as the one above. The value of tagA is stored only once at the beginning of the routine's execution. The routine uses this stored value throughout the routine.



You can use the same tag in multiple IREFs and an OREF in the same routine. Because the values of tags in IREFs are latched every scan through the routine, all IREFs will use the same value, even if an OREF obtains a different tag value during execution of the routine.

In this example, if tagA has a value of 25.4 when the routine starts executing this scan, and Block_01 changes the value of tagA to 50.9, the second IREF wired into Block_02 will still use a value of 25.4 when Block_02 executes this scan. The new tagA value of 50.9 will not be used by any IREFs in this routine until the start of the next scan.



Function Block Responses to Overflow Conditions

In general, the function block instructions that maintain history do not update history with ± NAN, or ±INF values when an overflow occurs. Each instruction has one of these responses to an overflow condition.

Response	Instruction
Response 1 Blocks execute their algorithm and check the result for ±NAN or ±INF. If ±NAN or ±INF, the block outputs ±NAN or ±INF.	ALM NTCH DEDT PMUL DERV POSP ESEL RLIM FGEN RMPS HPF SCRV LDL2 SEL LDLG SNEG LPF SRTP MAVE SSUM MAXC TOT MINC UPDN MSTD MUX
Response 2 Blocks with output limiting execute their algorithm and check the result for ±NAN or ±INF. The output limits are defined by the HighLimit and LowLimit input parameters. If ±INF, the block outputs a limited result. If ±NAN, the output limits are not used and the block outputs ±NAN.	HLL, INTG, PI, PIDE, SCL, SOC
Response 3 The overflow condition does not apply. These instructions typically have a boolean output.	BAND, BNOT, BOR, BXOR, CUTD, D2SD, D3SD, DFF, JKFF, OSFI, OSRI, RESD, RTOR, SETD, TOFR, TONR

Order of Execution

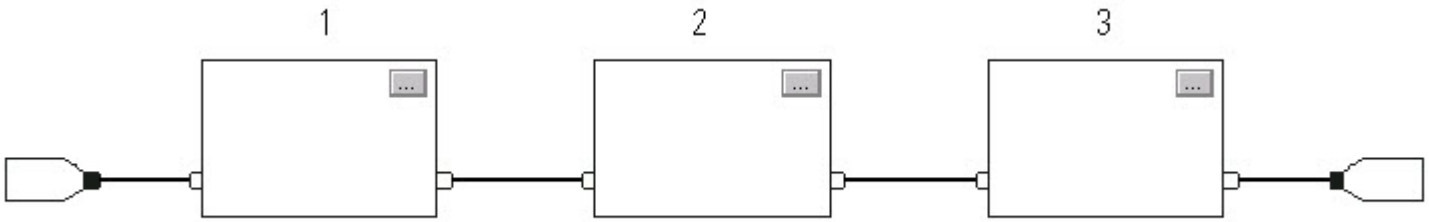
The Logix Designer programming application automatically determines the order of execution for the function blocks in a routine when you:

- verify a function block routine
- verify a project that contains a function block routine
- download a project that contains a function block routine

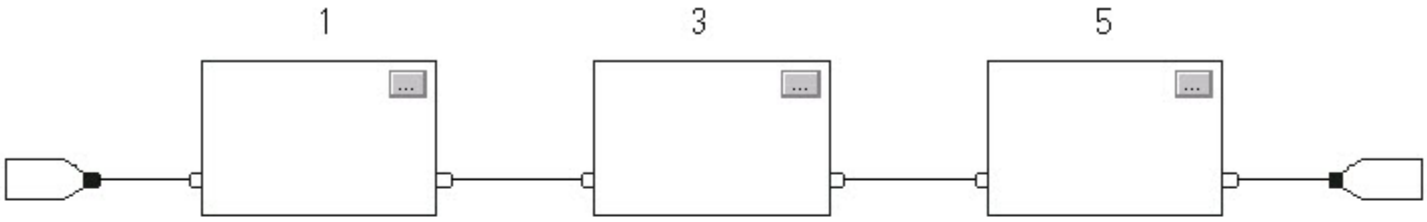
You define execution order by wiring function blocks together and indicating the data flow of any feedback wires, if necessary.

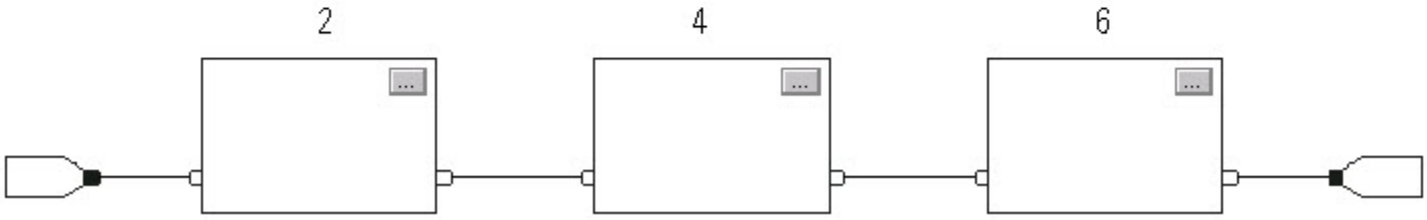
If function blocks are not wired together, it does not matter which block executes first. There is no data flow between the blocks

If you wire the blocks sequentially, the execution order moves from input to output. The inputs of a block require data to be available before the controller can execute that block. For example, block 2 has to execute before block 3 because the outputs of block 2 feed the inputs of block 3.



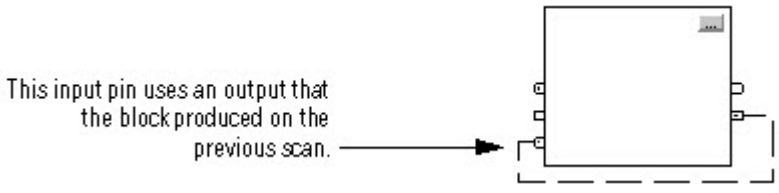
Execution order is only relative to the blocks that are wired together. The following example is fine because the two groups of blocks are not wired together. The blocks within a specific group execute in the appropriate order in relation to the blocks in that group.



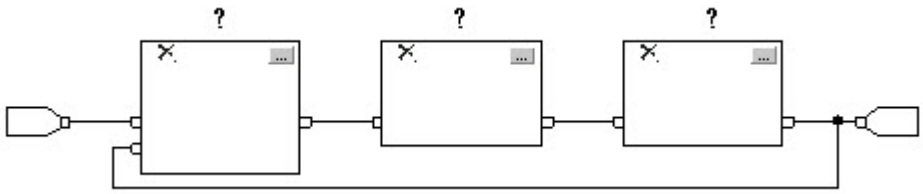


Resolve a Loop

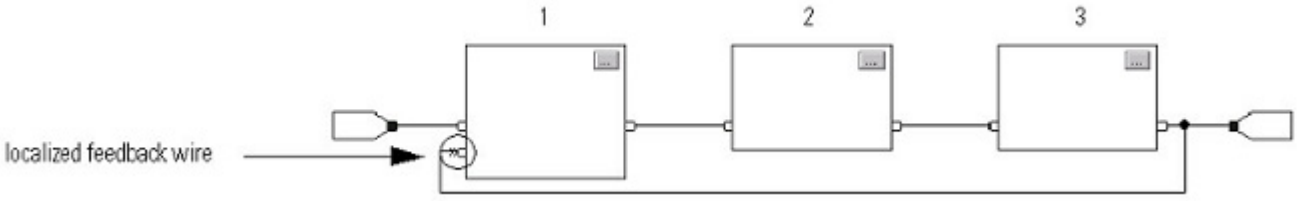
To create a feedback loop around a block, wire an output pin of the block to an input pin of the same block. The following example is OK. The loop contains only a single block, so execution order does not matter.



If a group of blocks are in a loop, the controller cannot determine which block to execute first. In other words, it cannot resolve the loop.

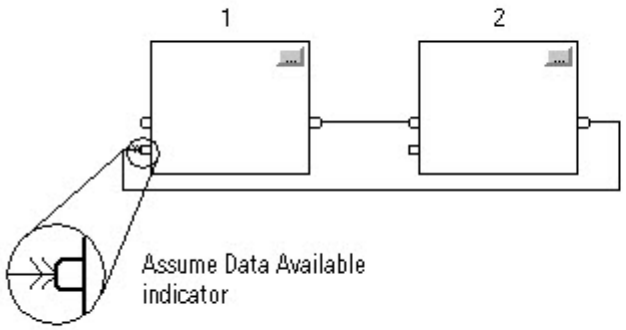
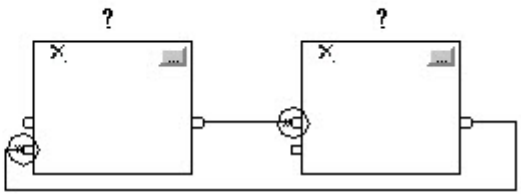


To identify which block to execute first, mark the input wire that creates the loop (the feedback wire) with the *Assume Data Available* indicator. In the following example, block 1 uses the output from block 3 that was produced in the previous execution of the routine.



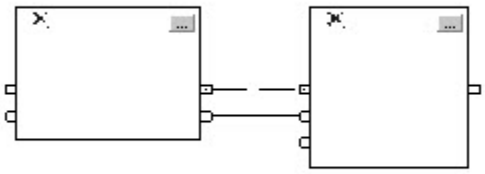
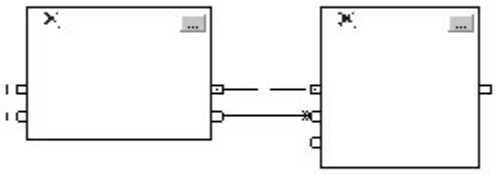
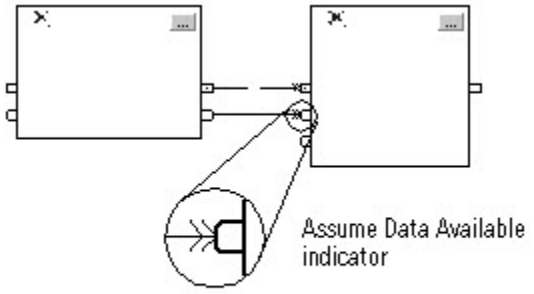
The *Assume Data Available* indicator defines the data flow within the loop. The arrow indicates that the data serves as input to the first block in the loop.

Do *not* mark all the wires of a loop with the *Assume Data Available* indicator.

This is OK	This is NOT OK
 The *Assume Data Available* indicator defines the data flow within the loop.	 The controller cannot resolve the loop because all the wires use the *Assume Data Available* indicator.

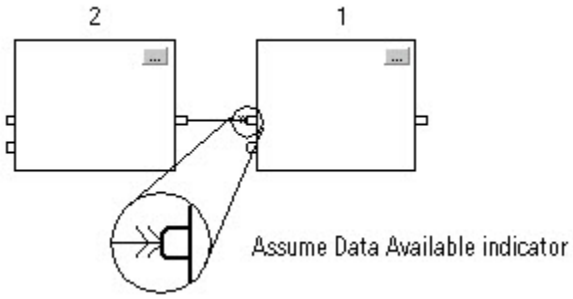
Resolve Data Flow Between Two Blocks

If you use two or more wires to connect two blocks, use the same data flow indicators for all of the wires between the two blocks.

This is OK	This is NOT OK
 Neither wire uses the *Assume Data Available* indicator.	 One wire uses the *Assume Data Available* indicator while the other wire does not.
 Both wires use the *Assume Data Available* indicator.	

Create a One Scan Delay

To produce a one scan delay between blocks, use the Assume Data Available indicator. In the following example, block 1 executes first. It uses the output from block 2 that was produced in the previous scan of the routine.



Summary

In summary, a function block routine executes in this order:

1. The controller latches all data values in IREFs.
2. The controller executes the other function blocks in the order determined by how they are wired.
3. The controller writes outputs in OREFs.

Timing Modes

These process control and drives instructions support different timing modes.

- DEDT
- DERV
- HPF
- INTG
- LDL2
- LDLG
- LPF
- NTCH
- PI
- PIDE
- RLIM
- SCRV
- SOC
- TOT

There are three different timing modes.

Timing Mode	Description	
Periodic	Periodic mode is the default mode and is suitable for most control applications. We recommend that you place the instructions that use this mode in a routine that executes in a periodic task. The delta time (DeltaT) for the instruction is determined as follows:	
	If the instruction executes in a	**Then DeltaT equals**
	Periodic task	Period of the task
	Event or continuous task	Elapsed time since the previous execution The controller truncates the elapsed time to whole milliseconds (ms). For example, if the elapsed time = 10.5 ms, the controller sets DeltaT = 10 ms.
	The update of the process input needs to be synchronized with the execution of the task or sampled 5-10 times faster than the task executes in order to minimize the sampling error between the input and the instruction.	
Oversample	In oversample mode, the delta time (DeltaT) used by the instruction is the value written into the OversampleDT parameter of the instruction. If the process input has a time stamp value, use the real time sampling mode instead. Add logic to your program to control when the instruction executes. For example, you can use a timer set to the OversampleDeltaT value to control the execution by using the EnableIn input of the instruction. The process input needs to be sampled 5-10 times faster than the instruction is executed in order to minimize the sampling error between the input and the instruction.	
Real time sampling	In the real time sampling mode, the delta time (DeltaT) used by the instruction is the difference between two time stamp values that correspond to the updates of the process input. Use this mode when the process input has a time stamp associated with its updates and you need precise coordination. The time stamp value is read from the tag name entered for the RTSTimeStamp parameter of the instruction. Normally this tag name is a parameter on the input module associated with the process input. The instruction compares the configured RTSTime value (expected update period) against the calculated DeltaT to determine if every update of the process input is being read by the instruction. If DeltaT is not within 1 millisecond of the configuration time, the instruction sets the RTSMissed status bit to indicate that a problem exists reading updates for the input on the module.	

Time-based instructions require a constant value for DeltaT in order for the control algorithm to properly calculate the process output. If DeltaT varies, a discontinuity occurs in the process output. The severity of the discontinuity depends on the instruction and range over which DeltaT varies.

A discontinuity occurs if the following happens:

- Instruction is not executed during a scan.
- Instruction is executed multiple times during a task.

- Task is running and the task scan rate or the sample time of the process input changes.
- User changes the time-base mode while the task is running.
- Order parameter is changed on a filter block while the task is running.
- Changing the Order parameter selects a different control algorithm within the instruction.

Common Instruction Parameters for Timing Modes

The instructions that support time-base modes have these input and output parameters.

Input Parameters

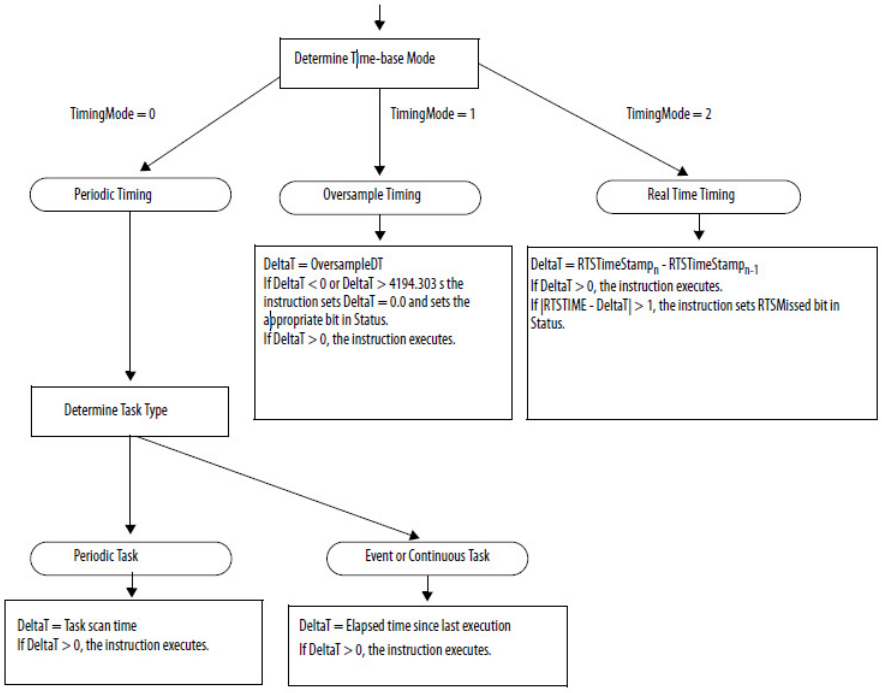
Input Parameter	Data Type	Description
TimingMode	DINT	Selects timing execution mode. Value: Description: 0 Periodic mode 1 Oversample mode 2 Real time sampling mode Valid = 0 to 2 Default = 0 When TimingMode = 0 and task is periodic, periodic timing is enabled and DeltaT is set to the task scan rate. When TimingMode = 0 and task is event or continuous, periodic timing is enabled and DeltaT is set equal to the elapsed time span since the last time the instruction was executed. When TimingMode = 1, oversample timing is enabled and DeltaT is set to the value of the OversampleDT parameter. When TimingMode = 2, real time sampling timing is enabled and DeltaT is the difference between the current and previous time stamp values read from the module associated with the input. If TimingMode invalid, the instruction sets the appropriate bit in Status.
OversampleDT	REAL	Execution time for oversample timing. The value used for DeltaT is in seconds. If TimingMode = 1, then OversampleDT = 0.0 disables the execution of the control algorithm. If invalid, the instruction sets DeltaT = 0.0 and sets the appropriate bit in Status. Valid = 0 to 4194.303 seconds Default = 0.0
RTSTime	DINT	Module update period for real time sampling timing. The expected DeltaT update period is in milliseconds. The update period is normally the value that was used to configure the module's update time. If invalid, the instruction sets the appropriate bit in Status and disables RTSMissed checking. Valid = 1...32,767ms Default = 1
RTSTimeStamp	DINT	Module time stamp value for real time sampling timing. The time stamp value that corresponds to the last update of the input signal. This value is used to calculate DeltaT. If invalid, the instruction sets the appropriate bit in Status, disables execution of the control algorithm, and disables RTSMissed checking. Valid =0...32,767ms (wraps from 32767 to 0) 1 count = 1 millisecond Default = 0

Output Parameters

Output Parameter	Data Type	Description
DeltaT	REAL	Elapsed time between updates. This is the elapsed time in seconds used by the control algorithm to calculate the process output. Periodic: DeltaT = task scan rate if task is Periodic task, DeltaT = elapsed time since previous instruction execution if task is Event or Continuous task Oversample: DeltaT = OversampleDT Real Time Sampling: DeltaT = (RTSTimeStampn - RTSTimeStampn-1)
Status	DINT	Status of the function block.
TimingModeInv (Status.27)	BOOL	Invalid TimingMode value.
RTSMissed (Status.28)	BOOL	Only used in real time sampling mode. Set when ABS | DeltaT - RTSTime | > 1 (.001 second).
RTSTimeInv (Status.29)	BOOL	Invalid RTSTime value.
RTSTimeStampInv (Status.30)	BOOL	Invalid RTSTimeStamp value.
DeltaTInv (Status.31)	BOOL	Invalid DeltaT value.

Overview of Timing Modes

The following diagram shows how an instruction determines the appropriate timing mode.



Program/Operator Control

The following instructions support the concept of Program/Operator control.

- Enhanced Select (ESEL)
- Totalizer (TOT)
- Enhanced PID (PIDE)

- Ramp/Soak (RMPS)
- Discrete 2-State Device (D2SD)
- Discrete 3-State Device (D3SD)

Program/Operator control lets you control these instructions simultaneously from both your user program and from an operator interface device. When in Program control, the instruction is controlled by the Program inputs to the instruction; when in Operator control, the instruction is controlled by the Operator inputs to the instruction.

Program or Operator control is determined by using these inputs.

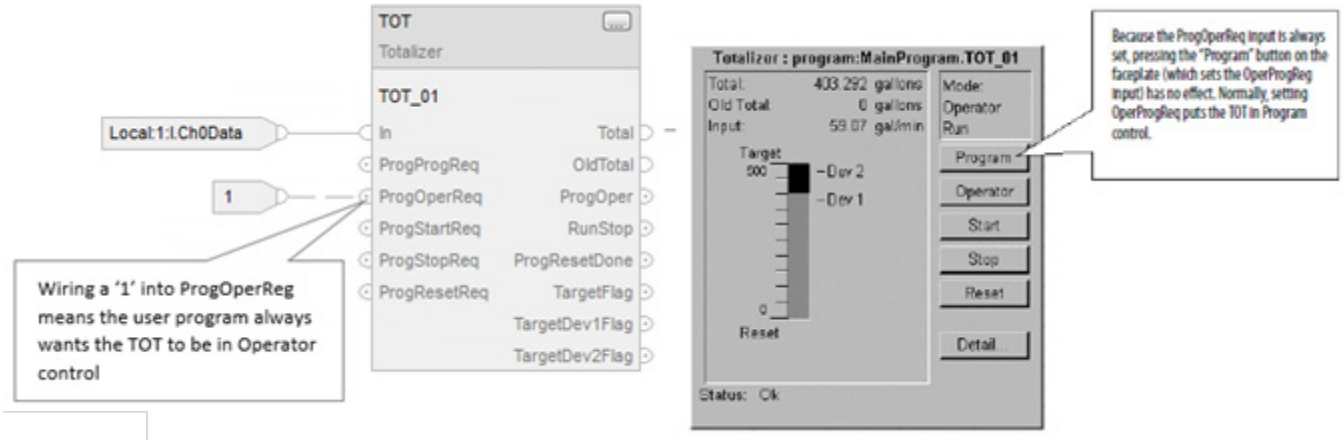
Input	Description
.ProgProgReq	A program request to go to Program control.
.ProgOperReq	A program request to go to Operator control.
.OperProgReq	An operator request to go to Program control.
.OperOperReq	An operator request to go to Operator control.

To determine whether an instruction is in Program or Operator control, examine the ProgOper output. If ProgOper is set, the instruction is in Program control; if ProgOper is cleared, the instruction is in Operator control.

Operator control takes precedence over Program control if both input request bits are set. For example, if ProgProgReq and ProgOperReq are both set, the instruction goes to Operator control.

The Program request inputs take precedence over the Operator request inputs. This provides the capability to use the ProgProgReq and ProgOperReq inputs to 'lock' an instruction in a desired control.

For example, let's assume that a Totalizer instruction will always be used in Operator control, and your user program will never control the running or stopping of the Totalizer. In this case, you could wire a literal value of 1 into the ProgOperReq. This would prevent the operator from ever putting the Totalizer into Program control by setting the OperProgReq from an operator interface device.

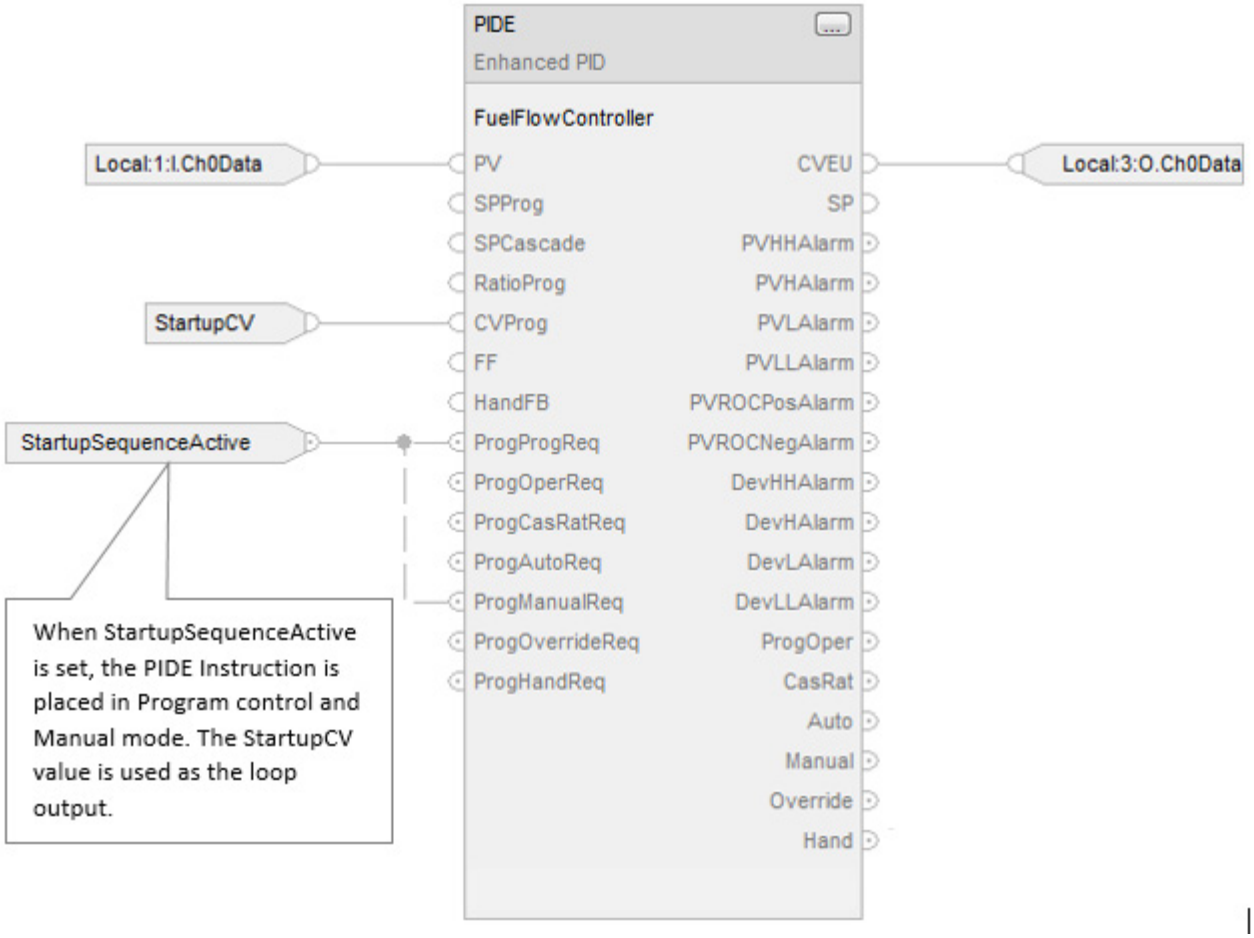


Likewise, constantly setting the ProgProgReq can 'lock' the instruction into Program control. This is useful for automatic startup sequences when you

want the program to control the action of the instruction without worrying about an operator inadvertently taking control of the instruction.

In this example, you have the program set the ProgProgReq input during the startup, and then clear the ProgProgReq input once the startup was complete. Once the ProgProgReq input is cleared, the instruction remains in Program control until it receives a request to change. For example, the operator could set the OperOperReq input from a faceplate to take over control of that instruction.

The following example shows how to lock an instruction into Program control.



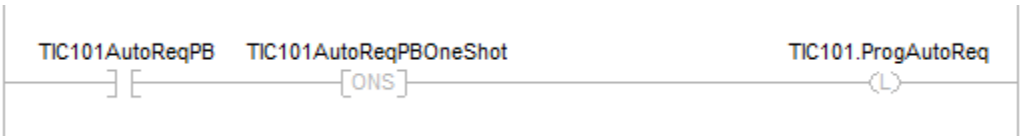
Operator request inputs to an instruction are always cleared by the instruction when it executes. This allows operator interfaces to work with these instructions by merely setting the desired mode request bit. You don't have to program the operator interface to reset the request bits. For example, if an operator interface sets the OperAutoReq input to a PIDE instruction, when the PIDE instruction executes, it determines what the appropriate response should be and clears the OperAutoReq.

Program request inputs are not normally cleared by the instruction because these are normally wired as inputs into the instruction. If the instruction clears these inputs, the input would just get set again by the wired input.

There might be situations where you want to use other logic to set the Program requests in such a manner that you want the Program requests to be cleared by the instruction. In this case, you can set the ProgValueReset input and the instruction will always clear the Program mode request inputs when it executes.

In this example, a rung of ladder logic in another routine is used to one-shot latch a ProgAutoReq to a PIDE instruction when a push button is pushed.

When the TIC101AutoReq push button is pressed, one-shot latch ProgAutoReq for the PIDE instruction TIC101. TIC101 has been configured with the ProgValueReset input set. ProgAutoReq get reset because ProgValuieReset is always set.



Function Block States

Logix-based controllers evaluate function block instructions based on the state of different conditions:

Condition	Description
prescan	Prescan for function block routines is the same as for ladder diagram routines. The only difference is that the EnableIn parameter for each function block instruction is cleared during prescan.
instruction first scan	Instruction first scan refers to the first time an instruction is executed after prescan. The controller uses instruction first scan to read current inputs and determine the appropriate state to be in.
instruction first run	Instruction first run refers to the first time the instruction executes with a new instance of a data structure. The controller uses instruction first run to generate coefficients and other data stores that do not change for a function block after initial download.

Every function block instruction also included EnableIn and EnableOut parameters:

- Function block instructions execute normally when EnableIn is set.
- When EnableIn is cleared, the function block instruction either executes prescan logic, postscan logic, or simply skips normal algorithm execution.
- EnableOut mirrors EnableIn. However, if Function Block detects an overflow condition, EnableOut is also cleared.
- Function Block resumes from where it left off when EnableIn toggles from cleared to set. However, there are some function block instructions that specify special functionality (for example, re-initialization) when EnableIn toggles from cleared to set. For function block instructions with time base parameters, whenever the timing mode is Oversample, the instruction always resumes from where it left off when EnableIn toggles from cleared to set.

If the EnableIn parameter is not wired, the instruction always executes as normal and EnableIn remains set. If you clear EnableIn, it changes to set the next time the instruction executes.

Structured Text Programming

These are the issues that are unique with structured text programming. Review the following topics to make sure you understand how your structured text programming executes.

Structured Text Syntax on page 1057

Structured Text Components: Comments on page 1058

Structured Text Components: Assignments on page 1059

Structured Text Components: Expressions on page 1061

Structured Text Components: Instructions on page 1066

Structured Text Components: Constructs on page 1067

CASE...OF on page 1069

FOR...DO on page 1071

IF...THEN on page 1074

REPEAT_UNTIL on page 1077

WHILE_DO on page 1080

Structured Text Syntax

Structured text is a textual programming language that uses statements to define what to execute.

- Structured text is not case sensitive.
- Use tabs and carriage returns (separate lines) to make your structured text easier to read. They have no effect on the execution of the structured text.

Structured text is not case sensitive. Structured text can contain these components.

Term	Definition	Examples
Assignment	Use an assignment statement to assign values to tags. The := operator is the assignment operator. Terminate the assignment with a semi colon ';.'	tag := expression;
Expression	An expression is part of a complete assignment or construct statement. An expression evaluates to a number (numerical expression), a String (string expression), or to a true or false state (BOOL expression)	
Tag Expression	A named area of the memory where data is stored (BOOL, SINT, INT, DINT, REAL, String).	value1
Immediate Expression	A constant value	4

Term	Definition	Examples
Operators Expression	A symbol or mnemonic that specifies an operation within an expression.	tag1 + tag2 tag1 >= value1
Function Expression	When executed, a function yields one value. Use parentheses to contain the operand of a function. Even though their syntax is similar, functions differ from instructions in that functions can be used only in expressions. Instructions cannot be used in expressions.	function(tag1)

Instruction	An instruction is a standalone statement.
	An instruction uses parentheses to contain its operands.
	Depending on the instruction, there can be zero, one, or multiple operands.
	When executed, an instruction yields one or more values that are part of a data structure. Terminate the instruction with a semi colon(;).
	Even though their syntax is similar, instructions differ from functions in that instructions cannot be used in expressions. Functions can be used only in expressions.
		instruction();
		
		instruction(operand);
		
		instruction(operand1, operand2,operand3);

Construct	A conditional statement used to trigger structured text code (that is, other statements). Terminate the construct with a semi colon (;).	IF...THEN CASE FOR...DO WHILE...DO REPEAT...UNTIL EXIT
Comment	Text that explains or clarifies what a section of structured text does. Use comments to make it easier to interpret the structured text. Comments do not affect the execution of the structured text. Comments can appear anywhere in structured text.	//comment (*start of comment . . . end of comment*) /*start of comment . . . end of comment*/

See also

Structured Text Components: Assignments on page 1059

Structured Text Components: Expressions on page 1061

Structured Text Components: Instructions on page 1066

Structured Text Components: Constructs on page 1067

Structured Text Components: Comments on page 1058

Structured Text Components: Comments

To make your structured text easier to interpret, add comments to it.

- Comments let you use plain language to describe how your structured text works.
- Comments do not affect the execution of the structured text.

To add comments to your structured text:

To add a comment	Use one of these formats
on a single line	//comment (*comment*) /*comment*/
at the end of a line of structured text	
within a line of structured text	(*comment*) /*comment*/

that spans more than one line	(*start of comment. . .end of comment*) /*start of comment. . .end of comment*/

For example:

Format	Example

//comment	**At the beginning of a line**
	//Check conveyor belt direction
	IF conveyor_direction THEN...
	At the end of a line
	ELSE //If conveyor isn't moving, set alarm light
	light := 1;
	END_IF;

(*comment*)	Sugar.Inlet[:=]1;(*open the inlet*) IF Sugar.Low (*low level LS*)& Sugar.High (*high level LS*)THEN... (*Controls the speed of the recirculation pump. The speed depends on the temperature in the tank.*) IF tank.temp > 200 THEN...
/*comment*/	Sugar.Inlet:=0;/*close the inlet*/ IF bar_code=65 /*A*/ THEN... /*Gets the number of elements in the Inventory array and stores the value in the Inventory_Items tag*/ SIZE(Inventory,0,Inventory_Items);

Structured Text Components: Assignments

Use an assignment to change the value stored within a tag. An assignment has this syntax:

tag := expression;

where:

Component	Description	
Tag	Represents the tag that is getting the new value; the tag must be a BOOL, SINT, INT, DINT, STRING, or REAL. **Tip:** The STRING tag is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.	
:=	Is the assignment symbol	
Expression	Represents the new value to assign to the tag	
	If tag is this data type	**Use this type of expression**
	BOOL	BOOL
	SINT INT DINT REAL	Numeric
	STRING (CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only).	String type, including string tag and string literal (CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only).
;	Ends the assignment	

The tag retains the assigned value until another assignment changes the value.

The expression can be simple, such as an immediate value or another tag name, or the expression can be complex and include several operators and functions, or both. Refer to Expressions for more information.

Tip: I/O module data updates asynchronously to the execution of logic. If you reference an input multiple times in your logic, the input could change state between separate references. If you need the input to have the same state for each reference, buffer the input value and reference that buffer tag. For more information, see Logix 5000 Controllers Common Procedures, publication 1756-PM001.
You can also use Input and Output program parameters which automatically buffer the data during logix execution. See LOGIX 5000 Controllers Program Parameters Programming Manual, publication 1756-PM021.

See also

Assign an ASCII character to a string data member on page 1061

Specify a non-retentive assignment on page 1060

Structured Text Components: Expressions on page 1061

Character string literals on page 1068

Specify a non-retentive assignment

The non-retentive assignment is different from the regular assignment described above in that the tag in a non-retentive assignment is reset to zero each time the controller:

- Enters the Run mode
- Leaves the step of an SFC if you configure the SFC for Automatic reset. This applies only if you embed the assignment in the action of the step or use the action to call a structured text routine by using a JSR instruction.

A non-retentive assignment has this syntax:

tag [:=] expression ;

where:

Component	Description	
tag	Represents the tag that is getting the new value; the tag must be a BOOL, SINT, INT, DINT, STRING, or REAL. **Tip:** The STRING tag is applicable toCompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.	
[:=]	Is the non-retentive assignment symbol.	
expression	Represents the new value to assign to the tag.	
	If tag is this data type	**Use this type of expression**
	BOOL	BOOL
	SINT	Numeric

Component	Description	
	INT	
	DINT	
	REAL	
	STRING (CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only).	String type, including string tag and string literal CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers(only)

See also

Assign an ASCII character to a string data member on page 1061

Structured Text Components: Assignments on page 1059

Assign an ASCII character to a string data member

Assign an ASCII character to a string data member

Use the assignment operator to assign an ASCII character to an element of the DATA member of a string tag. To assign a character, specify the value of the character or specify the tag name, DATA member, and element of the character. For example:

This is OK	This is not OK
string1.DATA[0] := 65;	string1.DATA[0] := A;
string1.DATA[0]:= string2.DATA[0];	string1 := string2; **Tip:** This assigns all content of string2 to string1 instead of just one character.

To add or insert a string of characters to a string tag, use either of these ASCII string instructions:

To	Use this instruction
Add characters to the end of a string	CONCAT
Insert characters into a string	INSERT

See also

Structured Text Components: Expressions on page 1061

Character string literals on page 1068

Structured Text Components: Expressions

An expression is a tag name, equation, or comparison. To write an expression, use any of the following:

- Tag name that stores the value (variable)
- Number that you enter directly into the expression (immediate value)

- String literal that you enter directly into the expression (CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only)
- Functions, such as: ABS, TRUNC
- Operators, such as: +, -, <, >, And, Or

Follow these guidelines for writing expressions:

- Use any combination of upper-case and lower-case letter. For example, these variations of "AND" are acceptable: AND, And, and.
- For more complex requirements, use parentheses to group expressions within expressions. This makes the whole expression easier to read, and ensures that the expression executes in the desired sequence.

Use these expressions for structured text:

BOOL expression: An expression that produces the BOOL value of 1 (true) or 0 (false).

- A bool expression uses bool tags, relational operators, and logical operators to compare values or check if conditions are true or false. For example, tag1>65.
- A simple bool expression can be a single BOOL tag.
- Typically, use bool expressions to condition the execution of other logic.

Numeric expression: An expression that calculates an integer or floating-point value.

- A numeric expression uses arithmetic operators, arithmetic functions, and bitwise operators. For example, tag1+5.
- Nest a numeric expression within a BOOL expression. For example, (tag1+5)>65.

String expression: An expression that represents a string

- A simple expression can be a string literal or a string tag

Use this table to select the operators for expressions.

If	Use
Calculating an arithmetic value	Arithmetic operators and functions
Comparing two values or strings	Relational operators
Verifying if conditions are true or false	Logical operators
Comparing the bits within values	Bitwise operators

See also

Use arithmetic operators and functions on page 1063

Use relational operators on page 1065

Use logical operators on page 1064

Use bitwise operators on page 1064

Use arithmetic operators and functions

Combine multiple operators and functions in arithmetic expressions.

Operators calculate new values.

To	Use this operator	Optimal data type
Add	+	DINT, REAL
Subtract/negate	-	DINT, REAL
Multiply	*	DINT, REAL
Exponent (x to the power of y)	**	DINT, REAL
Divide	/	DINT, REAL
Modulo-divide	MOD	DINT, REAL

Functions perform math operations. Specify a constant, a non-Boolean tag, or an expression for the function.

For	Use this function	Optimal data type
Absolute value	ABS (numeric_expression)	DINT, REAL
Arc cosine	ACOS (numeric_expression)	REAL
Arc sine	ASIN (numeric_expression)	REAL
Arc tangent	ATAN (numeric_expression)	REAL
Cosine	COS (numeric_expression)	REAL
Radians to degrees	DEG (numeric_expression)	DINT, REAL
Natural log	LN (numeric_expression)	REAL
Log base 10	LOG (numeric_expression)	REAL
Degrees to radians	RAD (numeric_expression)	DINT, REAL
Sine	SIN (numeric_expression)	REAL
Square root	SQRT (numeric_expression)	DINT, REAL
Tangent	TAN (numeric_expression)	REAL
Truncate	TRUNC (numeric_expression)	DINT, REAL

The table provides examples for using arithmetic operators and functions.

Use this format	Example	
	For this situation	Write
value1 operator value2	If gain_4 and gain_4_adj are DINT tags and your specification says: 'Add 15 to gain_4 and store the result in gain_4_adj'"	gain_4_adj := gain_4+15;
operator value1	If alarm and high_alarm are DINT tags and your specification says: 'Negate high_alarm and store the result in alarm.'	alarm:= -high_alarm;
function(numeric_expression)	If overtravel and overtravel_POS are DINT tags and your specification says: 'Calculate the absolute value of overtravel and store the result in overtravel_POS.'	overtravel_POS := ABS(overtravel);
value1 operator (function((value2+value3)/2)	If adjustment and position are DINT tags and sensor1 and sensor2 are REAL tags and your specification says: 'Find the absolute value of the average of sensor1 and sensor2, add the adjustment, and store the result in position.'	position := adjustment + ABS((sensor1 + sensor2)/2);

See also

Structured Text Components: Expressions on page 1061

Use bitwise operators

Bitwise operators manipulate the bits within a value based on two values.

The following provides an overview of the bitwise operators.

For	Use this operator	Optimal data type
bitwise AND	&, AND	DINT
bitwise OR	OR	DINT
bitwise exclusive OR	XOR	DINT
bitwise complement	NOT	DINT

This is an example.

Use this format	Example	
	For this situation	Use
value1 operator value2	If input1, input2, and result1 are DINT tags and your specification says: "Calculate the bitwise result of input1 and input2. Store the result in result1."	result1 := input1 AND input2;

See also

Structured Text Components: Expressions on page 1061

Use logical operators

Use logical operators to verify if multiple conditions are true or false. The result of a logical operation is a BOOL value.

If the comparison is	The result is
true	1
false	0

Use these logical operators.

For this comparison	Use this operator	Optimal data type
logical AND	&, AND	BOOL
logical OR	OR	BOOL
logical exclusive OR	XOR	BOOL
logical complement	NOT	BOOL

The table provides examples of using logical operators.

Use this format	Example	
	For this situation	Use
BOOLtag	If photoeye is a BOOL tag and your specification says: "If photoeye_1 is on then..."	IF photoeye THEN...
NOT BOOLtag	If photoeye is a BOOL tag and your specification says: "If photoeye is off then..."	IF NOT photoeye THEN...
expression1 & expression2	If photoeye is a BOOL tag, temp is a DINT tag, and your specification says: "If photoeye is on and temp is less than 100 then..."	IF photoeye & (temp<100) THEN...
expression1 OR expression2	If photoeye is a BOOL tag, temp is a DINT tag, and your specification says: "If photoeye is on or temp is less than 100 then...".	IF photoeye OR (temp<100) THEN...

expression1 XOR expression2	If photoeye1 and photoeye2 are BOOL tags and your specification says: "If: photoeye1 is on while photoeye2 is off or photoeye1 is off while photoeye2 is on then..."	IF photoeye1 XOR photoeye2 THEN...
BOOLtag := expression1 & expression2	If photoeye1 and photoeye2 are BOOL tags, open is a BOOL tag, and your specification says: "If photoeye1 and photoeye2 are both on, set open to true"	open := photoeye1 & photoeye2;

See also

Structured Text Components: Expressions on page 1061

Use relational operators

Relational operators compare two values or strings to provide a true or false result. The result of a relational operation is a BOOL value.

If the comparison is	The result is
True	1
False	0

Use these relational operators.

For this comparison	Use this operator	Optimal data type
Equal	=	DINT, REAL, String type
Less than	<	DINT, REAL, String type
Less than or equal	<=	DINT, REAL, String type
Greater than	>	DINT, REAL, String type
Greater than or equal	>=	DINT, REAL, String type
Not equal	<>	DINT, REAL, String type

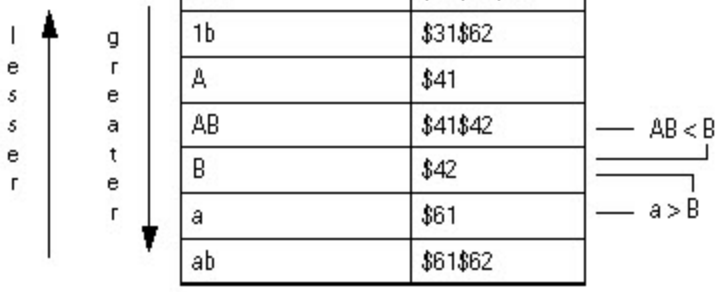
The table provides examples of using relational operators

Use this format	Example	
	For this situation	Write
value1 operator value2	If temp is a DINT tag and your specification says: 'If temp is less than 100· then...'	IF temp<100 THEN...
stringtag1 operator stringtag2	If bar_code and dest are string tags and your specification says: 'If bar_code equals dest then...'	IF bar_code=dest THEN...
stringtag1 operator 'character string literal'	If bar_code is a string tag and your specification says: 'If bar_code equals 'Test PASSED' then...'	IF bar_code='Test PASSED' THEN...
char1 operator char2 To enter an ASCII character directly into the expression, enter the decimal value of the character.	If bar_code is a string tag and your specification says: 'If bar_code.DATA[0] equals 'A' then...'	IF bar_code.DATA[0]=65 THEN...
bool_tag := bool_expressions	If count and length are DINT tags, done is a BOOL tag, and your specification says: 'If count is greater than or equal to length, you are done counting.'	Done := (count >= length);

How strings are evaluated

The hexadecimal values of the ASCII characters determine if one string is less than or greater than another string.

- When the two strings are sorted as in a telephone directory, the order of the strings determines which one is greater.



ASCII Characters	Hex Codes
1ab	$31$61$62
1b	$31$62
A	$41
AB	$41$42
B	$42
a	$61
ab	$61$62

- Strings are equal if their characters match.
- Characters are case sensitive. Upper case "A" ($41) is not equal to lower case "a" ($61).

See also

Structured Text Components: Expressions on page 1061

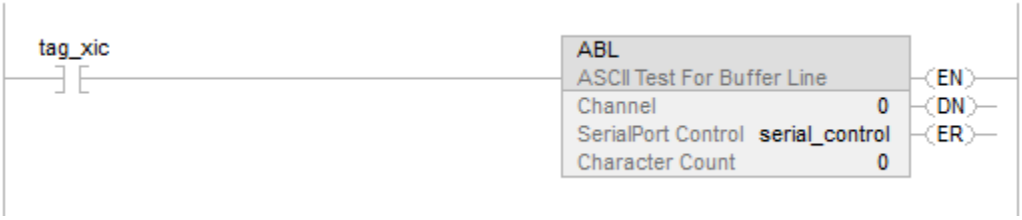
Structured Text Components: Instructions

Structured text statements can also be instructions. A structured text instruction executes each time it is scanned. A structured text instruction within a construct executes every time the conditions of the construct are true. If the conditions of the construct are false, the statements within the construct are not scanned. There is no rung-condition or state transition that triggers execution.

This differs from function block instructions that use EnableIn to trigger execution. Structured text instructions execute as if EnableIn is always set.

This also differs from ladder diagram instructions that use rung-condition-in to trigger execution. Some ladder diagram instructions only execute when rung- condition-in toggles from false to true. These are transitional ladder diagram instructions. In structured text, instructions execute when they are scanned unless pre-conditioning the execution of the structured text instruction.

For example, the ABL instruction is a transitional instruction in ladder diagram. In this example, the ABL instruction only executes on a scan when tag_xic transitions from cleared to set. The ABL instruction does not execute when tag_xic stays set or when tag_xic clears.



In structured text, if writting this example as:

IF tag_xic THEN ABL(0,serial_control);

END_IF;

The ABL instruction will execute every scan that tag_xic is set, not just when tag_xic transitions from cleared to set.

If you want the ABL instruction to execute only when tag_xic transitions from cleared to set, you have to condition the structured text instruction. Use a one-shot to trigger execution.

osri_1.InputBit := tag_xic;

OSRI(osri_1);

IF (osri_1.OutputBit) THEN

ABL(0,serial_control);

END_IF;

Structured Text Components: Constructs

Program constructs alone or nest within other constructs.

If	Use this construct
Doing something if or when specific conditions occur	IF. . . THEN
Selecting what to do based on a numerical value	CASE. . . OF
Doing something a specific number of times before doing anything else	FOR. . . DO
Continuing doing something when certain conditions are true	WHILE. . . DO
Continuing doing something until a condition is true	REPEAT. . . UNTIL

Some Key Words are Reserved

These constructs are not available:

- GOTO
- REPEAT

Logix Designer application will not let you use them as tag names or constructs.

See also

IF_THEN on page 1074

CASE_OF on page 1069

FOR_DO on page 1071

WHILE_DO on page 1080

REPEAT_UNTIL on page 1077

Character string literals

Character string literals include single byte or double byte encoded characters. A single-byte string literal is a sequence of zero or more characters that are prefixed and terminated by the single quote character ('). In single byte character strings, the three-character combination of the dollar sign ($) followed by two hexadecimal digits is interpreted as the hexadecimal representation of the eight-bit character code as shown in the following table.

Tips:
- Character string literals are only applicable to the CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers.
- Studio 5000 only supports single byte characters.

Character string literals

No.	Description	Example
1a	Empty string (length zero)	''
1b	String of length one or character CHAR containing a single character	'A'
1c	String of length one or character CHAR containing the “space” character	' '
1d	String of length one or character CHAR containing the “single quote” character	'$''
1e	String of length one or character CHAR containing the “double quote” character	'"'
1f	Support of two character combinations	'RL'
1g	Support of a character representation with ‘$’ and two hexadecimal characters	'$0A'

Two-character combinations in character strings

No.	Description	Example
1	Dollar sign	$$
2	Single quote	$'
3	Line feed	$L or $l
4	Newline	$N or $n
5	Form feed (page)	$P or $p
6	Carriage return	$R or $r
7	Tabulator	$T or $t

Tips:
- The newline character provides an implementation-independent means of defining the end of a line of data for both physical and file I/O; for printing, the effect is that of ending a line of data and resuming printing at the beginning of the next line.
- The $' combination is only valid inside single quoted string literals.

See also

Structured Text Components: Assignments on page 1059

String Types on page 1069

String Types

Store ASCII characters in tags that use a string type data type to:

- Use the default STRING data type, which stores up to 82 characters
- Create a new string type that stores less or more characters

To create a new string type, refer to the Logix 5000 Controllers ASCII Strings Programming Manual publication 1756-PM013.

Each string type contains the following members:

Name	Data Type	Description	Notes
LEN	DINT	number of characters in the string	The LEN automatically updates to the new count of characters whenever using: • The String Browser to enter characters • Instructions that read, convert, or manipulate a string The LEN shows the length of the current string. The DATA member may contain additional, old characters, which are not included in the LEN count.
DATA	SINT array	ASCII characters of the string	To access the characters of the string, address the name of the tag. For example, to access the characters of the string_1 tag, enter string_1. Each element of the DATA array contains one character. Create new string types that store less or more characters.

See also

Character string literals on page 1068

CASE_OF

Use CASE_OF to select what to do based on a numerical value.

Operands

CASE numeric_expression OF

selector1: statement;

selectorN: statement; ELSE

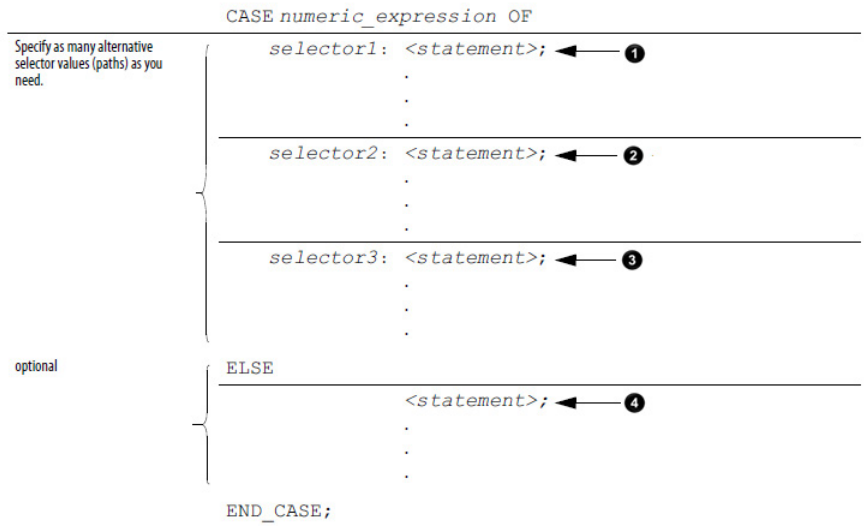
Structured Text

Operand	Type	Format	Enter
Numeric_ expression	SINT INT DINT REAL	Tag expression	Tag or expression that evaluates to a number (numeric expression)
Selector	SINT INT DINT REAL	Immediate	Same type as numeric_expression

IMPORTANT If using REAL values, use a range of values for a selector because a REAL value is more likely to be within a range of values than an exact match of one, specific value.

Description

The syntax is described in the table.



These are the syntax for entering the selector values.

When selector is	Enter
One value	value: statement
Multiple, distinct values	value1, value2, valueN : <statement> Use a comma (,) to separate each value.
A range of values	value1..valueN : <statement> Use two periods (..) to identify the range.
Distinct values plus a range of values	valuea, valueb, value1..valueN : <statement>

The CASE construct is similar to a switch statement in the C or C++ programming languages. With the CASE construct, the controller executes only the statements that associated with the first matching selector value. Execution always breaks after the statements of that selector and goes to the END_CASE statement.

Affects Math Status Flags

No

Major/Minor Faults

None

Example

If you want this	Enter this structured text
If recipe number = 1 then Ingredient A outlet 1 = open (1) Ingredient B outlet 4 = open (1)	CASE recipe_number OF
	1: Ingredient_A.Outlet_1 :=1; Ingredient_B.Outlet_4 :=1;
If recipe number = 2 or 3 then Ingredient A outlet 4 = open (1) Ingredient B outlet 2 = open (1)	2,3: Ingredient_A.Outlet_4 :=1; Ingredient_B.Outlet_2 :=1;
If recipe number = 4, 5, 6, or 7 then Ingredient A outlet 4 = open (1) Ingredient B outlet 2 = open (1)	4 to 7: Ingredient_A.Outlet_4 :=1; Ingredient_B.Outlet_2 :=1;
If recipe number = 8, 11, 12, or 13 then Ingredient A outlet 1 = open (1) Ingredient B outlet 4 = open (1)	8,11...13 Ingredient_A.Outlet_1 :=1; Ingredient_B.Outlet_4 :=1;
Otherwise all outlets = closed (0)	ELSE
	Ingredient_A.Outlet_1 [:=]0; Ingredient_A.Outlet_4 [:=]0; Ingredient_B.Outlet_2 [:=]0; Ingredient_B.Outlet_4 [:=]0;
	END_CASE;

The [:=] tells the controller to also clear the outlet tags whenever the controller does the following:

Enters the RUN mode.

Leaves the step of an SFC if configuring the SFC for Automatic reset. This applies only embedding the assignment in the action of the step or using the action to call a structured text routine via a JSR instruction.

FOR_DO

Use the FOR_DO loop to perform an action a number of times before doing anything else.

When enabled, the FOR instruction repeatedly executes the Routine until the Index value exceeds the Terminal value. The step value can be positive or negative. If it is negative, the loop ends when the index is less than the terminal value.. If it is positive, the loop ends when the index is greater than the terminal value.

Each time the FOR instruction executes the routine, it adds the Step size to the Index.

Do not loop too many times in a single scan. An excessive number of repetitions causes the controller watchdog to timeout and causes a major fault.

Operands

FOR count:= initial_value TO

final_value BY increment DO

<statement>;

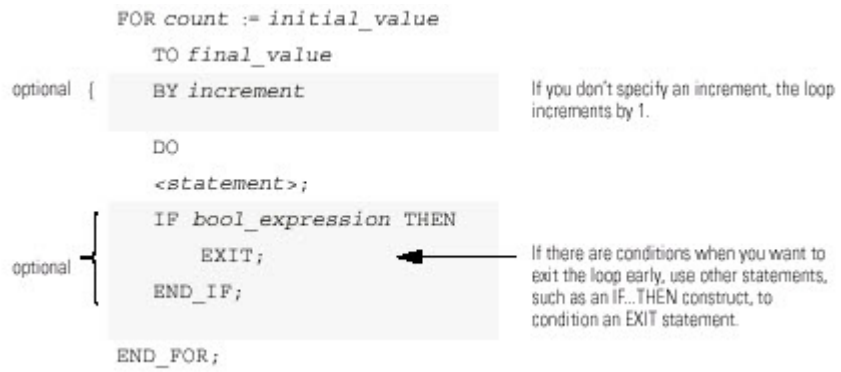
END_FOR;

Operand	Type	Format	Description
count	SINT INT DINT	Tag	Tag to store count position as the FOR_DO executes
initial_ value	SINT INT DINT	Tag expression Immediate	Must evaluate to a number Specifies initial value for count
final_ value	SINT INT DINT	Tag expression Immediate	Specifies final value for count, which determines when to exit the loop
increment	SINT INT DINT	Tag expression Immediate	(Optional) amount to increment count each time through the loop If you don't specify an increment, the count increments by 1.

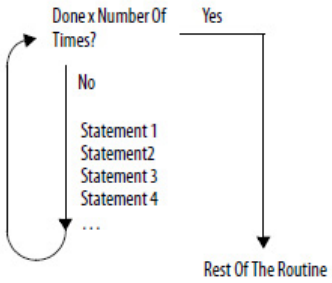
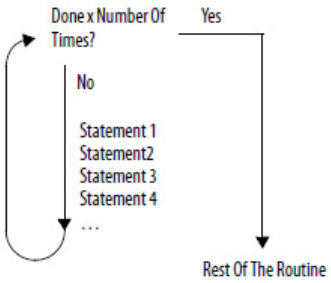
IMPORTANT Do not iterate within the loop too many times in a single scan.
The controller does not execute other statements in the routine until it completes the loop.
A major fault occurs when completing the loop takes longer than the watchdog timer for the task.
Consider using a different construct, such as IF_THEN.

Description

The syntax is described in the table.



This diagrams illustrates how a FOR_DO loop executes, and how an EXIT statement leaves the loop early.

	
The FOR_DO loop executes a specific number of times.	To stop the loop before the count reaches the last value, use an EXIT statement.

Affects Math Status Flags

No

Major/Minor Faults

A major fault will occur if	Fault type	Fault code
The construct loops too long.	6	1

Example 1

If performing the following,	Enter this structured text

Clear bits 0...31 in an array of BOOLs:
Initialize the subscript tag to 0.
Clear i . For example, when subscript = 5, clear array[5].
Add 1 to subscript.
If subscript is ≤ to 31, repeat 2 and 3.
Otherwise, stop.
	For subscript:=0 to 31 by 1 do
	array[subscript] := 0;
	End_for;

Example 2

If performing the following,	Enter this structured text

A user-defined data type (structure) stores the following information about an item in your inventory:
• Barcode ID of the item (String data type)
• Quantity in stock of the item (DINT data type)
	SIZE(Inventory,0,Inventory_Items);
	For position:=0 to Inventory_Items - 1 do
	If Barcode = Inventory[position].ID then
	Quantity := Inventory[position].Qty;
	Exit;
	End_if;

An array of the above structure contains an element for each different item in your inventory. You want to search the array for a specific product (use its bar code) and determine the quantity that is in stock. 1. Get the size (number of items) of the Inventory array and store the result in 2. Inventory_Items (DINT tag). Initialize the position tag to 0. 3. If Barcode matches the ID of an item in the array, then: Set the Quantity tag = Inventory[position].Qty. This produces the quantity in stock of the item. Stop. Barcode is a string tag that stores the bar code of the item for which you are searching. For example, when position = 5, compare Barcode to Inventory[5].ID. 4. Add 1 to position. 5. If position is ≤ to (Inventory_Items -1), repeat 3 and 4. Since element numbers start at 0, the last element is 1 less than the number of elements in the array. Otherwise, stop.	End_for;

IF_THEN

Use IF_THEN to complete an action when specific conditions occur.

Operands

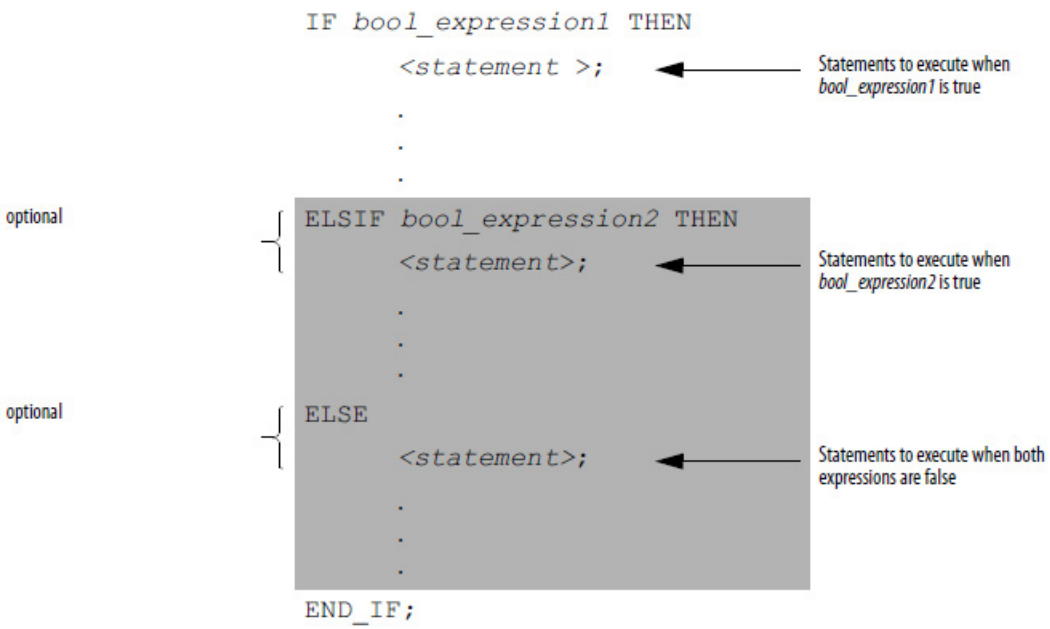
IF bool_expression THEN

<statement>;

Operand	Type	Format	Enter
Bool_ expression	BOOL	Tag expression	BOOL tag or expression that evaluates to a BOOL value (BOOL expression)

Description

The syntax is described in the table.



To use ELSIF or ELSE, follow these guidelines.

To select from several possible groups of statements, add one or more ELSIF statements.

Each ELSIF represents an alternative path.

Specify as many ELSIF paths as you need.

The controller executes the first true IF or ELSIF and skips the rest of the ELSIFs and the ELSE.

To do something when all of the IF or ELSIF conditions are false, add an ELSE statement.

The table summarizes different combinations of IF, THEN, ELSIF, and ELSE.

If	And	Use this construct
Doing something if or when conditions are true	Do nothing if conditions are false	IF_THEN
	Do something else if conditions are false	IF_THEN_ELSE
Selecting alternative statements or groups of statements based on input conditions	Do nothing if conditions are false	IF_THEN_ELSIF
	Assign default statements if all conditions are false	IF_THEN_ELSIF_ELSE

Affects Math Status Flags

No

Major/Minor Faults

None.

Examples

Example 1

IF...THEN

If performing this	Enter this structured text
IF rejects > 3 then	IF rejects > 3 THEN
conveyor = off (0)	conveyor := 0;
alarm = on (1)	alarm := 1;
	END_IF;

Example 2

IF_THEN_ELSE

If performiing this	Enter this structured text
If conveyor direction contact = forward (1) then	IF conveyor_direction THEN
light = off	light := 0;
Otherwise light = on	ELSE
	light [:=] 1;
	END_IF;

The [:=] tells the controller to clear light whenever the controller does the following :

Enters the RUN mode.

Leaves the step of an SFC if you configure the SFC for Automatic reset. (This applies only if you embed the assignment in the action of the step or use the action to call a structured text routine via a JSR instruction.)

Example 3

IF...THEN...ELSIF

If performing this	Enter this structured text
If sugar low limit switch = low (on) and sugar high limit switch = not high (on) then	IF Sugar.Low & Sugar.High THEN
inlet valve = open (on)	Sugar.Inlet [:=] 1;
Until sugar high limit switch = high (off)	ELSIF NOT(Sugar.High) THEN
	Sugar.Inlet := 0;
	END_IF;

The [:=] tells the controller to clear Sugar.Inlet whenever the controller does the following :

Enters the RUN mode.

Leaves the step of an SFC if you configure the SFC for Automatic reset. (This applies only if you embed the assignment in the action of the step or use the action to call a structured text routine via a JSR instruction.)

Example 4

IF...THEN...ELSIF...ELSE

If performing this	Enter this structured text
If tank temperature > 100	IF tank.temp > 200 THEN
then pump = slow	pump.fast :=1; pump.slow :=0; pump.off :=0;
If tank temperature > 200	ELSIF tank.temp > 100 THEN
then pump = fast	pump.fast :=0; pump.slow :=1; pump.off :=0;
Otherwise pump = off	ELSE
	pump.fast :=0; pump.slow :=0; pump.off :=1;
	END_IF;

REPEAT_UNTIL

Use the REPEAT_UNTIL loop to continue performing an action until conditions are true.

Operands

REPEAT

<statement>;

Structured Text

Operand	Type	Format	Enter
bool_ expression	BOOL	Tag expression	BOOL tag or expression that evaluates to a BOOL value (BOOL expression)

IMPORTANT Do not iterate within the loop too many times in a single scan.
The controller does not execute other statements in the routine until it completes the loop.
A major fault occurs when completing the loop takes longer than the watchdog timer for the task.
Consider using a different construct, such as IF_THEN.

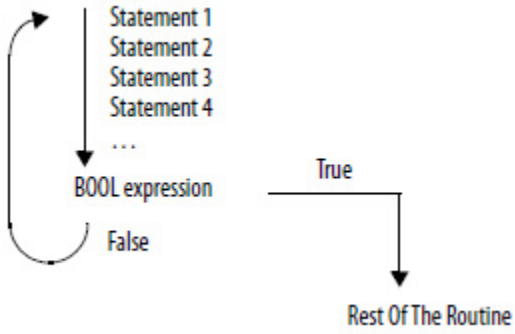
Description

The syntax is:

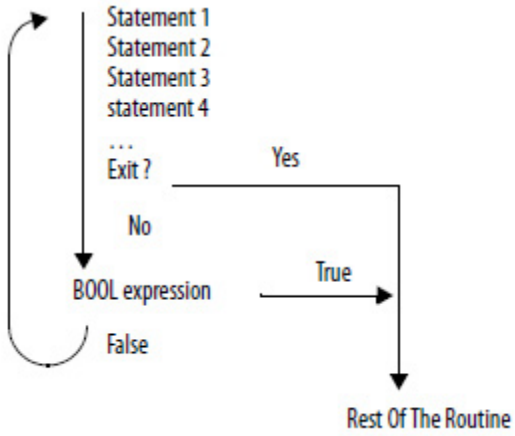
```
REPEAT
   <statement>;          <-- statements to execute while bool_expression1 is false
   IF bool_expression2 THEN
      EXIT;              <-- optional; If there are conditions when you want to exit the loop early, use other statements, such as an IF...THEN construct, to condition an EXIT statement.
   END_IF;
UNTIL bool_expression1
END_REPEAT;
```

The following diagrams show how a REPEAT_UNTIL loop executes and how an EXIT statement leaves the loop early.

While the bool_expression is false, the controller executes only the statements within the REPEAT_UNTIL loop.



To stop the loop before the conditions are false, use an EXIT statement.



Affects Math Status Flags

No

Fault Conditions

A major fault will occur if	Fault type	Fault code
The construct loops too long	6	1

Example 1

If performing the following,	Enter this structured text

The REPEAT_UNTIL loop executes the statements in the construct and then determines if the conditions are true before executing the statements again. This differs from the WHILE_DO loop because the WHILE_DO The WHILE_DO loop evaluates its conditions first.
If the conditions are true, the controller then executes the statements within the loop. The statements in a REPEAT_UNTIL loop are always executed at least once. The statements in a WHILE_DO loop might never be executed.
	pos := -1;
	REPEAT
	pos := pos + 2;
	UNTIL ((pos = 101) OR (structarray[pos].value = targetvalue))
	end_repeat;

Example 2

If performing the following,	Enter this structured text

Move ASCII characters from a SINT array into a string tag. (In a SINT array, each element holds one character.) Stop when you reach the carriage return.
Initialize Element_number to 0.
Count the number of elements in SINT_array (array that contains the ASCII characters) and store the result in SINT_array_size (DINT tag).
Set String_tag[element_number] = the character at SINT_array[element_number].
Add 1 to element_number. This lets the controller check the next character in
SINT_array.
Set the Length member of String_tag = element_number. (This records the number of characters in String_tag so far.)
If element_number = SINT_array_size, then stop. (You are at the end of the array and it does not contain a carriage return.)
If the character at SINT_array[element_number] = 13 (decimal value of the carriage return), then stop.
	element_number := 0;
	SIZE(SINT_array, 0, SINT_array_size);
	Repeat
	String_tag.DATA[element_number] := SINT_array[element_number];
	element_number := element_number + 1;
	String_tag.LEN := element_number;
	If element_number = SINT_array_size then
	exit;
	end_if;
	Until SINT_array[element_number] = 13
	end_repeat;

WHILE_DO

Use the WHILE_DO loop to continue performing an action while certain conditions are true.

Operands

WHILE bool_expression DO

<statement>;

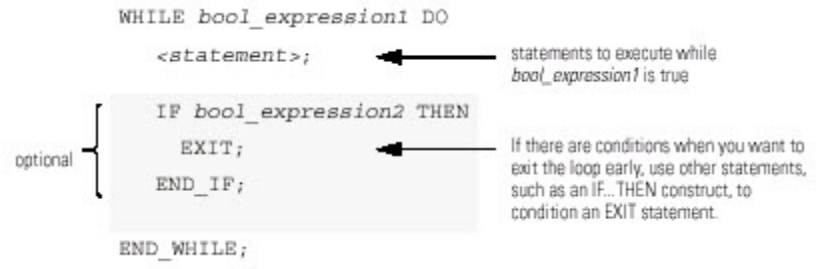
Structured Text

Operand	Type	Format	Description
bool_expression	BOOL	tag expression	BOOL tag or expression that evaluates to a BOOL value

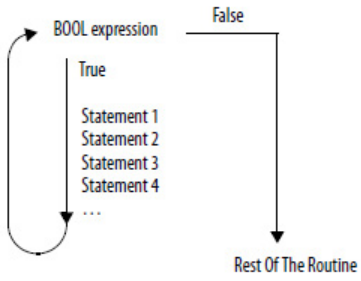
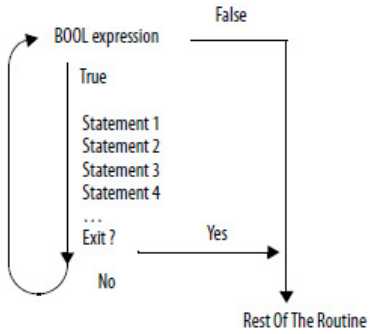
IMPORTANT	Do not iterate within the loop too many times in a single scan. The controller does not execute any other statements in the routine until it completes the loop. A major fault occurs when completing the loop takes longer than the watchdog timer for the task. Consider using a different construct, such as IF_THEN.

Description

The syntax is:



The following diagrams illustrate how a WHILE_DO loop executes, and how an EXIT statement leaves the loop early.

	
While the bool_expression is true, the controller executes only the statements within the WHILE_DO loop.	To stop the loop before the conditions are true, use an EXIT statement.

Affects Math Status Flags

No

Fault Conditions

A major fault will occur if	Fault type	Fault code
the construct loops too long	6	1

Example 1

If performing the following,	Enter this structured text

The WHILE_DO loop evaluates its conditions first. If the conditions are true, the controller then executes the statements within the loop.
This differs from the REPEAT_UNTIL loop because the REPEAT_UNTIL loop executes the statements in the construct and then determines if the conditions are true before executing the statements again. The statements in a REPEAT_UNTIL loop are always executed at least once. The statements in a WHILE_DO loop might never be executed.
	pos := 0;
	While ((pos <= 100) & structarray[pos].value <> targetvalue)) do
	  pos := pos + 2;
	  String_tag.DATA[pos] := SINT_array[pos];
	end_while;

Example 2

If performing the following,	Enter this structured text

Move ASCII characters from a SINT array into a string tag. (In a SINT array, each element holds one character.) Stop when you reach the carriage return.
Initialize Element_number to 0.
Count the number of elements in SINT_array (array that contains the ASCII characters) and store the result in SINT_array_size (DINT tag).
If the character at SINT_array[element_number] = 13 (decimal value of the carriage return), then stop.
	element_number := 0;
	SIZE(SINT_array, 0, SINT_array_size);
	While SINT_array[element_number] <> 13 do
	String_tag.DATA[element_number] := SINT_array[element_number];
	element_number := element_number + 1;
	String_tag.LEN := element_number;
	If element_number = SINT_array_size then
	exit;
	end_if;

Set String_tag[element_number] = the character at SINT_array[element_number]. Add 1 to element_number. This lets the controller check the next character in SINT_array. Set the Length member of String_tag = element_number. (This records the number of characters in String_tag so far.) If element_number = SINT_array_size, then stop. (You are at the end of the array and it does not contain a carriage return.)	end_while;

Structured Text Attributes

Click a topic below for more information on issues that are unique to structured text programming. Review this information to make sure you understand how your structured text programming will execute.

See also

Structured Text Components: Assignments on page 1059

Structured Text Components: Expressions on page 1061

Structured Text Instructions on page 1066

Structured Text Components: Constructs on page 1067

Structured Text Components: Comments on page 1058

Common Attributes for Advanced Process Control and Drives Instructions

Follow the guidelines in this chapter for the common attributes for the Advanced Process Control and Drives Instructions.

Common Attributes

For more information on attributes that are common to the Logix 5000™ instructions, click any of the topics below.

Math Status Flags on page 1083

Immediate Values on page 1085

Data Conversions on page 1086

Elementary data types on page 1089

Floating Point Values on page 1092

Index Through Arrays on page 1094

Bit Addressing on page 1094

Math Status Flags

Follow the guidelines in this topic for Math Status Flags.

Description

Controllers	Description
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	A set of Math Status Flags for accessing directly with instructions. These flags are only updated in ladder diagram routines, and are not tags, and flag aliases are not applicable.
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	A set of Math Status Flags for accessing directly with instructions. These flags are updated in all routine types, but are not tags, and flag aliases are not applicable.

Status Flags

Status Flag	Description (For CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers)	Description (For CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers)
S:FS First scan flag	The first scan flag is set by the controller: • The first time a program is scanned after the controller goes to Run mode • The first time a program is scanned after the program is uninhibited • When a routine is called from an SFC Action and the step that owns that Action is first scanned. Use the first scan flag to initialize data for use in later scans. It is also referred to as the first pass bit.	The first scan flag is set by the controller: • The first time a program is scanned after the controller goes to Run mode • The first time a program is scanned after the program is uninhibited • When a routine is called from an SFC Action and the Step that owns that Action is first scanned. Use this flag to initialize data for use in later scans. It is also referred to as the first pass bit.
S:N Negative flag	The controller sets the negative flag when the result of a math or logical operation is a negative value. Use this flag as a quick test for a negative value.	The controller sets the negative flag when the result of a math or logical operation is a negative value. Use this flag as a quick test for a negative value. Using S:N is more efficient than using the CMP instruction.
S:Z Zero flag	The zero flag is set by the controller when the result of a math or logical operation is zero. Use this flag as a quick test for a zero value. The zero flag clears at the start of executing an instruction capable of setting this flag.	The controller sets the zero flag when the result of a math or logical operation is zero. Use this flag as a quick test for a zero value.
S:V Overflow flag	The controller sets the overflow flag when: • The result of a math operation results in an overflow. For example, adding 1 to a SINT generates an overflow when the value goes from 127 through -128. • The destination tag is too small to hold the value. For example, if you try to store the value 123456 to a SINT or INT tag. Use the overflow flag to verify the result of an operation is still in range. If the data being stored is a string type, S:V is set if the string is too large to fit into the destination tag. **Tip:** If applicable, set S:V with an OTE or OTL instruction. Click **Controller Properties > Advanced tab > Report Overflow Faults** to enable or disable reporting overflow faults. If an overflow occurs while evaluating an array subscript, a minor fault is generated and a major fault is generated to indicate the index is out of range.	The controller sets the overflow flag when: • The result of a math operation results in an overflow. For example, adding 1 to a SINT generates an overflow when the value goes from 127...-128. • The destination tag is too small to hold the value. For example, if you try to store the value 123456 to a SINT or INT tag. Use the overflow flag to check that the result of an operation is still in range. A minor fault is generated anytime an overflow flag is set. **Tip:** If applicable,set S:V with an OTE or OTL instruction.
S:C Carry flag	The controller sets the carry flag when the result of a math operation resulted in the generation of a carry out of the most significant bit. Only the ADD and SUB instructions, and not the + and – operators, with integer values affect this flag.	The controller sets the carry flag when the result of a math operation resulted in the generation of a carry out of the most significant bit.

Status Flag	Description (For CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers)	Description (For CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers)
S:MINOR Minor fault flag	The controller sets the minor fault flag when there is at least one minor program fault. Use the minor fault tag to test if a minor fault occurred. This bit only triggers by programming faults, such as overflow. It is not triggered by a battery fault. The bit clears at the beginning of every scan. **Tip:** If applicable, explicitly set S:MINOR with an OTE or OTL instruction.	The controller sets the minor fault flag when there is at least one minor program fault. Use the minor fault flag to test if a minor fault occurred and take appropriate action. This bit is triggered only by programming faults, such as overflow. It is not triggered by a battery fault. The bit clears at the beginning of every scan. **Tip:** If applicable, explicitly set S:MINOR with an OTE or OTL instruction.
IMPORTANT	The math status flags are set based on the stored value. Instructions that normally do not affect math status flags might appear to affect math status flags if type conversion occurs from mixed data types for the instruction parameters. The type conversion process sets the math status flags.	

Expressions in Array Subscripts

Controllers	Description
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Expressions do not set status flags based on the results of math operations. If expressions overflow: • A minor fault generates if the controller is configured to generate minor faults. • A major fault (type 4, code 20) generates because the resulting value is out of range.
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Expressions set status flags based on the results of math operations. If an array subscript is an expression, the expression and the instruction could generate minor faults.

Tip: If an array subscript is too large (out of range), a major fault (type 4, code 20) generates.

Immediate values

When you enter an immediate value (constant) in decimal format (for example, -2, 3) the controller stores the value by using 32 bits. If you enter a value in a radix other than decimal, such as binary or hexadecimal, and do not specify all 32 bits, the controller places a zero in the bits that you do not specify (zero-fill).

IMPORTANT Zero-fill of immediate binary, octal or hexadecimal values less than 32 bits.

If you enter	The controller stores
-1	16#ffff ffff (-1)
16#ffff (-1)	16#0000 ffff (65535)

8#1234 (668)	16#0000 029c (668)
2#1010 (10)	16#0000 000a (10)

Integer Immediate Values

If you enter	The controller stores
Without any suffix	DINT
"U" or "u"	UDINT
"L" or "l"	LINT
"UL", "ul", "Ul", or "uL"	ULINT

Floating Point Immediate Values

If you enter	The controller stores
Without any suffix	REAL
"L" or "l"	LREAL

Data Conversions

Data conversions occur when mixing data types in programming.

When programming:	Conversions can occur when you:
Ladder Diagram Structured Text	Mix data types for the parameters within one Instruction or expression.
Function Block	Wire two parameters that have different data types

Instructions execute faster and require less memory if all the operands of the instruction use:

- The same data type.
- An intermediate data type:
  - All function block instructions support one data type operand only.
  - If mixing data types or use tags that are not the optimal data type, the controller converts the data according to these rules:
    - Operands are converted according to the ranking of data types from SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, and LREAL with ranking from 1 (the lowest) to 10 (the highest).

Tip: To reduce the time and memory for converting data, use the same data type for all the operands of an instruction.

Convert SINT or INT to DINT or DINT to LINT

A SINT or INT input source tag gets promoted to a DINT value by a sign-extension for Source Tag. Instructions that convert SINT or INT values to DINT values use one of the following conversion methods.

This conversion method	Converts data by placing

Sign-extension	The value of the leftmost bit (the sign of the value) into each bit position to the left of the existing bits until there are 32 or 64 bits.
Zero-fill	Zeroes to the left of the existing bits until there are 32 or 64 bits.

Logical instructions use zero fill. All other instructions use sign-extension

The following example shows the results of converting a value using sign-extension and zero-fill.

This value	**2#1111_1111_1111_1111**	**(-1)**
Converts to this value by sign-extension	2#1111_1111_1111_1111_1111_1111_1111_1111	(-1)
Converts to this value by zero-fill	2#0000_0000_0000_0000_1111_1111_1111_1111	(65535)

If you use a SINT or INT tag and an immediate value in an instruction that converts data by sign-extension, use one of these methods to handle immediate values.

Specify any immediate value in the decimal radix.

If you enter the value in a radix other than decimal, specify all 32 bits of the immediate value. To do so, enter the value of the leftmost bit into each bit position to its left until there are 32 bits.

Create a tag for each operand and use the same data type throughout the instruction. To assign a constant value, either:

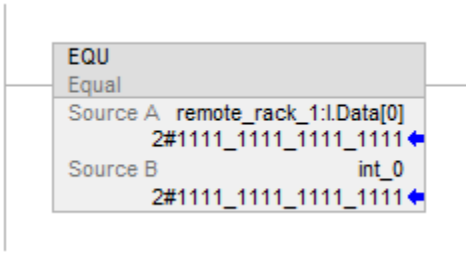
Enter it into one of the tags.

Add a MOV instruction that moves the value into one of the tags.

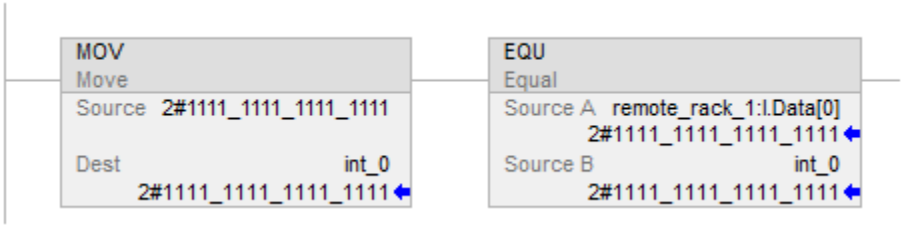
Use a MEQ instruction to check only the required bits.

The following examples show two ways to mix an immediate value with an INT tag. Both examples check the bits of a 1771 I/O module to determine if all the bits are on. Since the input data word of a 1771 I/O module is an INT tag, it is easiest to use a 16-bit constant value.

IMPORTANT Mixing an INT tag with an immediate value
Since remote_rack_1:I.Data[0] is an INT tag, the value to check it against is also entered as an INT tag.



IMPORTANT Mixing an INT tag with an immediate value
Since remote_rack_1:I.Data[0] is an INT tag, the value to check it against first moves into int_0, also an INT tag. The EQU instruction then compares both tags.



Convert Integer to REAL

The controller stores REAL values in IEEE single-precision, floating-point number format. It uses one bit for the sign of the value, 23 bits for the base value, and eight bits for the exponent (32 bits total). If you mix an integer tag (SINT, INT, or DINT) and a REAL tag as inputs in the same instruction, the controller converts the integer value to a REAL value before the instruction executes.

- A SINT or INT value always converts to the same REAL value.
- A DINT value may not convert to the same REAL value:
- A REAL value uses up to 24 bits for the base value (23 stored bits plus a 'hidden' bit).
- A DINT value uses up to 32 bits for the value (one for the sign and 31 for the value).

If the DINT value requires more than 24 significant bits, it might not convert to the same REAL value. If it will not, the controller stores the uppermost 24 bits rounded to the nearest even value.

Convert DINT to SINT or INT

To convert a DINT value to a SINT or INT value, the controller truncates the upper portion of the DINT and stores the lower bits that fit in the data type. If the value is too large the conversion generates an overflow.

	Convert a DINT to an INT and a SINT	
This DINT value	Converts to this smaller value	
16#0001_0081 (65,665)	INT:	16#0081 (129)
	SINT:	16#81 (-127)

Convert REAL to SINT, INT, or DINT

To convert a REAL value to an integer value, the controller rounds any fractional part and stores the bits that fit in the result data type. If the value is too large the conversion generates an overflow.

Numbers round as in the following examples.

Fractions < 0.5 round down to the nearest whole number.

Fractions > 0.5 round up to the nearest whole number.

Fractions = 0.5 round up or down to the nearest even number.

Important:	Conversion of REAL values to DINT values
This REAL value	**Converts to this DINT value**
-2.5 -3.5	-2 -4
-1.6	-2
-1.5	-2
-1.4	-1
1.4	1
1.5	2
1.6	2
2.5 3.5	2 4

Elementary data types

The controller supports the elementary data types defined in IEC 1131-3 defined data types. The elementary data types are:

Data type	Description	Range
BOOL	1-bit boolean	0 = cleared 1 = set
SINT	1-byte integer	-128 to 127
INT	2-byte integer	-32,768 to 32,767
DINT	4-byte integer	-2,147,483,648 to 2,147,483,647
REAL	4-byte floating-point number	$-3.402823E^{38}$ to $-1.1754944E^{-38}$ (negative values) and 0 and $1.1754944E^{-38}$ to $3.402823E^{38}$ (positive values)
LINT	8-byte integer **Note:** The LINT data type has limited use on CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers. They can be used only with copy (COP, CPS) instructions, the CST/WallClock Time attribute, time synchronization, and Add-On Instructions.	0 to 32,535,129,599,999,999
USINT	1-byte unsigned integer	0 to 255
UINT	2-byte unsigned integer	0 to 65,535
UDINT	4-byte unsigned integer	0 to 4,294,967,295
ULINT	8-byte unsigned integer	0 to 18,446,744,073,709,551,615

Data type	Description	Range
REAL	4-byte floating-point number	-3.4028235E38 to -1.1754944E-38 (negative values) and 0.0 and 1.1754944E-38 to 3.4028235E38 (positive values)
LREAL	8-byte floating-point number	-1.7976931348623157E308 to -2.2250738585072014E-308 (negative values) and 0.0 and 2.2250738585072014E-308 to 1.7976931348623157E308 (positive values)

These controllers support the following elementary data types:

Controllers	Data type
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	SINT, INT, DINT, LINT, REAL USINT, UINT, UDINT, ULINT, LREAL
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	SINT, INT, DINT, LINT, REAL.

The controller handles all immediate values as DINT data types.

The REAL data type also stores ± infinity and ± NAN, but the software display differs based on the display format.

Data type conversions

When data types are mixed for operands within an instruction, some instructions automatically convert data to an optimal data type for that instruction. In some cases, the controller converts data to fit a new data type; in some cases, the controller just fits the data as best it can.

Conversion	Result		
larger integer to smaller integer	The controller truncates the upper portion of the larger integer and generates an overflow. For example:		
	Decimal		**Binary**
	DINT	65,665	0000_0000_0000_0001_0000_0000_1000_0001
	INT	129	0000_0000_1000_0001
	SINT	-127	1000_0001
SINT or INT to REAL	No data precision is lost		

<table>
<tr><th>Conversion</th><th>Result</th></tr>
<tr><td>DINT to REAL</td><td>Data precision could be lost. Both data types store data in 32 bits, but the REAL type uses some of its 32 bits to store the exponent value. If precision is lost, the controller takes it from the least-significant portion of the DINT.</td></tr>
<tr><td>LREAL to LREAL</td><td>No data precision is lost.</td></tr>
<tr><td>LREAL TO REAL</td><td>Data precision could be lost.</td></tr>
<tr><td>LREAL/REAL to unsigned integer</td><td>Data precision could be lost. If the source value is too big to fit into destination the controller stores what it can and may produce an overflow.</td></tr>
<tr><td>Signed Integer/Unsigned Integer to LREAL/REAL</td><td>If the integer value has more significant bits than can be stored in the destination, the lower bits will be truncated.</td></tr>
<tr><td>Signed integer to unsigned integer</td><td>If the source value is too big to fit into destination, the controller stores what it can and may produce an overflow.</td></tr>
<tr><td>Unsigned integer to signed integer</td><td>If the source value is too big to fit into destination, the controller stores what it can and may produce an overflow.</td></tr>
<tr><td>REAL to integer</td><td>The controller rounds the fractional part and truncates the upper portion of the non-fractional part. If data is lost, the controller sets the overflow status flag.
Rounding is to the nearest whole number:
less than 0.5, round down; equal to 0.5, round to nearest even integer; greater than 0.5, round up
For example:
<table>
<tr><th>REAL (source)</th><th>DINT (result)</th></tr>
<tr><td>1.6</td><td>2</td></tr>
<tr><td>-1.6</td><td>-2</td></tr>
<tr><td>1.5</td><td>2</td></tr>
<tr><td>-1.5</td><td>-2</td></tr>
<tr><td>1.4</td><td>1</td></tr>
<tr><td>-1.4</td><td>-1</td></tr>
<tr><td>2.5</td><td>2</td></tr>
<tr><td>-2.5</td><td>-2</td></tr>
</table></td></tr>
</table>

Do not convert data to or from the BOOL data type.

IMPORTANT The math status flags are set based on the value being stored. Instructions that normally do not affect math status keywords might appear to do so if type conversion occurs because of mixed data types for the instruction parameters. The type conversion process sets the math status keywords.

Safety Data Types

The Logix Designer application prevents the modification of a User Defined or Add-On Defined type that would cause an invalid data type for User Defined or Add-On Defined types that are referenced directly or indirectly by a Safety tag. (This includes nested structures.)

Safety tags can be composed of the following data types:

- All elementary data types.
- Predefined types that are used for safety application instructions.
- User-defined data types or arrays that are composed of the previous two types.

Online edits of user-defined data type member names in safety tags

Online editing is allowed for member names of user-defined data types on CompactLogix 5380, Compact GuardLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. However, online editing is disabled when a user-defined data type is used on a safety tag and the controller is in the Safety Secured state.

See also

Math Status Flags on page 1083

Floating Point Values

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

Logix controllers handle floating point values according to the IEEE 754 standard for floating-point arithmetic. This standard defines how floating point numbers are stored and calculated. The IEEE 754 standard for floating point math was designed to provide speed and the ability to handle very large numbers in a reasonable amount of storage space.

A REAL tag stores a single-precision, normalized floating-point number.

An LREAL tag stores a double-precision, normalized floating-point number.

The controllers support these elementary data types:

Controllers	Data Type
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	REAL, LREAL
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	REAL

Denormalized numbers and -0.0 are treated as 0.0

If a computation results in a NAN value, the sign bit could be positive or negative. In this situation, the software displays 1#.NAN with no sign.

Not all decimal values can be exactly represented in this standard format, which results in a loss of precision. For example, if you subtract 10 from 10.1, you expect the result to be 0.1. In a Logix controller, the result could very well be 0.10000038. In this example, the difference between 0.1 and 0.10000038 is .000038%, or practically zero. For most operations, this small inaccuracy is insignificant. To put things in perspective, if you were sending a floating point value to an analog output module, there would be no difference in the output voltage for a value being sent to the module that differs by .000038%.

Guidelines for Floating-point Math Operations

Follow these guidelines:

When performing certain floating-point math operations, there may be a loss of precision due to rounding error. Floating-point processors have their own internal precision that can impact resultant values.

Do not use floating point math for money values or for totalizer functions. Use INT or DINT values, scale the values up, and keep track of the decimal place (or use one INT or DINT value for dollars, and a second INT or DINT value for cents).

Do not compare floating-point numbers. Instead, check for values within a range. The LIM instruction is provided specifically for this purpose.

Totalizer Examples

The precision of the REAL data type affects totalization applications such that errors occur when adding very small numbers to very large numbers.

For example, add 1 to a number over a period of time. At some point the add will no longer affect the result because the running sum is much greater than 1, and there are not enough bits to store the entire result. The add stores as many upper bits as possible and discards the remaining lower bits.

To work around this, do math on small numbers until the results get large. Then, transfer them to another location for additional large-number math. For example:

- x is the small incremented variable.
- y is the large incremented variable.
- z is the total current count that can be used anywhere.
- x = x+1;
- if x = 100,000;
- {
- y = y + 100,000;
- x = 0;
- }
- z = y + x;

Or another example:

- x = x + some_tiny_number;
- if (x >= 100)
- {
- z = z + 100;
- x = x - 100; // there might be a tiny remainder
- }

Index Through Arrays

To dynamically change the array element that your logic references, use tag or expression as the subscript to point to the element. This is similar to indirect addressing in PLC-5 logic. Use these operators in an expression to specify an array subscript:

Tip:
- Logix Designer allows subscripts that are extended data type tags only, and does not support subscript expressions that have extended data types.
- All available integer elementary data types can be used as a subscript index. Only use SINT, INT, and DINT tags with operators to create a subscript expression.

Operator	Description
+	add
-	subtract/negate
*	multiply
/	divide
AND	AND
FRD	BCD to integer
NOT	complement
OR	OR
TOD	integer to BCD
SQR	square root
XOR	exclusive OR

For example:

Definitions	Example	Description
my_list defined as DINT[10]	my_list[5]	This example references element 5 in the array. The reference is static because the subscript value remains constant.
my_list defined as DINT[10] position defined as DINT	MOV the value 5 into position my_list[position]	This example references element 5 in the array. The reference is dynamic because the logic can change the subscript by changing the value of position.
my_list defined as DINT[10] position defined as DINT offset defined as DINT	MOV the value 2 into position MOV the value 5 into offset my_list[position+offset]	This example references element 7 (2+5) in the array. The reference is dynamic because the logic can change the subscript by changing the value of position or offset.

Tip: When entering an array subscript, make sure it is within the boundaries of the specified array. Instructions that view arrays as a collection of elements generate a major fault (type 4, code 20) if a subscript exceeds its corresponding dimension.

Bit Addressing

Bit addressing is used access a particular bit within a larger container. Larger containers include any integer, structure or BOOL array. For example:

Definition	Example	Description
Variable0 defined as LINT has 64 bits	variable0.42	This example references the bit 42 of variable0.
variable1 defined as DINT has 32 bits	variable1.2	This example references the bit 2 of variable1.
variable2 defined as INT has 16 bits	variable2.15	This example references the bit 15 of variable2.
variable3 defined as SINT holds 8 bits	variable3.[4]	This example references bit 4 of variable3.
variable4 defined as COUNTER structure has 5 status bits	variable4.DN	This example references the DN bit of variable4.
MyVariable defined as BOOL[100] MyIndex defined as SINT	MyVariable[(MyIndex AND NOT 7) / 8].[MyIndex AND 7]	This example references a bit within a BOOL array.
MyArray defined as BOOL[20]	MyArray[3]	This example references the bit 3 of MyArray.
variable5 defined as ULINT holds 64 bits	variable5.53	This example references the bit 53 of variable5.

Use Bit Addressing anywhere a BOOL typed tag is allowed.

See also

Index Through Arrays on page 1094

Function Block Faceplate Controls

The Logix Designer programming application includes faceplate controls for some function block instructions. Faceplates are Active-X controls used in applications that acts as an Active-X container. The faceplates communicate with the controller via the <RSLC> OPC Server or the FactoryTalk Linx Gateway.

IMPORTANT The Logix Designer programming application is not a valid Active-X container. An Active-X container is required to use the faceplates.

These instructions have faceplates:

- Process Discrete Input (PDI)
- Process Discrete Output (PDO)
- Process Analog Input (PAI)
- Process Analog Output (PAO)
- Alarm (ALM)
- Enhanced Select (ESEL)
- Totalizer (TOT)
- Ramp/Soak (RMPS)

- Discrete 2-State Device (D2SD)
- Discrete 3-State Device (D3SD)
- Enhanced PID (PIDE)

Configure the faceplates through property pages that open through container applications.

All faceplates have these property pages in common:

- General
- Display
- Font
- Locale

See also

Faceplate Control Properties Dialog - General Tab on page 1097

Faceplate Control Properties Dialog - Display Tab on page 1097

Faceplate Control Properties Dialog - Font Tab on page 1098

Faceplate Control Properties Dialog - LocaleTab on page 1100

Faceplate Control Properties Dialog - General Tab

Use this tab to define/modify how the control operates.

Parameters

Communication

Select RSLinx Classic OPC Server or FactoryTalk Linx. If you select RSLinx Classic OPC Server, you must also specify:

- whether to launch remotely
- the access path to the remote machine

If you select FactoryTalk Linx FactoryTalk, you must also specify the FactoryTalk Area

Tag

Enter the name of a specific function block instruction to connect with this control.

Update Rate

Enter the Update Rate of the control in seconds. You can click the arrows to increment this value in 0.25 second increments. The default value is 1.00 second.

See also

Faceplate Control Properties Dialog - Display Tab on page 1097

Faceplate Control Properties Dialog - Font Tab on page 1098

Faceplate Control Properties Dialog - LocaleTab on page 1100

Faceplate Control Properties Dialog - Display Tab

Use this tab to define how the faceplate control will appear on your screen.

Parameters

Background Color

This button indicates the color of the faceplate's background. Click the button to change this color. The default color is light gray.

Show Frame

Check or uncheck this box, depending on whether or not you wish to display a 3-dimensional frame around this control. Using this option allows you to separate the control from other items that may appear on your display. This option is checked, by default.

OK

Click this button to accept your edits and close the Faceplate Control Properties dialog.

Cancel

Click this button to cancel your edits and close the Faceplate Control Properties dialog.

Apply

Click this button to apply your edits and continue editing in the Faceplate Control Properties dialog.

See also

Faceplate Control Properties Dialog - General Tab on page 1097

Faceplate Control Properties Dialog - Font Tab on page 1098

Faceplate Control Properties Dialog - LocaleTab on page 1100

Faceplate Control Properties Dialog - Font Tab

Use this tab to define the fonts that appear on the faceplates. From here, you can configure a ControlFont to be used in the main part of the faceplate, and a MinorFont to be used in scales and other minor portions of the faceplate.

Parameters

Property Name

Choose the font you want to configure from the pull-down menu. Choose from ControlFont or MinorFont; the default is ControlFont.

Font

Choose the font you wish to use for the control from the list of available fonts. The default font is Arial.

Style

Choose the style you wish to use for the control from the pull-down menu. The default style is Regular.

Size

Enter the point size you wish to use for this font. The default size of the ControlFont is 10.5 points; the default size of the MinorFont is 8.25 points.

Strikeout

Check this box if you want to use the strikeout effect, which draws a line through the font. This option is unchecked, by default.

Underline

Check this box if you want to use the underline effect, which draws a line below the font. This option is unchecked, by default.

OK

Click this button to accept your edits and close the Faceplate Control Properties dialog.

Cancel

Click this button to cancel your edits and close the Faceplate Control Properties dialog.

Apply

Click this button to apply your edits and continue editing in the Faceplate Control Properties dialog.

See also

Faceplate Control Properties Dialog - General Tab on page 1097

Faceplate Control Properties Dialog - Display Tab on page 1097

Faceplate Control Properties Dialog - LocaleTab on page 1100

Faceplate Control Properties Dialog - LocaleTab

Use this tab to define the language requirements for the faceplates.

Parameters

Locale

Choose the language you want to use from the pull-down menu. Choose from:

- English
- Portuguese
- French
- Italian
- German
- Spanish

OK

Click this button to accept your edits and close the Faceplate Control Properties dialog.

Cancel

Click this button to cancel your edits and close the Faceplate Control Properties dialog.

Apply

Click this button to apply your edits and continue editing in the Faceplate Control Properties dialog.

See also

Faceplate Control Properties Dialog - General Tab on page 1097

Faceplate Control Properties Dialog - Display Tab on page 1097

Faceplate Control Properties Dialog - Font Tab on page 1098

ASCII Character Codes

ASCII character codes

Character	Dec	Hex	Character	Dec	Hex	Character	Dec	Hex	Character	Dec	Hex
[ctrl-@] NUL	0	$0	SPACE	32	$20	@	64	$40	‘	96	$60
[ctrl-A] SOH	1	$1	!	33	$21	A	65	$41	a	97	$61
[ctrl-B] STX	2	$2	“	34	$22	B	66	$42	b	98	$62
[ctrl-C] ETX	3	$3	#	35	$23	C	67	$43	c	99	$63
[ctrl-D] EOT	4	$4	$	36	$24	D	68	$44	d	100	$64
[ctrl-E] ENQ	5	$5	%	37	$25	E	69	$45	e	101	$65
[ctrl-F] ACK	6	$6	&	38	$26	F	70	$46	f	102	$66
[ctrl-G] BEL	7	$7	‘	39	$27	G	71	$47	g	103	$67
[ctrl-H] BS	8	$8	(	40	$28	H	72	$48	h	104	$68
[ctrl-I] HT	9	$9	)	41	$29	I	73	$49	i	105	$69
[ctrl-J] LF	10	$I ($0A)	*	42	$2A	J	74	$4A	j	106	$6A
[ctrl-K] VT	11	$0B	+	43	$2B	K	75	$4B	k	107	$6B
[ctrl-L] FF	12	$0C	,	44	$2C	L	76	$4C	l	108	$6C
[ctrl-M] CR	13	$r ($0D)	-	45	$2D	M	77	$4D	m	109	$6D
[ctrl-N] SO	14	$0E	.	46	$2E	N	78	$4E	n	110	$6E
[ctrl-O] SI	15	$0F	/	47	$2F	O	79	$4F	o	111	$6F
[ctrl-P] DLE	16	$10	0	48	$30	P	80	$50	p	112	$70
[ctrl-Q] DC1	17	$11	1	49	$31	Q	81	$51	q	113	$71
[ctrl-R] DC2	18	$12	2	50	$32	R	82	$52	r	114	$72
[ctrl-S] DC3	19	$13	3	51	$33	S	83	$53	s	115	$73
[ctrl-T] DC4	20	$14	4	52	$34	T	84	$54	t	116	$74
[ctrl-U] NAK	21	$15	5	53	$35	U	85	$55	u	117	$75
[ctrl-V] SYN	22	$16	6	54	$36	V	86	$56	v	118	$76
[ctrl-W] ETB	23	$17	7	55	$37	W	87	$57	w	119	$77
[ctrl-X] CAN	24	$18	8	56	$38	X	88	$58	x	120	$78
[ctrl-Y] EM	25	$19	9	57	$39	Y	89	$59	y	121	$79
[ctrl-Z] SUB	26	$1A	:	58	$3A	Z	90	$5A	z	122	$7A
ctrl-[ESC	27	$1B	;	59	$3B	[	91	$5B	{	123	$7B
[ctrl-\] FS	28	$1C	<	60	$3C	\	92	$5C	|	124	$7C

ctrl-] GS	29	$1D	=	61	$3D	]	93	$5D	}	125	$7D
[ctrl-^] RS	30	$1E	>	62	$3E	^	94	$5E	~	126	$7E
[ctrl-_] US	31	$1F	?	63	$3F	_	95	$5F	DEL	127	$7F

Index

A

ALM 18

C

CC 139

D

DEDT 48
derivative (DERV) 867
DFF 947
discrete 2-state device (D2SD) 38
discrete 3-state device (D3SD) 23

E

enhanced select (ESEL) 895
Equipment diagram instructions 1022
equipment phase 961, 970, 972, 979, 989, 993, 996, 1000, 1004
- equipment phase clear failure - PCLF 970
- equipment phase command - PCMD 972
- equipment phase external request - PXRQ 979
- equipment phase failure - PFL 989
- equipment phase instructions 961
- equipment phase new parameters - PRNP 993
- equipment phase override command - POVR 996
- equipment phase paused - PPD 1000
- phase state complete (PSC) 1004

Equipment Sequence Instruction examples 1029, 1030, 1031, 1032
Equipment Sequence Instructions 1011, 1013, 1015, 1017, 1019, 1022

F

Faceplate Control Properties dialog 1085, 1086, 1087, 1088
- Display tab 1086
- faceplates 1085
- Font tab 1087
- General tab 1085
- Locale tab 1088

FGEN 53
Filter Instructions 867
function block 1035, 1036, 1037, 1038, 1042, 1045, 1048, 1085
- attributes 1035, 1036, 1037, 1038, 1042, 1045
- faceplates 1085
- latching data 1036
- order of execution 1038
- program/operator control 1045
- timing modes 1042

H

HLL 903
HPF 871

I

IMC 178
INTG 817

J

JKFF 951

L

LDL2 887
LDLG 59
LPF 876

M

MAVE 927
MAXC 934
MINC 937
MMC 198
MSTD 940
MUX 907

N

NTCH 882

P

PATT 962
PCLF 970

PCMD 972
PDET 967
PFL 989
PI 823
PIDE 64
PMUL 834
POSP 97
POVR 996
PPD 1000
PRNP 993
PSC 1004
PXRQ 979

R

ramp/soak (RMPS) 105
reset dominant (RESD) 954
Result Codes for Equipment Sequence Instructions 1026, 1027, 1028, 1029
RLIM 911

S

scale (SCL) 119
S-Curve (SCRV) 841
second-order controller (SOC) 849
SEL 915
selected negate (SNEG) 918
selected summer (SSUM) 921
set dominant (SETD) 956
SOC 849
SRTP 123

T

totalizer (TOT) 130

U

UPDN 858

Rockwell Automation support

Use these resources to access support information.

Technical Support Center	Find help with how-to videos, FAQs, chat, user forums, and product notification updates.	rok.auto/support
Knowledgebase	Access Knowledgebase articles.	rok.auto/knowledgebase
Local Technical Support Phone Numbers	Locate the telephone number for your country.	rok.auto/phonesupport
Literature Library	Find installation instructions, manuals, brochures, and technical data publications.	rok.auto/literature
Product Compatibility and Download Center (PCDC)	Get help determining how products interact, check features and capabilities, and find associated firmware.	rok.auto/pcdc

Documentation feedback

Your comments help us serve your documentation needs better. If you have any suggestions on how to improve our content, complete the form at rok.auto/docfeedback.

Waste Electrical and Electronic Equipment (WEEE)

At the end of life, this equipment should be collected separately from any unsorted municipal waste.

Rockwell Automation maintains current product environmental information on its website at rok.auto/pec.

Allen-Bradley, expanding human possibility, Logix, Rockwell Automation, and Rockwell Software are trademarks of Rockwell Automation, Inc.

EtherNet/IP is a trademark of ODVA, Inc.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

Rockwell Otomayson Ticaret A.Ş. Kar Plaza İş Merkezi E Blok Kat:6 34752, İçerenkÖy, İstanbul, Tel: +90 (216) 5698400 EEE YÖnetmeliğine Uygundur

Connect with us.

AMERICAS: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
EUROPE/MIDDLE EAST/AFRICA: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
ASIA PACIFIC: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication 1756-RM006L-EN-P - September 2020
Supersedes Publication 1756-RM006K-EN-P - November 2018

Copyright © 2020 Rockwell Automation Technologies, Inc. All Rights Reserved. Printed in the U.S.A.