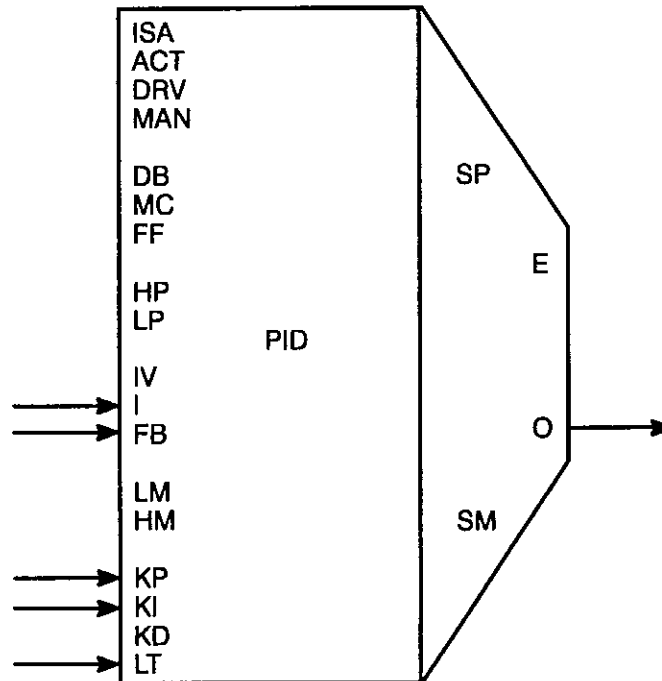


50.0 PID

This function can be used in AutoMax Control Block tasks only. It cannot be used in UDC Control Block tasks.



Function

If MANUAL is true then

ERROR = 0
INTEGRATOR = INITIAL_VALUE
SUM = INTEGRATOR

Else

(automatic mode)
If REVERSE_ACTION is true then
ERROR = FEEDBACK - INPUT

Else

ERROR = INPUT - FEEDBACK
P_TERM = ERROR * KP
I_TERM = (ERROR + ERROR (n-1)) * KI * LOOP_TIME / 2
If abs (I_TERM) < DEAD_BAND then
I_TERM = 0
If I_TERM > 0 and HOLD_PLUS is true then
I_TERM = 0
If I_TERM < 0 and HOLD_MINUS is true then
I_TERM = 0
INTEGRATOR = INTEGRATOR + I_TERM
If INTEGRATOR > LIMIT_PLUS then
INTEGRATOR = LIMIT_PLUS
If INTEGRATOR < LIMIT_MINUS then
INTEGRATOR = LIMIT_MINUS

```

If DERIVATIVE = TRUE then
  IF ACTION = FALSE
    D_ERROR = -FEEDBACK
  Else D_ERROR = FEEDBACK
  End If
Else
  If ACTION = FALSE then
    D_ERROR = INPUT - FEEDBACK
  Else
    D_ERROR = FEEDBACK - INPUT
  End If
End If
Else
  D_ERROR = ERROR
  D_TERM = (D_ERROR - D_ERROR (n-1) ) * KD
  SUM = INTEGRATOR + P_TERM + D_TERM

```

(evaluated if in manual or automatic mode)

```

SUM = SUM + FEED_FORWARD
CHANGE = SUM-OUTPUT (n-1)
If CHANGE > MAX_CHANGE then
  CHANGE = MAX_CHANGE
Else if CHANGE < MAX_CHANGE then
  CHANGE = -MAX_CHANGE
OUTPUT = OUTPUT + CHANGE

```

```

If OUTPUT >= LIMIT_PLUS then
  OUTPUT = LIMIT_PLUS, SATURATED_PLUS = 1

```

```

Else if OUTPUT <= LIMIT_MINUS then
  OUTPUT = LIMIT_MINUS, SATURATED_MINUS = 1

```

```

Else

```

```

SATURATED_PLUS = 0, SATURATED_MINUS = 0

```

Note: If ISA is true then $KI = KI * KP$ and $KD = KD * KP$.


```

FEED_FORWARD=feed_forward%,      &
INPUT=input%,                      &
FEEDBACK=feedback%,               &
DEAD_BAND=dead_band%,             &
MAX_CHANGE=max_change%,           &
LIMIT_PLUS=limit_plus%,           &
LIMIT_MINUS=limit_minus%,         &
HOLD_PLUS=hold_plus@,             &
HOLD_MINUS=hold_minus@,          &
SATURATED_PLUS=saturated_plus@,   &
SATURATED_MINUS=saturated_minus@, &
ERROR=error%,                     &
OUTPUT=output%)

```

Inputs

ISA (ISA) =

This input selects if KI and KD are scaled by KP. If this optional input is programmed, it must be a boolean literal. If it is not programmed, it defaults to a value of FALSE. If ISA = TRUE then the KI and KD coefficients are automatically multiplied by KP. If ISA = FALSE then KI and KD are independent of KP.

ACTION (ACTION) =

This input selects how ERROR is calculated. If this optional input is programmed, it must be a boolean literal. If it is not programmed, it defaults to a value of FALSE. If ACTION = TRUE then ERROR = FEEDBACK - INPUT. If ACTION = FALSE then ERROR = INPUT - FEEDBACK.

DRV (DERIVATIVE) =

This input selects which signal to use in calculating the derivative term. If this optional input is programmed, it must be a boolean literal. If it is not programmed, it defaults to a value of FALSE.

```

If DERIVATIVE = TRUE then
  IF ACTION = FALSE
    D_ERROR = -FEEDBACK
  Else D_ERROR = FEEDBACK
  End If
Else
  If ACTION = FALSE then
    D_ERROR = INPUT - FEEDBACK
  Else
    D_ERROR = FEEDBACK - INPUT
  End If
End If

```

MAN (MANUAL) =

This input selects between manual and automatic mode of operation. If this optional input is programmed, it must be a boolean variable or literal. If it is not programmed, it defaults to a value of FALSE.

If MANUAL = TRUE then the PID block is in manual mode. Internal variables are reset to zero and the output is the sum of the INITIAL_VALUE and FEED_FORWARD.

If MANUAL = FALSE then the PID block is in automatic mode and the selected algorithm is calculated.

KP (KP) =

This input is the gain on the proportional term. It must be programmed as a real variable or constant. A decimal point must be included in a constant value. The value of KP is dimensionless.

KI (KI) =

This input is the gain on the integral term. It must be programmed as a real variable or constant. A decimal point must be included in a constant value. The units on KI are repeats per second (1/seconds).

KD (KD) =

This input is the gain on the derivative term. If this optional input is programmed, it must be programmed as a real variable or constant. A decimal point must be included in a constant value. The units on KD are seconds.

LT (LOOP_TIME) =

This input is used to tell the PID block how often it is being executed. This parameter must be specified as a constant. A variable name is not accepted. A decimal point must be included in a constant value. The units are in seconds. If the PID block is evaluated on every scan of the task, this value is the number of ticks per scan times the tick rate. If the PID block is not evaluated on every scan, this value is the time in seconds between iterations. The PID block must be called on a consistent time basis.

IV (INITIAL_VALUE) =

This input is used to define an initial value for the integrator. If this optional input is programmed, it must be an integer variable or constant. If not programmed, it defaults to a value of zero. When the PID block is in manual mode, this value is loaded into the integrator.

FF (FEED_FORWARD) -

This input is used to define a feed forward variable. If this optional input is programmed, it must be an integer variable or constant. If not programmed, it defaults to a value of zero. FEED_FORWARD is added to SUM to produce OUTPUT.

I (INPUT) =

This input is used to define the set point signal to the PID block. This parameter must be programmed as an integer variable. A constant is not accepted.

FB (FEEDBACK) =

This input is used to define the feedback signal to the PID block. This parameter must be programmed as an integer variable. A constant is not accepted.

DB (DEAD_BAND) =

This input is used to define a deadband on integral calculation. If this optional input is programmed, it must be an integer variable or constant. If not programmed, it defaults to a value of zero. If the absolute value of ERROR is less than DEAD_BAND, no change is made to the value of the integrator.

MC (MAX_CHANGE) =

This input is used to define a maximum rate of change of OUTPUT. If this optional input is programmed, it must be an integer variable or constant. If not programmed, it defaults to a value of 32767.

LP (LIMIT_PLUS) =

This input is used to define a positive limit on the INTEGRATOR and on OUTPUT. If this optional input is programmed, it must be an integer variable or constant. If not programmed, it defaults to a value of 32767. The INTEGRATOR is limited to never be greater than this value. The OUTPUT is limited to never be greater than this value.

LM (LIMIT_MINUS) =

This input is used to define a minus limit on the INTEGRATOR and on OUTPUT. If this optional input is programmed, it must be an integer variable or constant. If not programmed, it defaults to a value of -32767. The INTEGRATOR is limited to never be less than this value. The OUTPUT is limited to never be less than this value.

HP (HOLD_PLUS) =

This input is used to hold increasing the INTEGRATOR. If this optional input is programmed, it must be a boolean variable. If not programmed, it defaults to a value of FALSE. If ERROR > 0 and HOLD_PLUS is true, the INTEGRATOR is not allowed to become more positive.

HM (HOLD_MINUS) =

This input is used to hold decreasing the INTEGRATOR. If this optional input is programmed, it must be a boolean variable. If not programmed, it defaults to a value of FALSE. If ERROR < 0 and HOLD_MINUS is true, the INTEGRATOR is not allowed to become more negative.

Outputs

SP (SATURATED_PLUS) =

This output is used to indicate that OUTPUT is being limited by LIMIT_PLUS. If this optional output is programmed, it must be a boolean variable. If the calculated value for OUTPUT would be \geq LIMIT_PLUS, this output is turned on. Otherwise, it is turned off.

SM (SATURATED_MINUS) =

This output is used to indicate that OUTPUT is being limited by LIMIT_MINUS. If this optional input is programmed, it must be a boolean variable. If the calculated value for OUTPUT would be \leq LIMIT_MINUS, this output is turned on. Otherwise, it is turned off.

E (ERROR) =

This output is used to define the variable where ERROR can be displayed. If this optional output is programmed, it must be an integer variable. The value displayed will be $\text{INPUT} - \text{FEEDBACK}$ or $\text{FEEDBACK} - \text{INPUT}$ as defined by the ACTION input.

O (OUTPUT) =

This output is used to define the variable where the resulting calculation will be written. This output is required, and must be defined as an integer variable. When the PID block is in manual mode, OUTPUT is the sum of INITIAL_VALUE and FEED_FORWARD. When the PID block is in automatic mode, it is the sum of the KP, KI, and KD terms, and FEED_FORWARD.

50.1 KP Limitations

The value of KP is dimensionless. KP is limited to values between the following limits:

Low limit = 0
High limit = 9.2233717×10^{18}

The smallest value KP can represent (other than 0) is $5.4210107 \times 10^{-20}$.

When KP is programmed as a symbol, values less than zero are clamped at zero. When KP is programmed as a literal, values less than zero will cause a compiler error.

50.2 KI Limitations

KI units are 1/seconds. KI is limited to values between the following limits:

Low limit = 0
High limit = 9.2233717×10^{18}

The smallest value KI can represent (other than 0) is $5.4210107 \times 10^{-20}$.

When KI is programmed as a symbol, values less than zero are clamped at zero. When KI is programmed as a literal, values less than zero will cause a compiler error.

50.3 **KD Limitations**

KD units are seconds. KD is limited to the following values:

Low limit = 0
High limit = 9.2233717×10^{18}

The smallest value KD can represent (other than 0) is $5.4210107 \times 10^{-20}$.

When KD is programmed as a symbol, values less than zero are clamped at zero. When KD is programmed as a literal, values less than zero will cause a compiler error.

50.4 **LOOP_TIME Limitations**

LOOP_TIME units are seconds (the time between updates). LOOP_TIME is limited to the following values:

Low limit = .01 or 1.0×10^{-2} seconds
High limit = 9.2233717×10^{18} seconds

Values less than .01 will cause a compiler error.

50.5 **DEAD_BAND Limitations**

DEAD_BAND is limited to the following values:

Low limit = 0
High limit = 32767

When DEAD_BAND is programmed as a symbol, values less than zero are clamped at zero. When DEAD_BAND is programmed as a literal, values less than zero will cause a compiler error.

50.6 **MAX_CHANGE Limitations**

When MAX_CHANGE is programmed as a symbol, it is limited to the following values:

Low limit = 0
High limit = 32767

Values less than zero are clamped at zero.

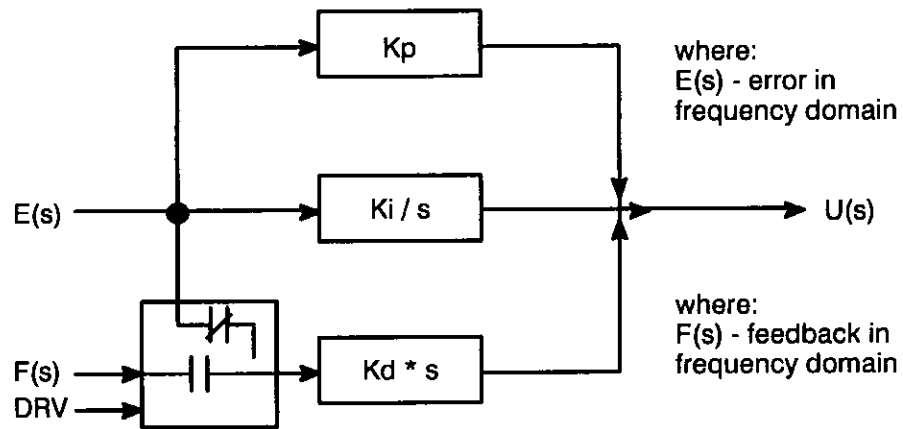
When MAX_CHANGE is programmed as a literal, it is limited to the following values:

Low limit = 1
High limit = 32767

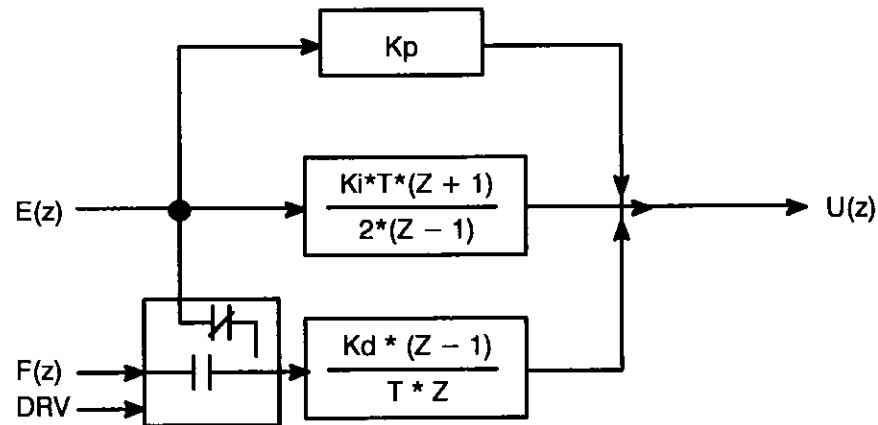
Values less than one will cause a compiler error.

Independent Mode: Derivative on Error or Feedback (n)

s - domain functional block:



Z - domain functional block:



where:
E(z) - error in z-domain
F(z) - feedback in z-domain

Difference Equation:

$$\text{Error}(n) = \begin{cases} \text{Input}(n) - \text{Feedback}(n) & \text{if action} = \text{FALSE} \\ \text{Feedback}(n) - \text{Input}(n) & \text{if action} = \text{TRUE} \end{cases}$$

$$\text{Dev_err}(n) = \begin{cases} \text{Error}(n) & \text{if derivative} = \text{FALSE} \\ \text{Feedback}(n) & \text{if derivative} = \text{TRUE} \end{cases}$$

$$\begin{aligned} \text{Incr}(n) = & \\ & \text{(P term)} \quad K_p * [\text{Error}(n) - \text{Error}(n - 1)] \\ & \text{(I term)} \quad + \quad K_1 * [\text{Error}(n) + \text{Error}(n - 1)] \\ & \text{(D term)} \quad + \quad K_2 * [\text{Dev_err}(n) - 2 * \text{Dev_err}(n-1) + \text{Dev_err}(n-2)] \\ & \quad + \quad \text{Incr_remainder}(n-1) \end{aligned}$$

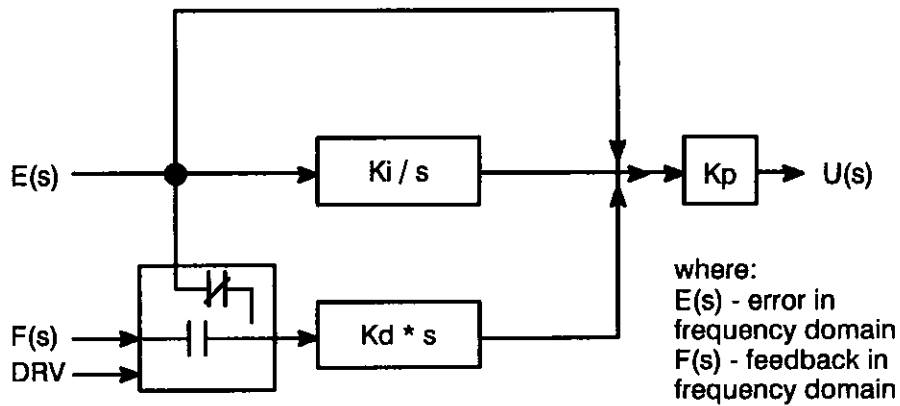
$$\text{Output}(n) = \text{Output}(n-1) + \text{Incr}(n)$$

$$\text{where: } K_1 = \frac{K_i * T}{2}$$

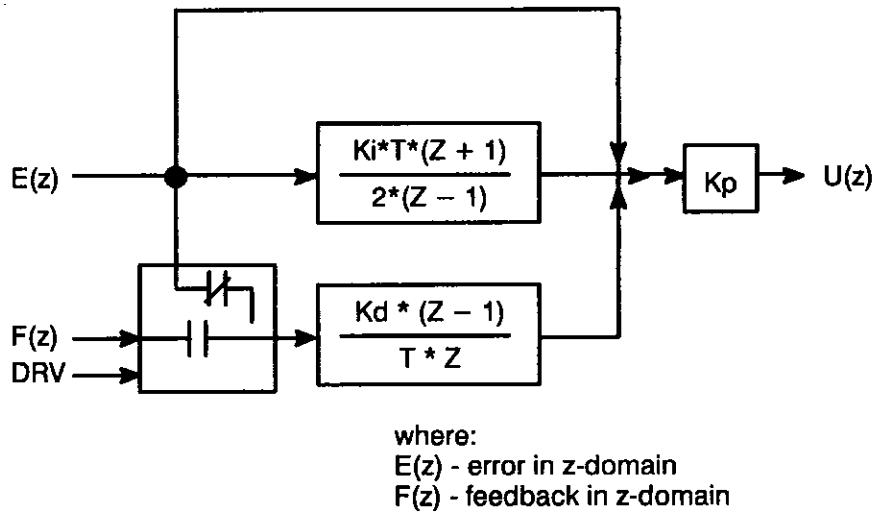
$$K_2 = \frac{K_d}{T}$$

ISA Mode: Derivative on Error(n) or Feedback (n)

s - domain functional block:



Z - domain functional block:



Difference Equation:

$$\text{Error}(n) = \begin{cases} \text{Input}(n) - \text{Feedback}(n) & \text{if action} = \text{FALSE} \\ \text{Feedback}(n) - \text{Input}(n) & \text{if action} = \text{TRUE} \end{cases}$$

$$\text{Dev_err}(n) = \begin{cases} \text{Error}(n) & \text{if derivative} = \text{FALSE} \\ \text{Feedback}(n) & \text{if derivative} = \text{TRUE} \end{cases}$$

$$\begin{aligned} \text{Incr}(n) = & K_p * ([\text{Error}(n) - \text{Error}(n - 1)] \\ & \text{(I term)} + K_1 * [\text{Error}(n) + \text{Error}(n - 1)] \\ & \text{(D term)} + K_2 * [\text{Dev_err}(n) - 2 * \text{Dev_err}(n-1) + \text{Dev_err}(n-2)] \\ & + \text{Incr_remainder}(n-1) \end{aligned}$$

$$\text{Output}(n) = \text{Output}(n-1) + \text{Incr}(n)$$

$$\text{where: } K_1 = \frac{K_i * T}{2} \qquad K_2 = \frac{K_d}{T}$$