Enhanced PID (PIDE)

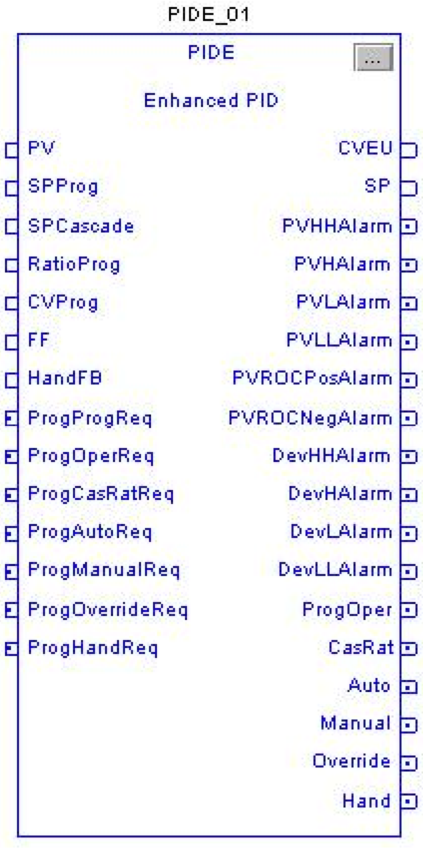
The PIDE instruction provides enhanced capabilities over the standard PID instruction. The instruction uses the velocity form of the PID algorithm. The gain terms are applied to the change in the value of error or PV, not the value of error or PV.

Available Languages

Relay Ladder

This instruction is not available in relay ladder logic.

Function Block



Structured Text

PIDE(PIDE_tag);

Operands

Function Block

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

Structured Text

Operand	Type	Format	Description
PIDE tag	PIDE_ENHANCED	structure	PIDE structure

D2SD Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	**Function Block:** Enable input. If cleared, the instruction does not execute and outputs are not updated.

		Default is set. **Structured Text:** No effect. The instruction executes.
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 set, it indicates the input signal has an error. Default is cleared = "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 set and UseRatio is cleared, then SP = SPCascade. This is typically the CVEU of a primary loop. If CascadeRatio and UseRatio are set, 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
UseRatio	BOOL	Allow ratio control permissive. Set to enable ratio control when in Cascade/Ratio mode. Default is cleared.
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 set, CVFault indicates an error on the output module and the instruction sets the appropriate bit in Status. Default is cleared = "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 cleared.
CVInitValue	REAL	CVEU initialization value, scaled in CVEU units. When CVInitializing is set, 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 set. 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 set, 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 set, the instruction sets the appropriate bit in Status and limits the CV value. Valid = 0.0 to 100.0 Default = 0.0
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 valu CV from the previous execution. CVPrevious is ignored when in manual, overric hand mode or when CVInitializing is set. If CVPrevious < 0 or > 100, or < CVLL or > CVHLimit when in Auto or cascade/ratio mode, the instruction sets the appropriate bit in Status and limits the CV_{n-1} value. Valid = 0.0 to 100.0 Default = 0.0
CVSetPrevious	BOOL	Request to use CVPrevious. If set, CV_{n-1} = CVPrevious. Default is cleared.
CVManLimiting	BOOL	Limit CV in manual mode request. If Manual mode and CVManLimiting is set, CV is limited by the CVHLimit and

		CVLLimit values. Default is cleared.
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 set. 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 ≤ 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 set. 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 ≤ 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 set. 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
FFPrevious	REAL	FF_{n-1} value. If FFSetPrevious is set, the instruction sets FF_{n-1} = FFPrevious. FF the value of 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 set, FF_{n-1} = FFPrevious. Default is cleared.
HandFB	REAL	CV Hand feedback value. CV equals this value when in Hand mode and HandFBFault is cleared (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 set, HandFBFault indicates an error on the input module and the instruction sets the appropriate bit in Status. Default is cleared = "good health"

WindupHIn	BOOL	Windup high request. When set, the CV is not allowed to increase in value. This signal is typically obtained from the WindupHOut output from a secondary loop. Default is cleared.
WindupLIn	BOOL	Windup low request. When set, the CV is not allowed to decrease in value. This signal is typically obtained from the WindupLOut output from a secondary loop. Default is cleared.
ControlAction	BOOL	Control action request. Set to calculate error as E = PV - SP; clear to calculate error as E = SP - PV. Default is cleared.
DependIndepend	BOOL	Dependent/independent control request. When set, use the dependent form of the PID equation; when cleared, use the independent form of the equations. Default is cleared.
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 of 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
PVEProportional	BOOL	Proportional PV control request. When set, calculate the proportional term (DeltaPTerm) using the change in process variable (PVPercent). When cleared, use the change in error (EPercent). Default is cleared.
PVEDerivative	BOOL	Derivative PV control request. When set, calculate the derivative term (DeltaDTerm) using the change in process variable (PVPercent). When cleared, use the change in error (EPercent). Default is set
DSmoothing	BOOL	Derivative Smoothing request. When set, 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 cleared.
PVTracking	BOOL	SP track PV request. Set to cause SP to track PV when in manual mode. Ignored when in Cascade/Ratio or Auto mode. Default is cleared.
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 disable zero crossing for the deadband calculation. Default is cleared.
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 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
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 allow Cascade/Ratio mode to be selected using either ProgCascadeRatioReq or OperCascadeRatioReq. Default is cleared.
ManualAfterInit	BOOL	Manual mode after initialization request. When set, the instruction is placed in Manual mode when CVInitializing is set, unless the current mode is Override or Hand. When ManualAfterInit is cleared, the instruction's mode is not changed, unless requested to do so. Default is cleared.
ProgProgReq	BOOL	Program program request. Set by the user program to request Program control. Ignored if ProgOperReq is set. Holding this set and ProgOperReq cleared locks the instruction in Program control. When ProgValueReset is set, the instruction clears the input each execution. Default is cleared.
ProgOperReq	BOOL	Program operator request. Set by the user program to request Operator control. Holding this set locks the instruction in Operator control. When ProgValueReset is set, the instruction clears the input each execution. Default is cleared.
ProgCasRatReq	BOOL	Program cascade/ratio mode request. Set by the user program to request Cascade/Ratio mode. When ProgValueReset is set, the instruction clears the input each execution. Default is cleared.
RogoAutoReq	BOOL	Program auto mode request. Set by the user program to request Auto mode. When ProgValueReset is set, the instruction clears the input each execution. Default is cleared.
ProgManualReq	BOOL	Program manual mode request. Set by the user program to request Manual mode. When ProgValueReset is set, the instruction clears the input each execution. Default is cleared.
ProgOverrideReq	BOOL	Program override mode request. Set by the user program to request Override mode. When ProgValueReset is set, the instruction clears the input each execution. Default is cleared.
ProgHandReq	BOOL	Program hand mode request. Set 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 set, the instruction clears the input each execution. Default is cleared.

OperProgReq	BOOL	Operator program request. Set by the operator interface to request Program control. The instruction clears this input each execution. Default is cleared.
OperOperReq	BOOL	Operator operator request. Set by the operator interface to request Operator control. The instruction clears this input each execution. Default is cleared.
OperCasRatReq	BOOL	Operator cascade/ratio mode request. Set by the operator interface to request Cascade/ Ratio mode. The instruction clears this input each execution. Default is cleared.
OperAutoReq	BOOL	Operator auto mode request. Set by the operator interface to request Auto mode. The instruction clears the input each execution. Default is cleared.
OperManualReq	BOOL	Operator manual mode request. Set by the operator interface to request Manual mode. The instruction clears the input each execution. Default is cleared.
ProgValueReset	BOOL	Reset program control values. When set, all the program request inputs are cleared by the instruction each execution. When set and in Operator control, the instruction sets SPProgram = SP and CVProgram = CV. Default is cleared.
TimingMode	DINT	Selects timing 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 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	Enable output.
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. Arithmetic flags are set for this output. 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 set. Otherwise this value is limited by 0 and 100 percent. Arithmetic flags are set for this output.
CVInitializing	BOOL	Initialization mode indicator. CVInitializing is set when CVInitReq is set, during instruction first scan, and on a set to cleared transition of CVHealth (bad to good). CVInitializing is cleared after the instruction has been initialized and CVInitReq is cleared.

CVHAlarm	BOOL	CV high alarm indicator. Set when the calculated value of CV > 100 or CVHLimit.
CVLAlarm	BOOL	CV low alarm indicator. Set when the calculated value of CV < 0 or CVLLimit.
CVROCAlarm	BOOL	CV rate of change alarm indicator. Set 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 when the SP > SPHLimit.
SPLAlarm	BOOL	SP low alarm indicator. Set 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 when not in Cascade/Ratio mode or when CVInitializing is set. This signal is normally used by the CVInitReq input of a primary PID loop.
WindupHOut	BOOL	Windup high indicator. Set when either a SP high, CV high, or CV low limit (depending on the control action) 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 when either a SP, CV high, or CV low limit (depending on the control action) 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 when Ratio > RatioHLimit.
RatioLAlarm	BOOL	Ratio low alarm indicator. Set when Ratio < RatioLLimit.
ZCDeadbandOn	BOOL	Zero crossing deadband indicator. When set the value of CV does not change. If ZCOff is set, then ZCDeadbandOn is set when | E | is within the ZCDeadband range. If ZCOff is cleared, then ZCDeadbandOn is set when | E | crosses zero and remains within the ZCDeadband range. ZCDeadbandOn is cleared when | E | exceeds the deadband range or when ZCDeadband = 0.
PVHHAlarm	BOOL	PV high-high alarm indicator. Set when PV ≥ PVHHLimit. Cleared when PV < (PVHHLimit - PVDeadband)
PVHAlarm	BOOL	PV high alarm indicator. Set when PV ≥ PVHLimit. Cleared when PV < (PVHLimit - PVDeadband)
PVLAlarm	BOOL	PV low alarm indicator. Set when PV ≤ PVLLimit. Cleared when PV > (PVLLimit + PVDeadband)
PVLLAlarm	BOOL	PV low-low alarm indicator. Set when PV ≤ PVLLLimit. Cleared when PV > (PVLLLimit + PVDeadband)
PVROCPosAlarm	BOOL	PV positive rate-of-change alarm indicator. Set when calculated PV rate-of-change ≥ PVROCPosLimit.
PVROCNegAlarm	BOOL	PV negative rate-of-change alarm indicator. Set when calculated PV rate-of-change ≤ (PVROCNegLimit x -1).
DevHHAlarm	BOOL	Deviation high-high alarm indicator. Set when PV ≥ (SP + DevHHLimit). Cleared when PV < (SP + DevHHLimit - DevDeadband)
DevHAlarm	BOOL	Deviation high alarm indicator. Set when PV ≥ (SP + DevHLimit). Cleared when

		PV < (SP + DevHLimit - DevDeadband)
DevLAlarm	BOOL	Deviation low alarm indicator. Set when PV ≤ (SP - DevLLimit). Cleared when PV > (SP - DevLLimit + DevDeadband)
DevLLAlarm	BOOL	Deviation low-low alarm indicator. Set when PV ≤ (SP - DevLLLimit). Cleared when PV > (SP - DevLLLimit + DevDeadband)
ProgOper	BOOL	Program/operator control indicator. Set when in Program control. Cleared when in Operator control.
CasRat	BOOL	Cascade/ratio mode indicator. Set when in the Cascade/Ratio mode.
Auto	BOOL	Auto mode indicator. Set when in the Auto mode.
Manual	BOOL	Manual mode indicator. Set when in the Manual mode.
Override	BOOL	Override mode indicator. Set when in the Override mode.
Hand	BOOL	Hand mode indicator. Set 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.
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	Low limit < 0 or High limit < low limit.
CVProgInv (Status1.12)	BOOL	CVProg < 0 or CVProg > 100, or CVProg < CVLLimit or CVProg > CVHLimit when CVManLimiting is set. 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 set. 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 < CVLLimit or > CVHLimit when in aut cascade/ratio mode. The instruction uses the limited value for CV_{n-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.
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 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.
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 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.
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 (i.e. CV_{n-1}). When CVSetPrevious is set, CVPrevious = CV_{n-1}. This lets you preset CV_{n-1} to a specified value before computing the value of CV.

$$CalculatedCV = CV_{n-1} + \Delta PTerm + \Delta ITerm + \Delta DTerm$$

Monitoring the PIDE Instruction

There is an operator faceplate available for the PIDE instruction.

Autotuning the PIDE Instruction

The RSLogix 5000 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 RSLogix 5000 software. 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 RSLogix 5000 software, but you need an activation key to enable it. The autotuner is only supported in function block programming; it is not available in relay ladder or structured text programming.

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

Arithmetic Status Flags

Arithmetic status flags are set by the CV output.

Fault Conditions

None

Execution

Condition	Function Block Action	Structured Text Action:
prescan	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 CVEUMax or CVEUMin. When the instruction calculates CV as the corre 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 mode is not Override or Hand mode, the instruction changes to Manual mode. If ManualAfterInit is cleared t 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.	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.

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.

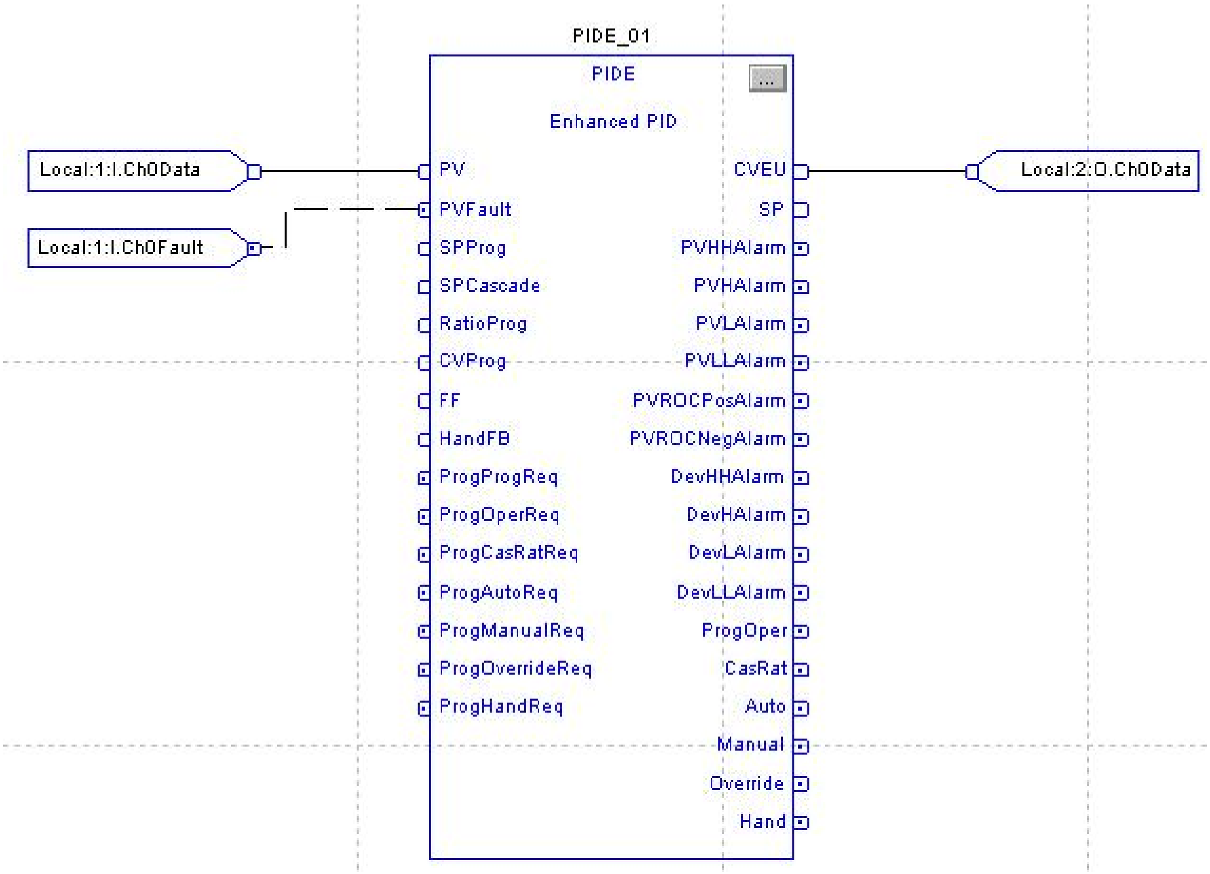
Example

Example 1

The easiest way to implement a PIDE instruction is to create an 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

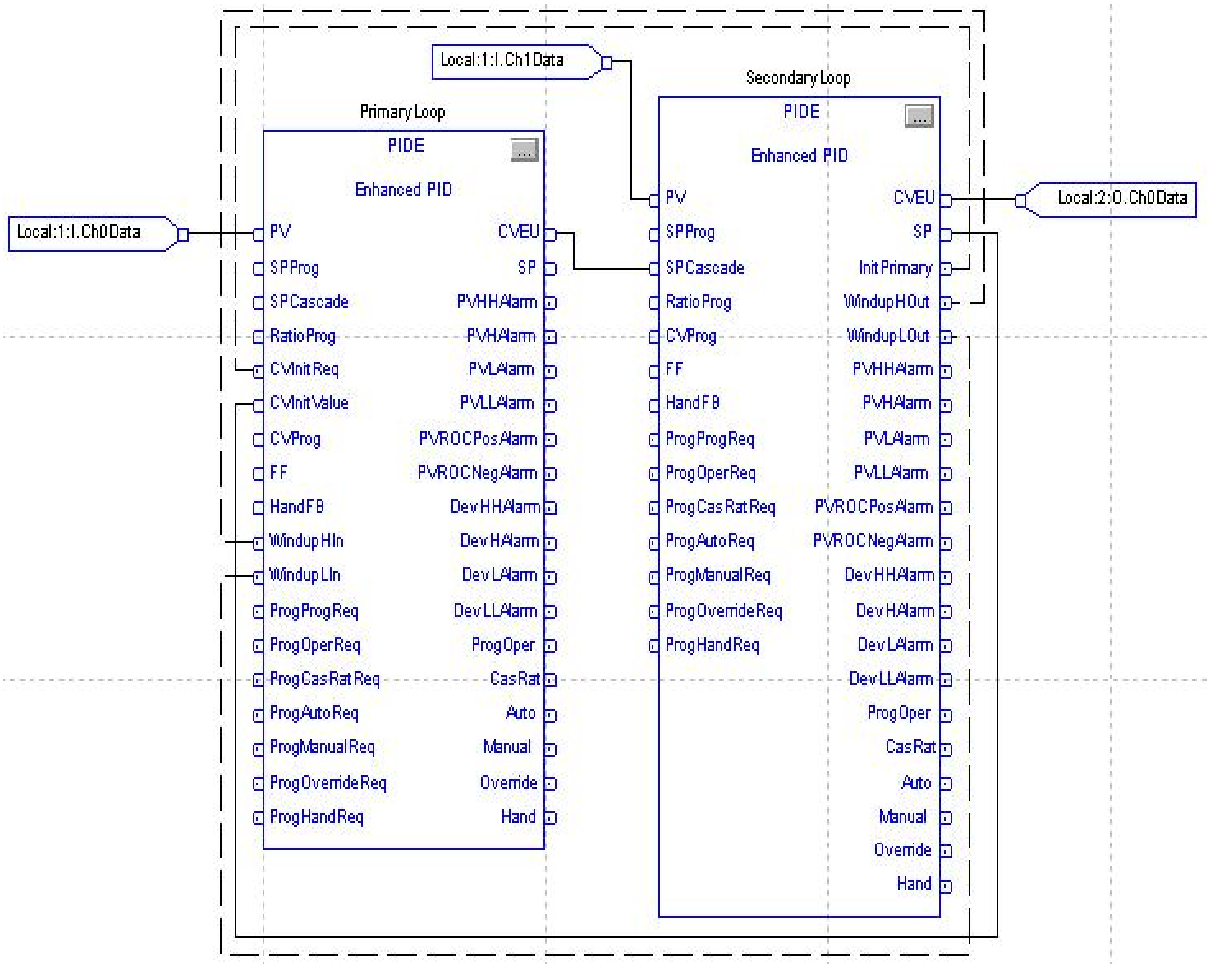
```
PIDE_01.PV := Local:1:I.Ch0Data;
PIDE_01.PVFault := Local:1:I.Ch0Fault;
PIDE(PIDE_01);
Local:2:)O.Ch0Data :=PIDE_01.CVEU;
```

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.CH0Data;
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.

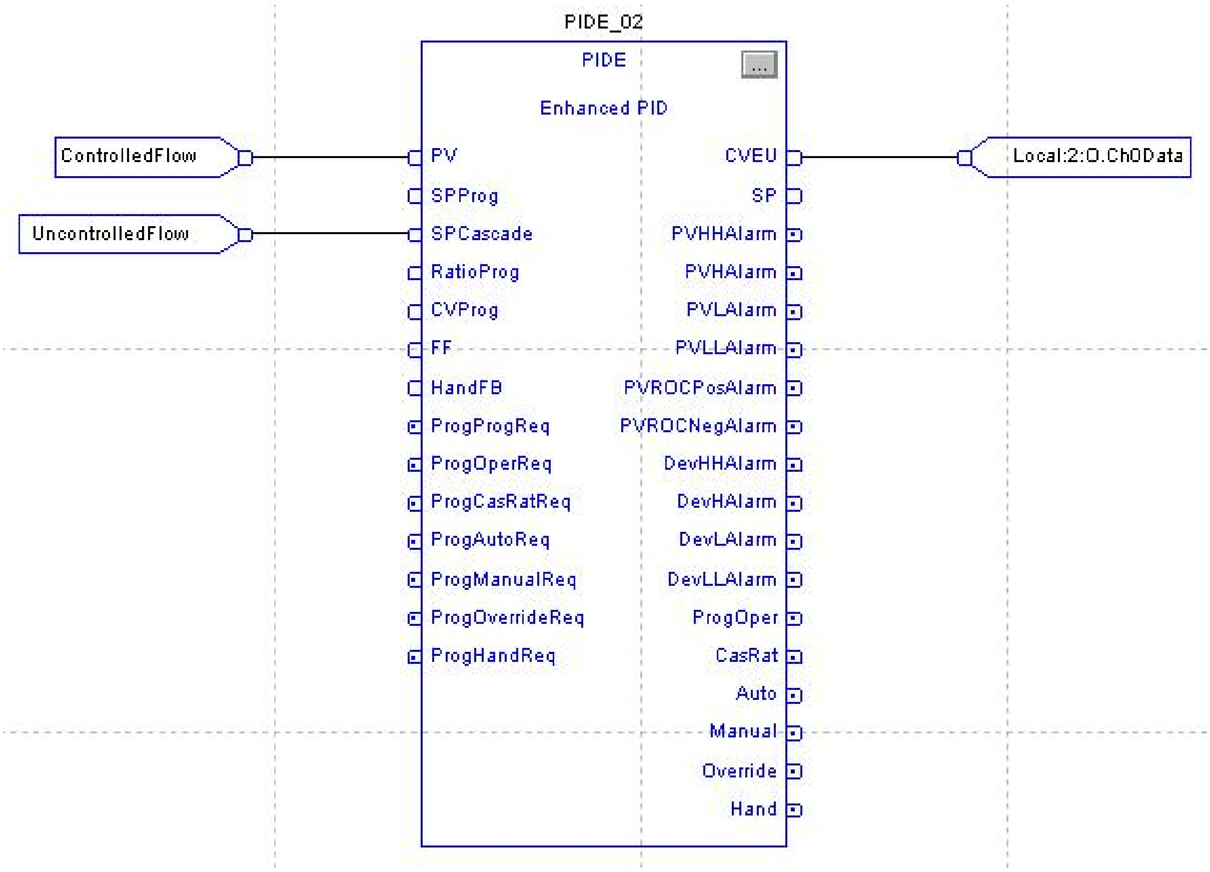
To set up a pair of cascaded PIDE instructions, set the *AllowCasRat* input parameter in the secondary loop. This allows the secondary loop to be placed into Cascade/Ratio mode. Next, wire the *CVEU* from the primary loop into the *SPCascade* parameter on the secondary loop. The *SPCascade* value is 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

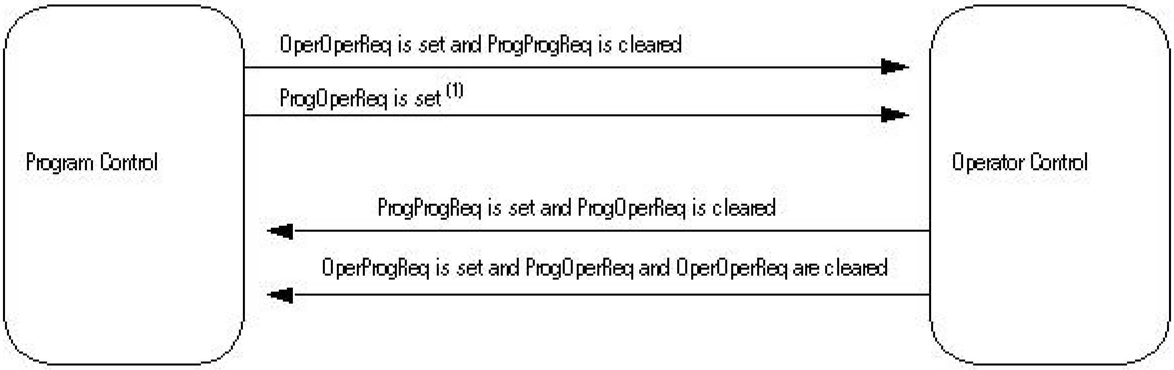
```
PIDE_01.PV := ControlledFlow;
```

```
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.
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.

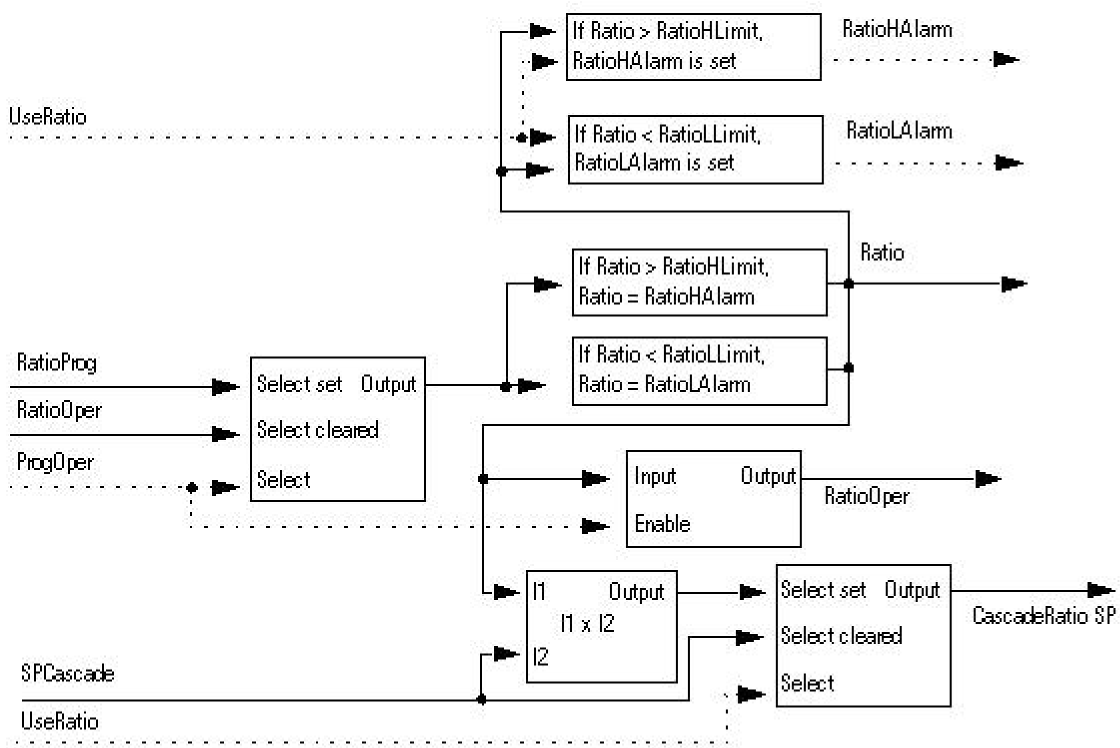
The Cascade/Ratio, Auto, and Manual modes can be controlled by a user program when in Program control or by an operator interface when in Operator control. The Override and Hand modes have a mode request input that can only be controlled by a user program; these inputs operate in both Program and Operator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

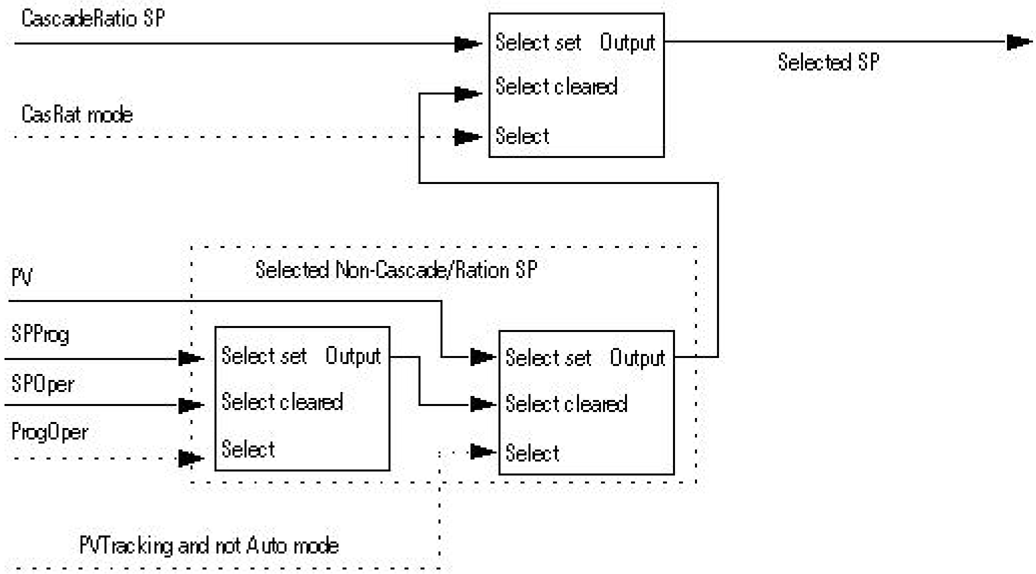
Cascade/Ratio SP

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

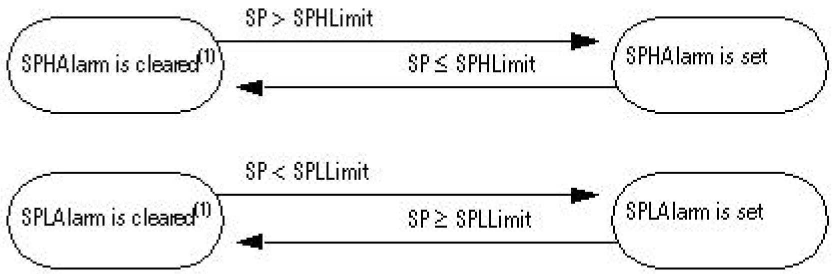


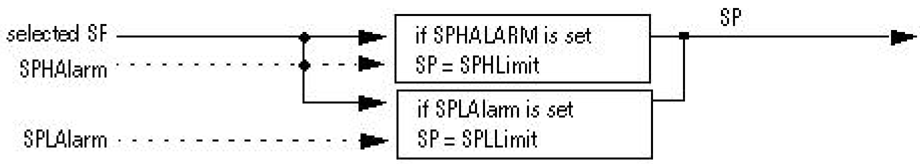
Current SP

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

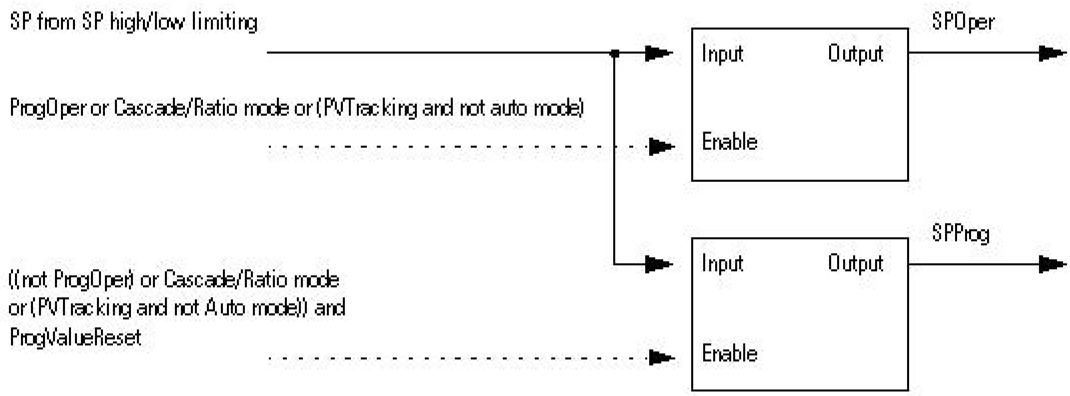


SP High/Low Limiting
The high-to-low alarming algorithm compares SP to the SPHLimit and SPLLimit alarm limits. SPHLimit cannot be greater than PVEUMax and SPLLimit cannot be less than PVEUMin.



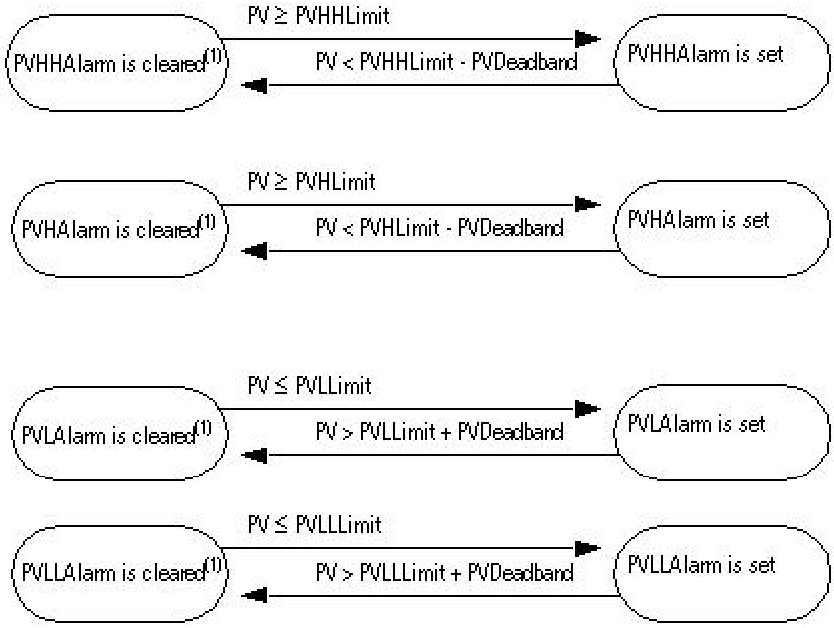


Updating the SPOper and SPProg Values
The PIDE instruction makes SPOper = SP or SPProg = SP to obtain bumpless control switching between Program and Operator control or when switching from Cascade/Ratio mode.



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.

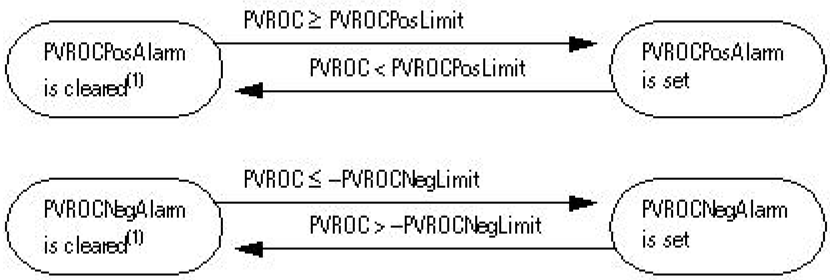
The ROC calculation is only performed when the PVROCPeriod has expired. The rate-of-change is calculated as:

ElapsedROCPeriod = ElapsedROCPeriod + ElapsedTimeSinceLastExecution

If ElapsedROCPeriod ≥ PVROCPeriod then:

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

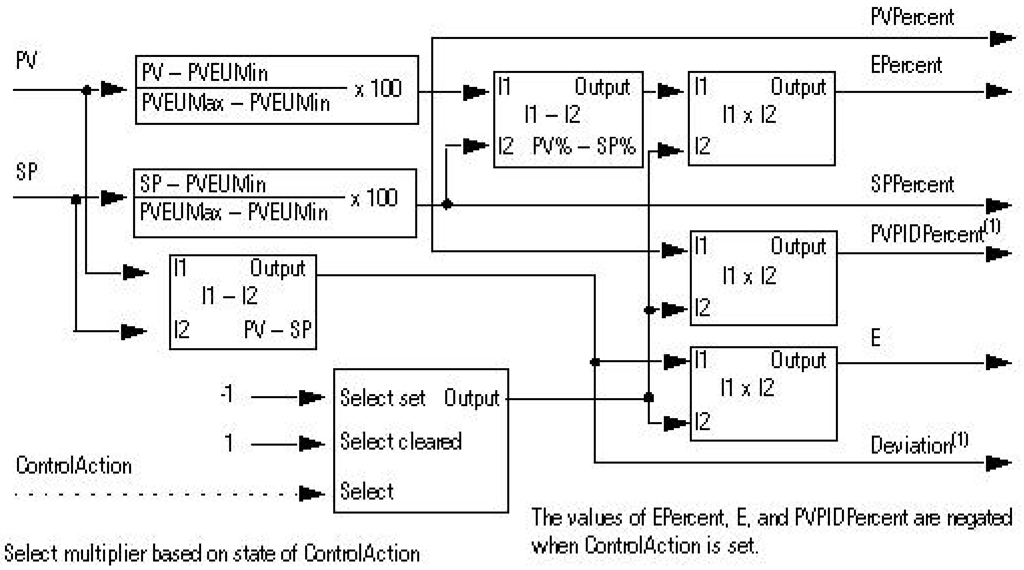
Once PVROC has been calculated, the PV ROC alarms are determined as follows:



(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.

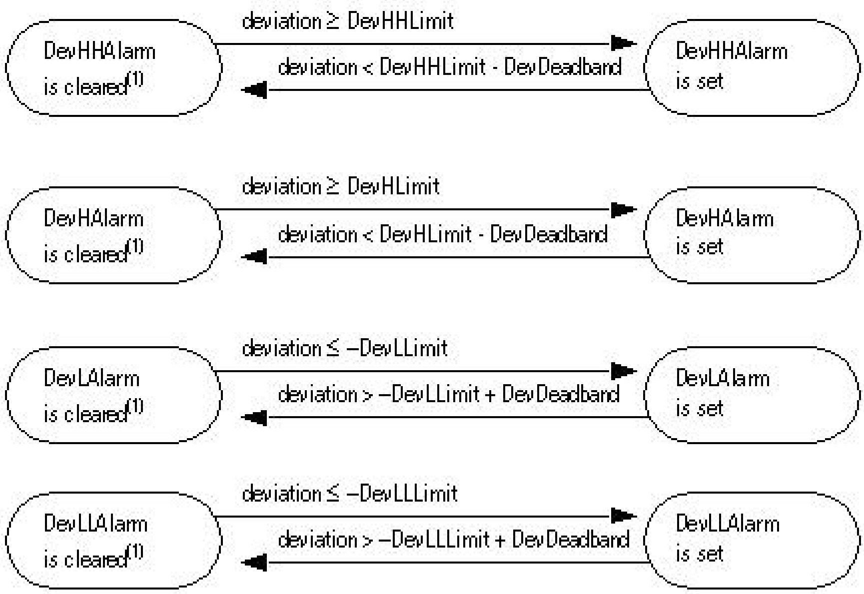


(1) PVPIDPercent and Deviation are internal parameters used by the PID control algorithm.

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.

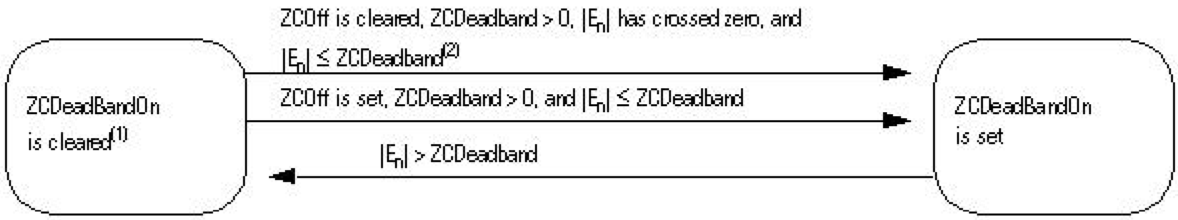
The high-high to low-low alarming algorithm compares the deviation to deviation alarm limits and the deviation alarm limits plus or minus the deadband.

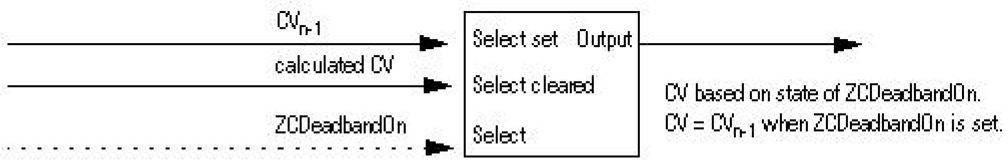


(1) During instruction first scan, the instruction clears the deviation alarm outputs. The instruction also clears the deviation alarm outputs and disables the alarming algorithm when PVFaulted or PVSpanInv is set.

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. E_n >= 0 and E_{n-1} < 0 or when E_n <= 0 and E_{n-1} > 0), and | E_n | <= ZCDeadband, the instruction sets ZCDeadbandOn.

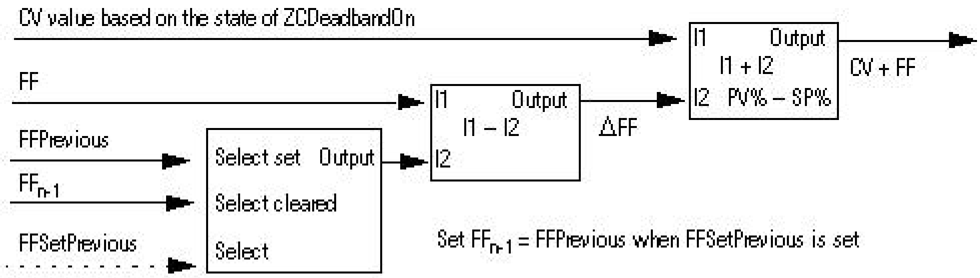
(2) On the transition to Auto or Cascade/Ratio mode, the instruction sets $E_{n-1} = E_n$.

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

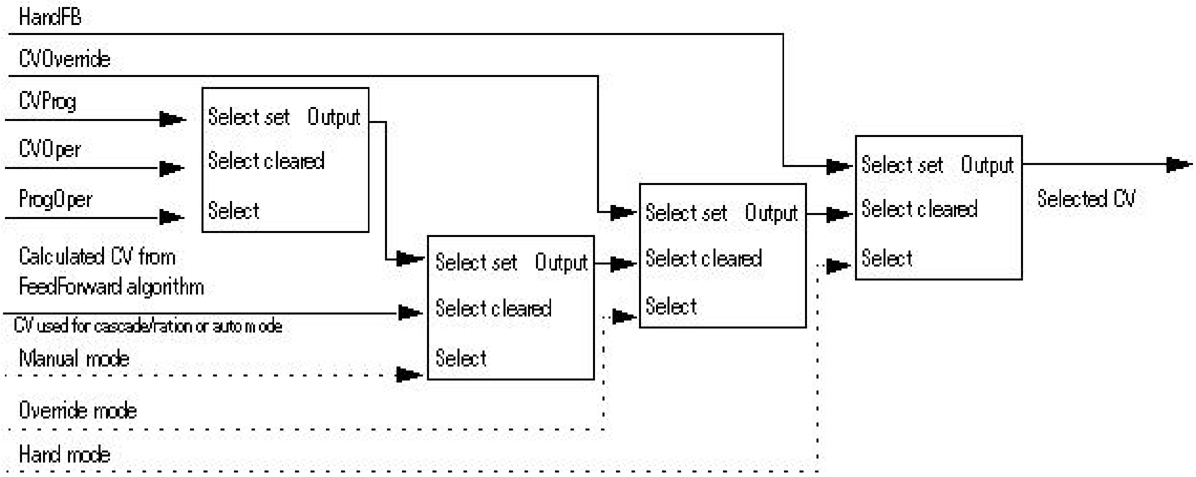
Feedforward Control

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



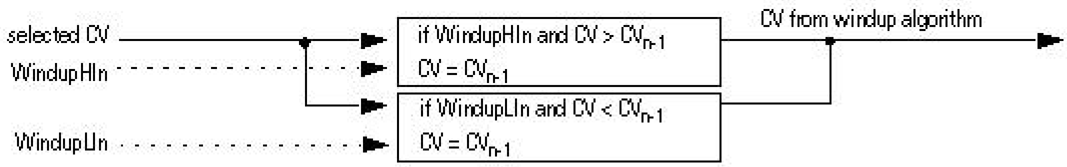
Selecting the Control Variable

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



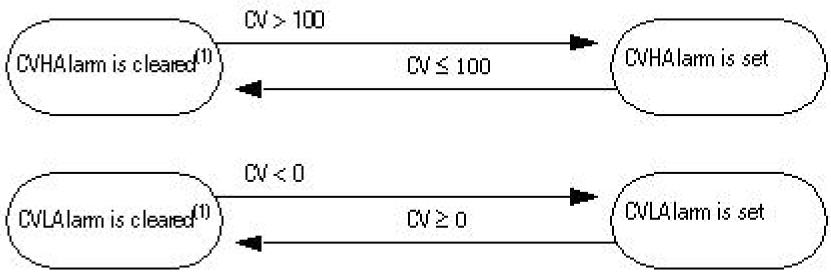
CV Windup Limiting

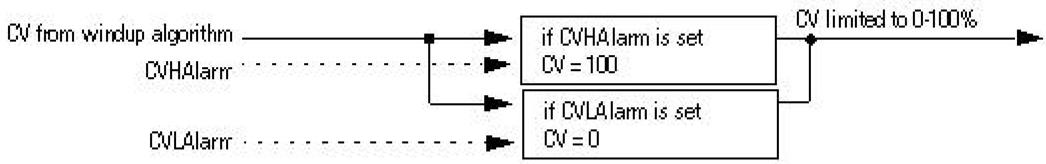
Limit the CV such that its value cannot increase when WindupHIn is set or decrease when WindupLIn is set. These inputs are typically the WindupHOut or WindupLOut outputs from a secondary loop. The WindupHIn and WindupLIn inputs are ignored if CVInitializing, CVFault, or CVEUSpanInv is set.



CV Percent Limiting

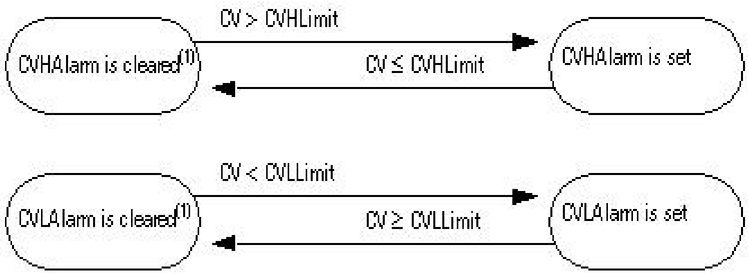
The following diagram illustrates how the instruction determines CV percent limiting.

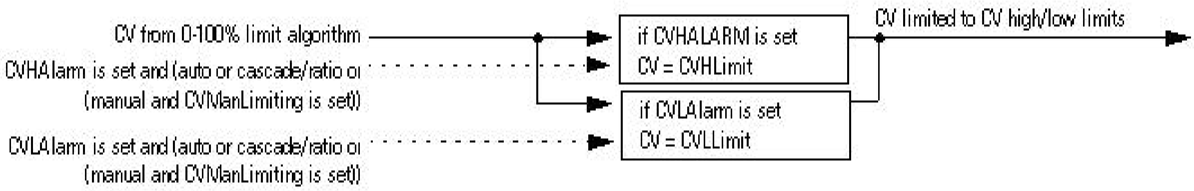




(1) During instruction first scan, the instruction clears the alarm outputs.

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.

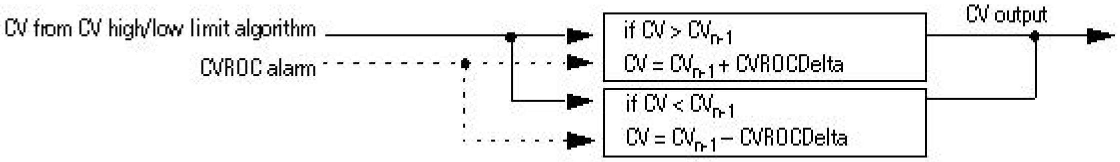
The CV rate-of-change is calculated as:

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

$$CVROCDelta = CVROCLimit \times DeltaT$$

where DeltaT is in seconds.

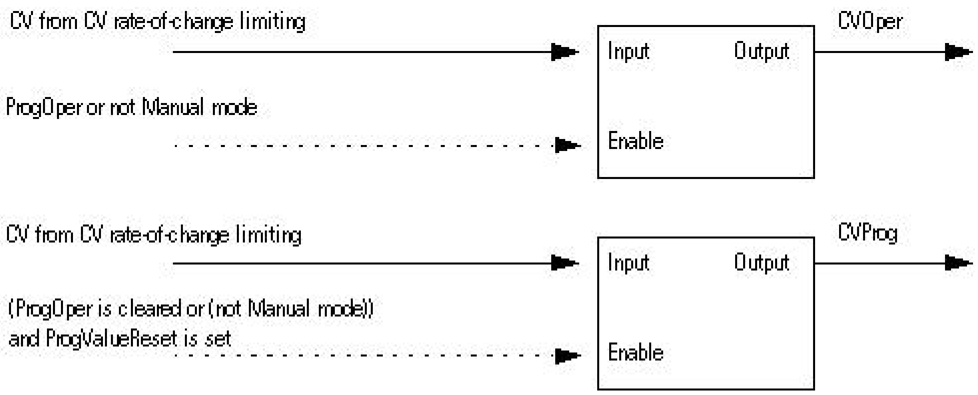
Once CV rate-of-change has been calculated, the CV rate-of-change alarms are determined as follows:



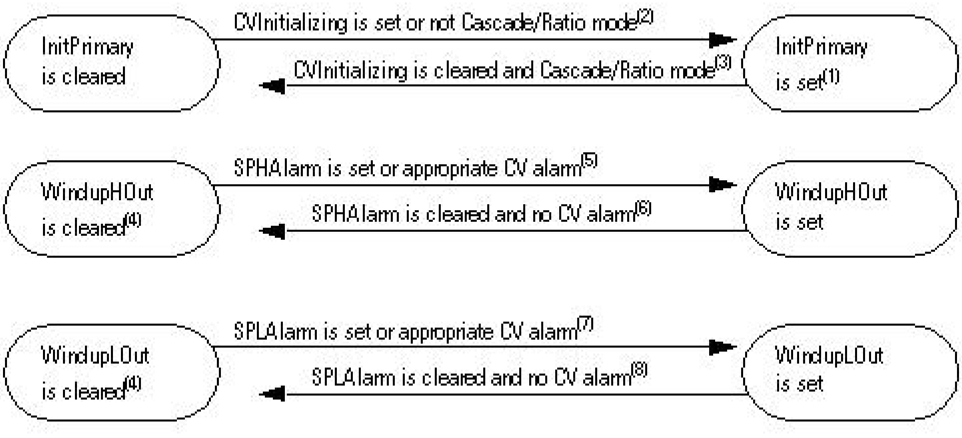
(1) During instruction first scan, the instruction clears the alarm output. The instruction also clears the alarm output and disables the CV rate-of-change algorithm when CVInitializing is set.

(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 set **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).
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 model ▪ 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