Trusted Process Control Algorithm Software Package

Product Overview

The system architecture of the Trusted® is intended to cover as large a cross section of control and interlock applications as is possible. The control algorithms detailed in this Product Description are primarily intended to support the limited process control capabilities required for Floating Production, Storage and Offloading (FPSO) installations. The algorithms may also be used for other process control applications as required.

Control functions require the use of floating point values. This ultimately limits the integrity that can be attributed to these functions because of the introduction of an element of 'loss of precision' and the general inability to exhaustively verify floating point capabilities.

The functions are integrated into the IEC 61131 TOOLSET execution environment. This environment is 'Commercial-Off-The-Shelf (COTS)' and designed to commercial standards. The overall system achieves higher levels of integrity by the use of additional on-line monitoring and internal state control.

The combination of the above factors indicate that the target integrity for the basic process control functions should be SIL 1 (AK3). These functions should therefore NOT be used within elements of application programs intended for SIL 3 (AK6) use.

The control algorithms are executed within the Trusted Controller as part of the standard application program and are divided into 'functions' and 'function blocks'.

'Functions' are those algorithms with no internal state, or time dependent operation, i.e. they have no storage element. The functions process the defined number of parameters and return a single resultant state or value. Functions, therefore, have only a single operating mode, always performing the same operation.

The 'function blocks' include retentive information and may return multiple values. Function blocks will have an initial state or value for each of their outputs. The initial value may be maintained for a single or multiple application iterations, depending on the detail of the algorithms.

Page intentionally left blank

PREFACE

In no event will Rockwell Automation be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment. The examples given in this manual are included solely for illustrative purposes. Because of the many variables and requirements related to any particular installation, Rockwell Automation does not assume responsibility or reliability for actual use based on the examples and diagrams.

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

All trademarks are acknowledged.

DISCLAIMER

It is not intended that the information in this publication covers every possible detail about the construction, operation, or maintenance of a control system installation. You should also refer to your own local (or supplied) system safety manual, installation and operator/maintenance manuals.

REVISION AND UPDATING POLICY

This document is based on information available at the time of its publication. The document contents are subject to change from time to time. The latest versions of the manuals are available at the Rockwell Automation Literature Library under "Product Information" information "Critical Process Control & Safety Systems".

TRUSTED RELEASE

This technical manual applies to **Trusted Release: 3.6.1**.

LATEST PRODUCT INFORMATION

For the latest information about this product review the Product Notifications and Technical Notes issued by technical support. Product Notifications and product support are available at the Rockwell Automation Support Centre at
http://rockwellautomation.custhelp.com

At the Search Knowledgebase tab select the option "By Product" then scroll down and select the Trusted product.

Some of the Answer ID’s in the Knowledge Base require a TechConnect Support Contract. For more information about TechConnect Support Contract Access Level and Features please click on the following link:

https://rockwellautomation.custhelp.com/app/answers/detail/a_id/50871

This will get you to the login page where you must enter your login details.

IMPORTANT A login is required to access the link. If you do not have an account then you can create one using the "Sign Up" link at the top right of the web page.

DOCUMENTATION FEEDBACK

Your comments help us to write better user documentation. If you discover an error, or have a suggestion on how to make this publication better, send your comment to our technical support group at http://rockwellautomation.custhelp.com

SCOPE

This manual specifies the maintenance requirements and describes the procedures to assist troubleshooting and maintenance of a Trusted system.

WHO SHOULD USE THIS MANUAL

This manual is for plant maintenance personnel who are experienced in the operation and maintenance of electronic equipment and are trained to work with safety systems.

SYMBOLS

In this manual we will use these notices to tell you about safety considerations.

	SHOCK HAZARD: Identifies an electrical shock hazard. If a warning label is fitted, it can be on or inside the equipment.
	WARNING: Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which can cause injury or death, property damage or economic loss.
	ATTENTION: Identifies information about practices or circumstances that can cause injury or death.
	CAUTION: Identifies information about practices or circumstances that can cause property damage or economic loss.
	BURN HAZARD: Identifies where a surface can reach dangerous temperatures. If a warning label is fitted, it can be on or inside the equipment.
	This symbol identifies items which must be thought about and put in place when designing and assembling a Trusted controller for use in a Safety Instrumented Function (SIF). It appears extensively in the Trusted Safety Manual.
IMPORTANT	Identifies information that is critical for successful application and understanding of the product.
NOTE	Provides key information about the product or service.
TIP	Tips give helpful information about using or setting up the equipment.

WARNINGS AND CAUTIONS

WARNING: EXPLOSION RISK

Do not connect or disconnect equipment while the circuit is live or unless the area is known to be free of ignitable concentrations or equivalent

AVERTISSEMENT - RISQUE D'EXPLOSION

Ne pas connecter ou déconnecter l'équipement alors qu'il est sous tension, sauf si l'environnement est exempt de concentrations inflammables ou équivalente

MAINTENANCE

Maintenance must be carried out only by qualified personnel. Failure to follow these instructions may result in personal injury.

CAUTION: RADIO FREQUENCY INTERFERENCE

Most electronic equipment is influenced by Radio Frequency Interference. Caution should be exercised with regard to the use of portable communications equipment around such equipment. Signs should be posted in the vicinity of the equipment cautioning against the use of portable communications equipment.

CAUTION:

The module PCBs contains static sensitive components. Static handling precautions must be observed. DO NOT touch exposed connector pins or attempt to dismantle a module.

ISSUE RECORD

Issue	Date	Comments
6	Oct 05	Format
7	Dec 06	Example PID loop
8	Sep 07	Corrections
9	Jun 16	Converted document to Rockwell branding layout with correction of typographical errors

Page intentionally left blank

Table of Contents

1. Description 3

2. Functions 5

2.1. Square Root Extraction 5
2.2. Binary-Coded Decimal (BCD) Translation 6
2.3. Analogue Select 7
2.4. Multiple Value Averaging 8
2.5. Drum Level Control 8
2.6. Mass Flow Computation 9

3. Function Blocks 11

3.1. Analogue Scaling 11
3.2. Analogue Value Clamping 12
3.3. Low Value Select 12
3.4. High Value Select 13
3.5. Median Value Select 13
3.6. Proportional PID Function 14
3.7. Incremental PID Function 15
3.8. Deviation Alarm 18
3.9. Manual Tracking 19
3.10. Time Averaged Value 19
3.11. Rate of Change Detection 21
3.12. Analogue Value Slew 22

4. Lead/Lag Control Function 25

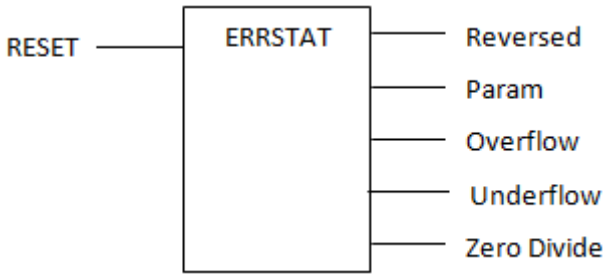
5. Sample Application Program 27

Page intentionally left blank

1. Description

Several of the basic control algorithms may generate error conditions at run-time as all function parameters are variable. This may result in reversed parameters, e.g. max < min, overflow and underflow conditions, divide by zero errors, etc. For each function, the required output under these conditions is defined and error counters are incremented. These error counters are available to the application programmer using the error statistics function block.

The function block has a single Boolean input that is used to reset the counters. Whilst this input is true, the error counters will be reset to zero. The initial value of all error counters is zero.



Data types are:

- RESET: Boolean
- All others: Integer

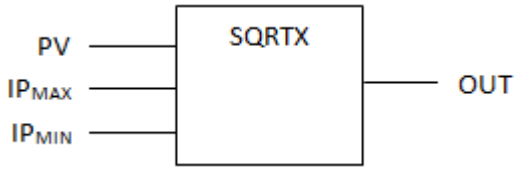
Output	Error Condition
Reversed	Pairs of function parameters are reversed, i.e. max < min. The associated function has reversed the two values in order to obtain a valid result. The count is incremented on each iteration of the function whilst the parameters are reversed.
Param	A function parameter is incorrect, e.g. out of range or not consistent with the functions required operation. The count is incremented on each iteration of the function whilst the parameter is incorrect.
Overflow	An overflow error occurred in the execution of the function. The count is incremented on each iteration of the function whilst the overflow condition occurs.

Output	Error Condition
Underflow	An underflow error occurred in the execution of the function. The count is incremented on each iteration of the function whilst the underflow condition occurs.
Zero Divide	A divide by zero error occurred in the execution of the function. The count is incremented on each iteration of the function whilst the divide by zero condition occurs.

Table 1 Error Statistics

2. Functions

2.1. Square Root Extraction



The Process Variable (PV) is scaled to a value 0…1 according to maximum input value (IP_{MAX}) and minimum input value (IP_{MIN}). Out of range values will result in the value being clamped to the range, i.e. a value greater than maximum input will use the IP_{MAX} value, similarly a value below the minimum will use the IP_{MIN} value. The square root of the resulting value is then taken and is returned as a percentage.

i.e.:

$$OUT = \sqrt{\frac{Min(Max(PV, IP_{MIN}), IP_{MAX}) - IP_{MIN}}{IP_{MAX} - IP_{MIN}}} \times 100$$

Note: All inputs (PV, IP_{MIN} and IP_{MAX}) and the returned value OUT must be real.

If $IP_{MAX} < IP_{MIN}$, the function assumes that the values have been reversed and ‘swap’ the values. The ‘Reversed’ error counter is incremented each time the function has to swap the parameters.

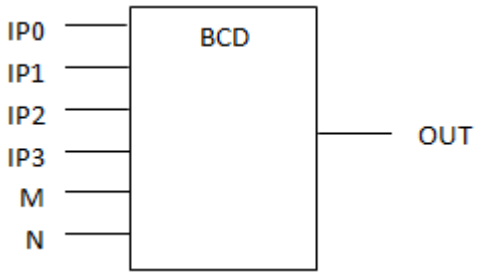
If $IP_{MAX} = IP_{MIN}$, the function returns a value of zero and increments the ‘zero divide’ error counter to indicate that a divide by zero error would have occurred.

If PV is not a number (NaN), the function returns a value of zero and increments the ‘Param’ error counter.

All input range parameters (IP_{MIN} and IP_{MAX}) must be finite, i.e. not ± infinities or NaNs. Behaviour is not defined for any other values.

If PV is not a NaN, and an overflow or underflow condition occurs the function returns zero and increment the corresponding error counter.

2.2. Binary-Coded Decimal (BCD) Translation



The BCD translation function requires the 4-bit BCD value represented as four Boolean parameters (IP0, IP1, IP2, and IP3), IP0 being the least significant bit, IP3 the most significant bit. All four inputs must be defined; inputs not required for specific application must be set to FALSE. M and N are of type integer.

M specifies the value multiplier and must be a power of 10, i.e. 1, 10, 100, etc. N allows the BCD functions to be cascade, by adding the value from a previous function. A value of -1 is assumed to be the result of an up-stream conversion error and will result in all chained BCD functions generating values of -1. The returned value OUT is an integer.

$$OUT = \left(BCD\left(IP0, IP1, IP2, IP3\right) \times M\right) + N$$

Invalid BCD numbers (see table below) to result in OUT = -1.

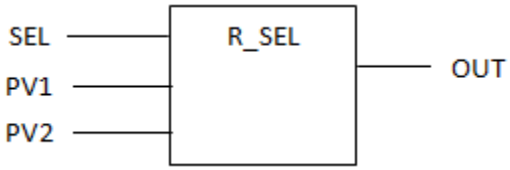
IP3	IP2	IP1	IP0	OUT (with M=1 and N=0)
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	-1

IP3	IP2	IP1	IP0	OUT (with M=1 and N=0)
1	0	1	1	-1
1	1	0	0	-1
1	1	0	1	-1
1	1	1	0	-1
1	1	1	1	-1

Table 2 BCD Translation

Multiple values (M) that are not powers of 10 are to result in a value of -1. If an overflow condition occurs, the function is to return a value of -1 and increment the 'Overflow', error counter.

2.3. Analogue Select



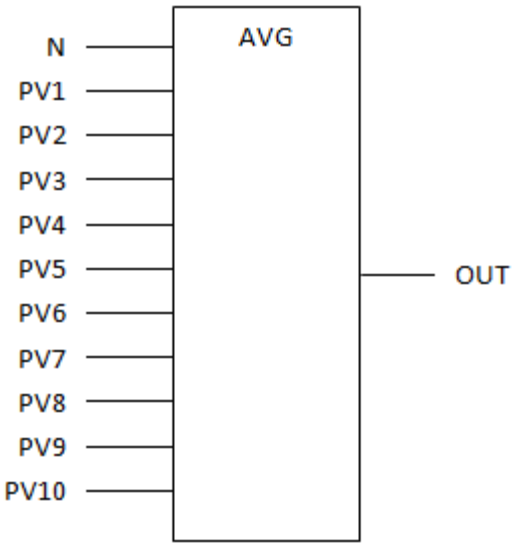
The process variables (PV1 and PV2) and the output (OUT) must all be of type REAL (see note below). The select signal (SEL) is of type BOOLEAN. When SEL is FALSE, PV1 is copied to the output; when SEL is TRUE, PV2 is copied to the output.

```
if SEL
        Set OUT = PV2
   else
        Set OUT = PV1
   end if
```

Out is finite if, and only if, the selected PV1 or PV2 is finite.

Note: The standard ISaGRAF SEL function provides the equivalent operation for integer values.

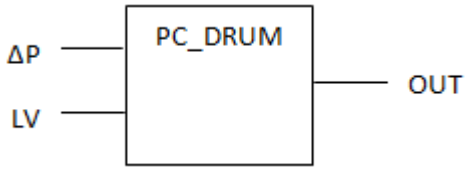
2.4. Multiple Value Averaging



The multiple value averaging calculates the average of between 1 and 10 values (PV1 to PV10). N specifies the number of values averaged; the first N values are averaged. E.g. if N=3, the average of PV1, PV2 and PV3 is returned. N is of type INTEGER, PV1 to PV10 and OUT are of type REAL. All input values must be connected; unused inputs should be tied to either one of the other used inputs or a constant value.

If the value of N is out of range, i.e. 1 > N > 10, the average of all ten input values is to be returned and the 'Param' error count incremented. Overflow errors occurring whilst calculating the average are to result in a value of PV1 being returned and the 'Overflow' error count incremented. If an 'Underflow' condition occurs, the function is to set OUT to zero and increment the 'Underflow' error counter.

2.5. Drum Level Control

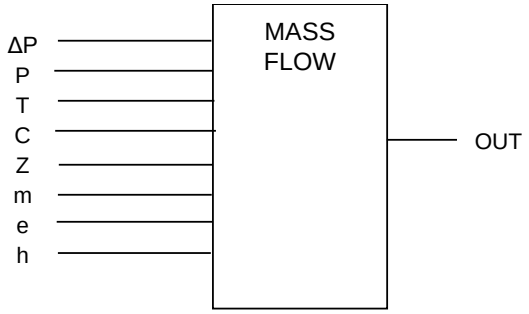


This function is to provide the pressure compensated drum level. ΔP is the differential pressure, and LV is the level in percentage of full-scale, the output (OUT) is return as a value 0 % to 100 %. ΔP, LV and OUT are all of type REAL. The function additionally to compensates for the reverse acting output of the level input (increasing ΔP value of decreasing differential pressure).

$$\mathrm{OUT} = \frac{101 - \left(5.43 \times 10^{-3} \times \Delta\mathrm{P}\right) - \mathrm{LV}}{0.88 - \left(0.173 \times 10^{-3} \times \Delta\mathrm{P}\right)}$$

Underflow, overflow and divide by zero error conditions are to result in value of zero being returned and the corresponding error counter incremented.

2.6. Mass Flow Computation



The mass-flow computation block provides the mass flow corrected for the current operating conditions.

$$OUT = \left(C \times Z \times m \times e \times \sqrt{\Delta P}\right) \times \left(h \times \frac{(P + 14.7)}{(T + 273)}\right)$$

ΔP is the differential pressure, P is pressure and T is temperature. C, Z, m, e and h are intended to be constant values. All parameters are of type REAL.

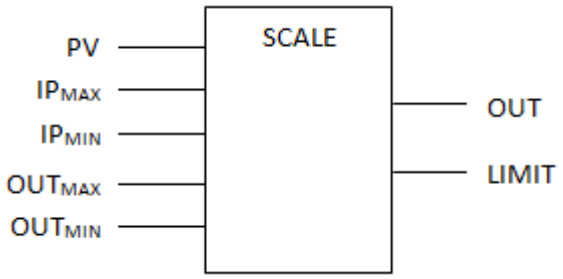
ΔP and P are in psi, T is in ˚C.

If T=-273, a divide by zero error condition is generated, this is to increment the associated error counter and the mass flow function is to return a value of -1. Over and underflow conditions are to result in an output value of -1 and increment the associated error counter.

Page intentionally left blank

3. Function Blocks

3.1. Analogue Scaling



The scale function linearly scales the input value (PV) according to the low input (IP_{MIN}), high input (IP_{MAX}), low output (OUT_{MIN}) and high output (OUT_{MAX}) values. The input value is clamped to the range IP_{MIN} to IP_{MAX}. I.e. a PV value less than IP_{MIN} results in the IP_{MIN} value being used, similarly a value greater than IP_{MAX} results in the IP_{MAX} value being used). The output (OUT) value varies linearly between OUT_{MIN} and OUT_{MAX} as the input (PV) varies between IP_{min} and IP_{MAX}. The LIMIT output is to indicate that the input value (PV) is out of range, i.e. LIMIT = IP_{MIN} > PV > IP_{MAX}, LIMIT is set TRUE if PV is out of range.

IP_{MAX}, IP_{MIN}, OUT_{MAX} and OUT_{MIN} are typically constants but may, in practise, use any analogue or register value.

$$OUT = \left(\left(\frac{\mathrm{Min}\left(\mathrm{Max}\left(PV, IP_{MIN}\right), IP_{MAX}\right) - IP_{MIN}}{IP_{MAX} - IP_{MIN}} \right) \times \left(OUT_{MAX} - OUT_{MIN} \right) \right) + OUT_{MIN}$$

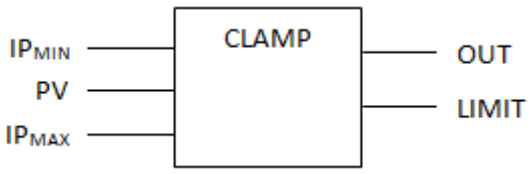
Note: PV, IP_{MAX}, IP_{MIN}, OUT_{MAX} and OUT_{MIN} have to be of type REAL, the result (OUT) is returned as type REAL and LIMIT is of type BOOLEAN.

If IP_{MAX} < IP_{MIN}, the function is to assume that the values have been reversed and 'swap' the values. Similarly, if OUT_{MAX} < OUT_{MIN} the two values are to be swapped. In both cases, the 'Reversed' error counter is to be incremented each time the function has to swap the parameters. All range parameters (IP_{MAX}, IP_{MIN}, OUT_{MAX}, OUT_{MIN}) must be finite, i.e. not ± infinities or NaNs. Behaviour is not defined for any other values.

If IP_{MAX} = IP_{MIN}, the function is to set OUT to OUT_{MIN}, LIMIT to FALSE and increment the 'Zero Divide' error counter to indicate that a divide by zero error would have occurred.

If PV is a NaN, the function is to set OUT to OUT_{MIN}, LIMIT to FALSE and increment the 'Param' error counter. If PV is not a NaN and overflow or underflow conditions occur during the execution of the function, the function is to set OUT to OUT_{MAX} or OUT_{MIN} respectively, LIMIT to FALSE and increment the corresponding error counter. If both overflow and underflow conditions occur, the function is to set OUT to OUT_{MIN}, LIMIT to FALSE and increment both corresponding error counters.

3.2. Analogue Value Clamping



The process variable (PV), IP_{MAX}, IP_{MIN} and OUT values are all of type REAL (see note below) The output value (OUT) is clamped to the range specified by IP_{MIN} and IP_{MAX}. The LIMIT output, which is of the type Boolean, is set to TRUE if the clamp is in operation, i.e. LIMIT = $IP_{MIN} > PV > IP_{MAX}$.

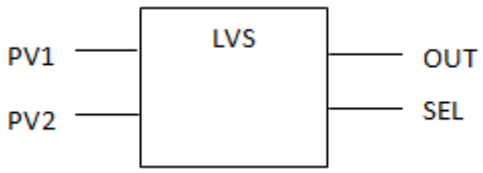
$$OUT = \mathrm{Min}\left(\mathrm{Max}\left(PV, IP_{MIN}\right), IP_{MAX}\right)$$

If $IP_{MAX} < IP_{MIN}$, the parameters are to be 'swapped' and the reversed error count incremented.

Input range parameters (IP_{MIN} and IP_{MAX}) must be finite, i.e. not ± infinities or NaNs. Behaviour is undefined for any other values. If PV is a NaN, the function is to set OUT to OUT_{MIN} and increment the 'Param' error counter.

Note: The standard ISaGRAF LIMIT function provides the equivalent of the CLAMP operation for integer values (without the limit output).

3.3. Low Value Select



The process variables (PV1 and PV2) and the output (OUT) are all REAL values (see note below) The output (OUT) is set to the lower of the two process variable values.

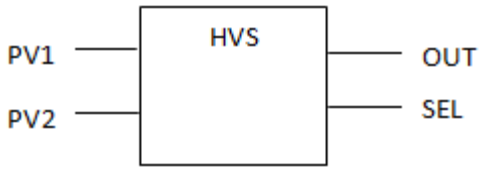
$$OUT = \mathrm{MIN}\left(PV1, PV2\right)$$

The SEL output indicates which of the process variables is being used to generate OUT. SEL is of type BOOLEAN, and is FALSE if using PV1 and TRUE if using PV2.

If PV1 or PV2 is a NaN, the function is to set OUT to PV1, SEL to FALSE and increment the 'Param' error counter. OUT is not a NaN if, and only if, PV1 is not a NaN and finite if, and only if, the lower of PV1 and PV2 is finite.

Note: The standard ISaGRAF MIN function provides the equivalent functionality for integer values (without the SEL output).

3.4. High Value Select



The process variables (PV1 and PV2) and the output (OUT) are all REAL (see note below) The output (OUT) is set to the higher process variable value.

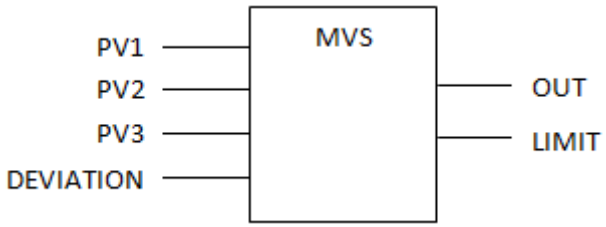
$$OUT = MAX(PV1, PV2)$$

The selected output (SEL) indicates which of the process variables is being used to generate OUT. SEL is of type Boolean, and is FALSE if using PV1 and TRUE if using PV2.

If PV1 or PV2 is a NaN, the function is to set OUT to PV1, SEL to FALSE and increment the 'Param' error counter. OUT is not a NaN if, and only if, PV1 is not a NaN and finite if, and only if, the higher of PV1 and PV2 is finite.

Note: The standard ISaGRAF MAX function provides the equivalent functionality for integer values (without the SEL output).

3.5. Median Value Select



The median value select sets OUT to the mean of the three process variables (PV1, PV2 and PV3) if the deviation between their values is less than the maximum (DEVIATION). If the variation in inputs is less than the allowed deviation the LIMIT output is set to FALSE. If the input values vary by more than DEVIATION, OUT is set to the median value and LIMIT is set to TRUE.

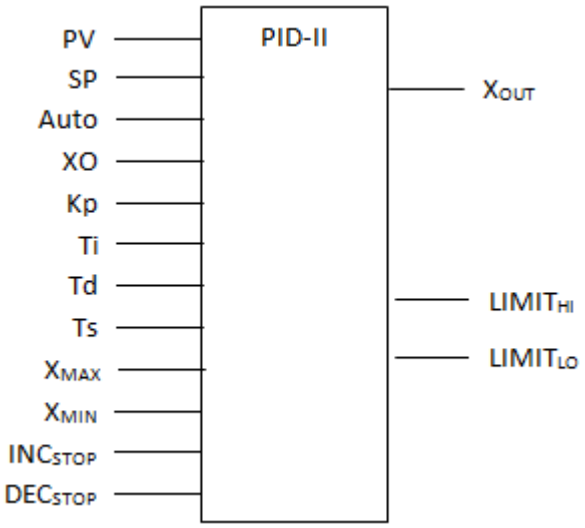
If an overflow condition occurs during the execution of the function, e.g. if the deviation is huge, the function is to set OUT to the median value, LIMIT to TRUE, and increment the 'Overflow' error counter.

All input parameters (PV1, PV2, PV3 and DEVIATION) must be finite, i.e. not ± infinities or NaNs.

3.6. Proportional PID Function

Note: The PID_II function block has been superseded by the IPID function block, which provides incremental PID and enhanced functions. All new strategies should use IPID. PID_II is retained for compatibility, but may be replaced in an application by IPID if desired.

A PID is a process regulator. Using the feedback concept, an output is regulated according to the difference between its current value (PV) and its required value (SP).



Auto is the auto/manual mode select (type BOOLEAN). When set to TRUE the PID calculation is in auto mode. XO is the manual set-point and is type REAL. Kp is the proportional constant (type REAL). Ti is the integral time constant (type TIMER). If this is set to zero the function provides only the P and D terms. Td is the derivative time constant (type TIMER). Ts is the sample time (type TIMER). If the specified sample time is less than the application program scan time, then the effective Ts will be the greater of the application scan time or PID function execution interval.

Figure 1 below gives a simplified overview of the PID function:

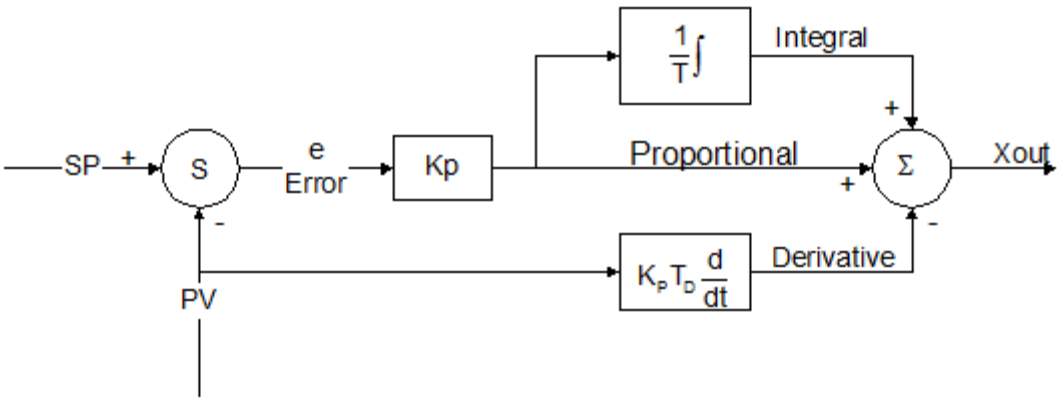


Figure 1 PID Function

At entry the error ε is calculated as the difference between the set-point (SP) and the process variable (PV). This error value is used in the calculation of the proportional and integral terms. The derivative term is derived from PV as shown.

INC_{STOP} prevents the output X_{OUT} increasing when TRUE. When this input is TRUE and the PID is in AUTO mode, the output is compared with the previous output and only values less than or equal to the previous output will be returned and $LIMIT_{HI}$ will be set to TRUE. The integral term is also adjusted to provide windup protection. Similarly, when DEC_{STOP} is TRUE Xout will only return values greater than or equal to the previous value, and $LIMIT_{LO}$ will be set to TRUE.

In manual mode (AUTO = FALSE), the output (Xout) follows the input XO. The function provides manual to automatic bumpless transfer by adjusting the integral term. If set-point tracking is required then the MANTRK function should be used in conjunction with the PID_II function.

The PID function provides reset windup protection by allowing high and low limits (Xmax and Xmin) to be specified and anti-reset windup on the integral term. If the PID output would have exceeded the specified high or low limits, the corresponding limit output ($LIMIT_{HI}$ and $LIMIT_{LO}$) is generated. Note that the limit outputs are updated on each iteration of the PID function. The maximum and minimum limits on Xout apply in both auto and manual modes.

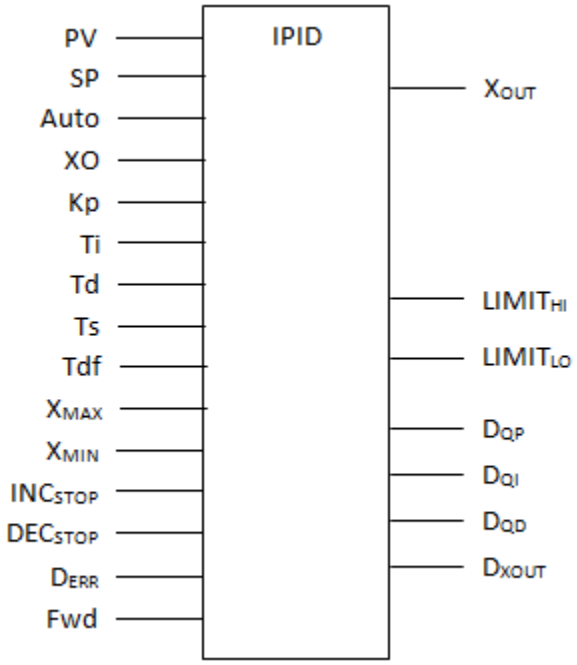
In auto mode, the PID function will return only the proportional element on the first application scan, and will the integral and derivative terms.

If Xmax < Xmin, the function assumes that the values have been reversed and ‘swaps’ the values. On each execution of the PID_II function with reversed parameters the ‘reversed’ error count will be incremented. If an overflow condition occurs during the execution of the PID function the output Xout will be set to Xmin in auto mode or XO in manual mode, and the integral term will be adjusted to provide windup protection.

3.7. Incremental PID Function

A PID is a process regulator. Using the feedback concept, an output is regulated according to the difference between its current value (PV) and its required value (SP).

The IPID function block is an improved version of the PID_II block with incremental calculation, derivative term switch for PV or error, and a derivative term filter. The IPID block is recommended for all new applications. It is also a relatively simple retrofit for the PID_II to provide a more stable output.



Auto is the auto/manual mode select (type BOOLEAN). When set to TRUE the IPID calculation is in auto mode. XO is the manual set-point and is type REAL. Kp is the proportional constant (type REAL). Ti is the integral time constant (type TIMER). If this is set to zero the function provides only the P and D terms. Td is the derivative time constant (type TIMER). Kp, Ti and Td may be dynamically changed in Auto mode without process bump.

Fwd switches between Forward and Reverse action (type BOOLEAN). When Fwd is FALSE and Kp is positive, an increasing PV causes a decreasing Xout (reverse action). When Fwd is TRUE and Kp is positive, an increasing PV causes an increasing Xout (forward action). If Kp is negative, the action is also inverted.

Ts is the sample time (type TIMER). If the specified sample time is less than the application program scan time, then the effective Ts will be the greater of the application scan time or PID function execution interval. Changing Ts when in Auto mode may cause a process bump due to recalculation of dynamic variables.

Tdf is the derivative action filter time constant (type TIMER). This may be used to reduce reaction to process noise, or to spread derivative action over time. Tdf sets the exponential time constant of a first order lag. If set to zero, the derivative action will act on the process fluctuations immediately.

D_{ERR} switches the derivative action input (type BOOLEAN). When set to TRUE, the derivative action is calculated based on the change in error (SP - PV). When set to false, the derivative action is calculated based on the change in PV. This allows a choice on whether setpoint changes demand a derivative kick or not. Note that when D_{ERR} is false, the proportional action will still react to setpoint changes, but with less step than P + D action.

The IPID calculation is incremental; at each scan, the required change in output is calculated. This means that the P,I and D settings may be dynamically tuned without significant change to the output. The total change at each scan is output as D_{XOUT} (type REAL). The change due to the P,I and D terms are output as D_{QP}, D_{QI} and D_{QD} (all type REAL). These may be used for continuous incremental action or for commissioning data.

The figure below gives an overview of the IPID function:

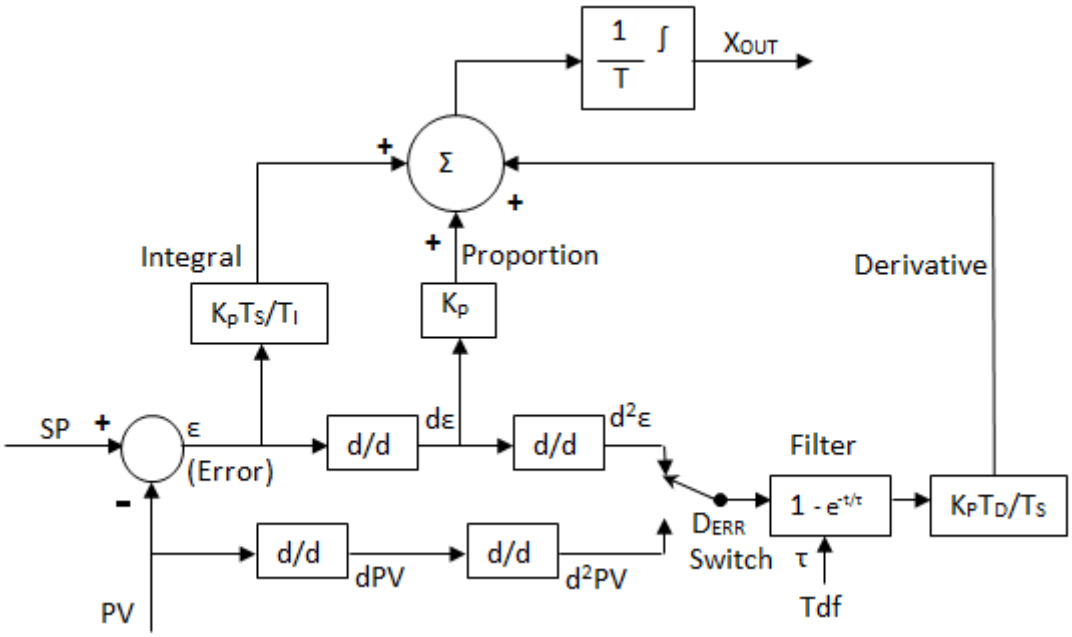


Figure 2 IPID Function

The calculation is performed one step into the derivative and is then integrated back for the real output.

At entry the error ε is calculated as the difference between the set-point (SP) and the process variable (PV). This error value is used directly for the integral action, after one differentiation for the proportional action, and after two differentiations for the derivative action. The derivative action can be switched using D_{ERR} to either use the second differential of error or PV, with a filter.

INC_{STOP} prevents the output X_{OUT} increasing when TRUE. When this input is TRUE and the IPID is in AUTO mode, the output is compared with the previous output and only values less than or equal to the previous output will be returned and $LIMIT_{HI}$ will be set to TRUE. The delta outputs D_{QP}, D_{QI}, D_{QD} and D_{XOUT} continue to update. Similarly, when DEC_{STOP} is TRUE Xout will only return values greater than or equal to the previous value, and $LIMIT_{LO}$ will be set to TRUE.

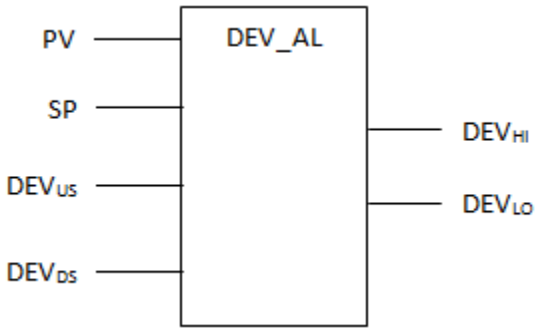
In manual mode (AUTO = False), the output (Xout) follows the input XO. Bumpless transfer to auto is inherent because only the change in output is calculated. If set-point tracking is required then the MANTRK function should be used in conjunction with the IPID function.

The IPID function provides high and low limits (Xmax and Xmin) to be specified. If the PID output would have exceeded the specified high or low limits, the corresponding limit output ($LIMIT_{HI}$ and $LIMIT_{LO}$) is generated. Note that the limit outputs are updated on each iteration of the IPID function. The maximum and minimum limits on Xout apply in both auto and manual modes. There is no need for anti-windup protection because there is no stored integral term.

On the first application scan, the PID function will clear its registers and will not calculate an output.

If Xmax < Xmin, the function assumes that the values have been reversed and 'swaps' the values. On each execution of the IPID function with reversed parameters the 'reversed' error count will be incremented. If an overflow condition occurs during the execution of the PID function the output Xout will be set to Xmin in auto mode or XO in manual mode, and the stored registers are cleared.

3.8. Deviation Alarm



The deviation alarm function block takes the current process variable (PV) and set-point (SP), together with the up-scale and down-scale deviation between the PV and SP (DEV_{US} and DEV_{DS} respectively). PV, SP, DEV_{US} and DEV_{DS} are of type REAL, both outputs (DEV_{HI} and DEV_{LO}) are of type Boolean.

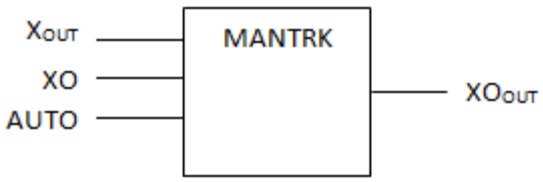
$$DEV_{HI} = ((PV - SP) \geq DEV_{US})$$

$$DEV_{LO} = ((SP - PV) \geq DEV_{DS})$$

If an overflow condition occurs during the execution of the function, DEV_{HI} is to be set to PV > 0, DEV_{LO} to PV < 0 and the 'Overflow' error counter is to be incremented.

PV, SP, DEV_{US} and DEV_{DS} must be finite, i.e. not ± infinities or NaNs.

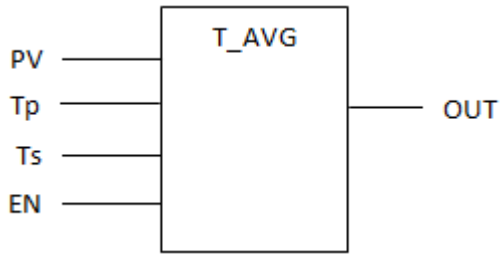
3.9. Manual Tracking



The manual tracking function requires the current PID function output (X_{OUT}), the manual set-point (XO) and auto/manual mode control (AUTO). Xout, XO and XO_{OUT} are of type REAL, AUTO is BOOLEAN.

Whilst AUTO is TRUE XO_{OUT} will be set to the current value of X_{OUT}. On a transition to manual mode, i.e. AUTO changes to FALSE, the XO_{OUT} value will remain at its previous value until the value of the manual set-point input (XO) changes. Following a change to the manual set-point, the value of XO_{OUT} will reflect the value of XO whilst AUTO is FALSE.

3.10. Time Averaged Value



The average function calculates the average process variable (PV) value over the specified period (Tp). The Ts specifies the minimum interval between samples. If the sample period is less than the elapsed period between application program scans, Ts will be the application scan time. PV and OUT are of type REAL; Tp and Ts are of type TIMER.

The enable input (EN) enables the use of the time averaged value; if this input is TRUE, OUT returns the time averaged value, if this input is FALSE, out is set to the current process variable (PV) value. The calculation of the time-averaged value continues irrespective of the EN state.

The function assumes that the input value (PV) changes linearly between samples, i.e. between application program scans. The maximum number of samples taken must be less than 64, i.e. $\frac{Tp}{Ts} \leq 64$. If the number of samples required to support $\frac{Tp}{Ts}$ is greater than 64, then 64 samples will be used and the time averaged value over these 64 samples returned (see Note 2 below). The function is to include time weighting of the sampled values to correct for variation in application scan times. If the sample period (Tp) is less than the sample interval (Ts) the function will effectively average over a single sample, and return OUT = PV.

The output value (OUT) will be updated only after a sample has been taken, i.e. after Ts.

If an underflow condition occurs during the execution of the function, the function is to set OUT to zero and increment the 'Underflow' error counter.

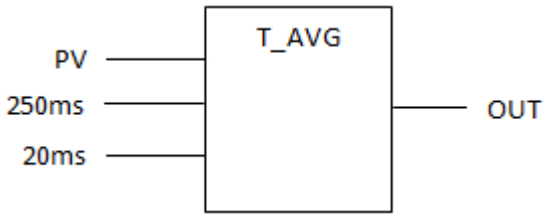
Overflow conditions are to result in OUT = PV, and increment the 'Overflow' error counter.

On initialisation, i.e. first application program scan, the average value is set to the current input value, and is assumed to have been stable at this value for greater than Tp.

Note 1: Tp and Ts are variable at run-time. Although the function should not normally be used in this manner, it is important that the behaviour with varying values is predictable. Changes in Tp or Ts are to result in OUT being held at its previous value until sufficient samples have been taken to calculate the valid time weighted average, i.e. after Tp.

Note 2: ISaGRAF does not support the verification of the values used for Tp and Ts within the workbench. Also, note that these values are variables and may be changed at runtime, although this is not recommended. Changing these values at runtime will result in the same behaviour as on initialisation, i.e. the input will be assumed to have been stable for ≥ Tp and the output set to PV.

Example:

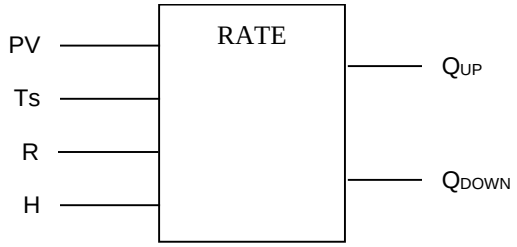


T (ms)	350	390	432	475	521	565	607	650	690	732	777	821
PV	23.5	15.46	10.01	56.03	58.61	90.17	85.68	84.56	18.09	95.94	75.66	49.38
Ts (ms)	46	40	42	43	46	44	42	43	40	42	45	44
% of Tp	103.2	102.4	101.6	102.8	102.8	104.4	102.8	104.0	103.2	102.8	102.4	102.4
OUT	23.50	22.87	21.09	22.71	28.76	37.26	48.00	58.53	64.87	68.91	73.96	71.92

Table 3 Example Time Average

The example assumes that PV has been static at 23.5 for greater than 250 ms before the first sample shown. The Ts row shows the actual interval between samples. The % of Tp row shows the figure used for the overall output adjustment; e.g., the actual period between the 521 ms and 777 ms iterations is 256 ms, i.e. 102.4 % of Tp.

3.11. Rate of Change Detection



The Rate of Change Detection function requires four parameters: the process variable to be monitored (PV), the sample interval (Ts), the maximum rate of change parameter (R), and rate of change hysteresis (H). The function detects when the rate of increase or decrease of the process variable (PV) has exceeded the defined limit R. When the rate of change has exceeded R, the output (Q) is set to TRUE. Once the output is TRUE, it is set to FALSE as soon as the rate of increase or decrease has fallen below R by at least the hysteresis value (H).

The process variable is sampled at the specified sample interval; the sample period used will be the greater of Ts and the application scan time (see note below) The change in PV and the actual period between samples is determined and used to establish the rate of change.

Note: The actual sample interval will be NOT LESS THAN the specified sample interval. The maximum period between samples is T plus the application scan time

On initialisation, Q is set to FALSE and the previous PV value to the current PV value

PV, R and H must be of type REAL. Ts must be of type TIMER, the output (Q) is of type Boolean. The rates of change parameters (R and H) are specified as rate of change per second.

i.e.

$$Q_{UP} = \left(\left[\frac{\left(PV_{t2} - PV_{t1}\right)}{T_{t2} - T_{t1}} \right] > R \right) \left\| \left(\left[\frac{\left(PV_{t2} - PV_{t1}\right)}{T_{t2} - T_{t1}} \right] > \left(R - H\right) \right) . Q_{UP_{t1}} \right.$$

where:

PV_{t1} is the input value on the previous sample, on the first sample this value is set to the current value of the input, i.e. PV_{t2}.

PV_{t2}.is the current input value

T_{t1}.is the time the previous sample was taken.

T_{t2} is the current time.

R is the rate of change limit value

H is the rate of change hysteresis value

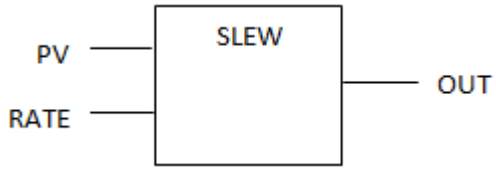
Q_1 is the previous value of Q

Parameters PV, R and H must be finite, i.e. not ± infinities or NaNs. Behaviour is not defined for any other values. If negative values are specified for R or H, their absolute (ABS) value will be used and the 'param' error counter will be incremented.

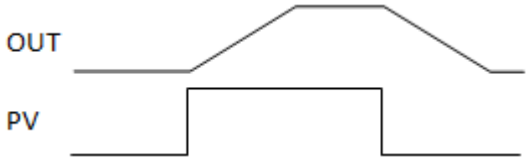
The function must record the sample time for the previous input value and calculate the rate of input value change for the actual elapsed period.

In an overflow condition occurs, the function is to set Q_{UP} to PV > 0, Q_{DOWN} to PV < 0 and increment the 'Overflow' error counter. In an underflow condition occurs, the function is to set Q_{UP} and Q_{DOWN} to FALSE and increment the 'Underflow' error counter.

3.12. Analogue Value Slew



The analogue value slew function limits the rate of change of the output value (OUT) to the maximum specified by the RATE input. On initialisation, OUT is set to PV. Changes in the input (PV) will be tracked by the output with the maximum rate of change. If the rate of change of PV is less than the maximum rate, the output will immediately follow the input. PV is assumed to be in units, RATE is then in units per second.



i.e.:

if PV > OUT_{t1} then

$$OUT = OUT_{t1} + \left((T_2 - T_{t1}) \times \text{Min}\left(\left(\frac{\text{ABS}(PV - OUT_{t1})}{T_2 - T_{t1}} \right), (RATE) \right) \right) \text{else}$$

$$OUT = OUT_{t1} - \left((T_2 - T_{t1}) \times \text{Min}\left(\left(\frac{\text{ABS}(PV - OUT_{t1})}{T_2 - T_{t1}} \right), (RATE) \right) \right) \text{end if}$$

Where:

T_2 is the current time (seconds)

T_{t1} is the time the algorithm was previously executed (seconds)

PV is the current input value

OUT_{t1} is the output value the time the algorithm was previously executed

RATE is the current RATE input value

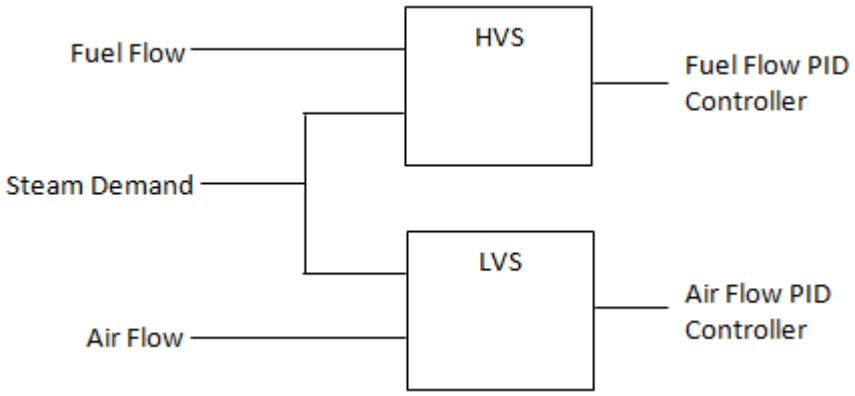
All input parameters (PV and RATE) must be finite, i.e. not ± infinities or NaNs. Behaviour is not defined for any other values. If RATE is negative, its ABS value will be used and the 'Param' error counter incremented.

If an overflow condition occurs during the execution of the function, the function is to set OUT to its previous value and increment the 'Overflow' error counter. If underflow occurs, the function is to set OUT to its previous value and increment the 'Underflow' error counter.

Page intentionally left blank

4. Lead/Lag Control Function

The lead/lag facility will be provided using the high value select and low value select functions to select the set-point values to be used for the fuel and airflow PID functions. The Lead/Lag is not a function or function block in its own right. This section illustrates the use of the previously defined high value select and low value select functions to provide the required lead/lag control function.



Page intentionally left blank

5. Sample Application Program

A sample application program using selected Process Control Algorithms to implement a simple PID loop is shown in Figure 3.

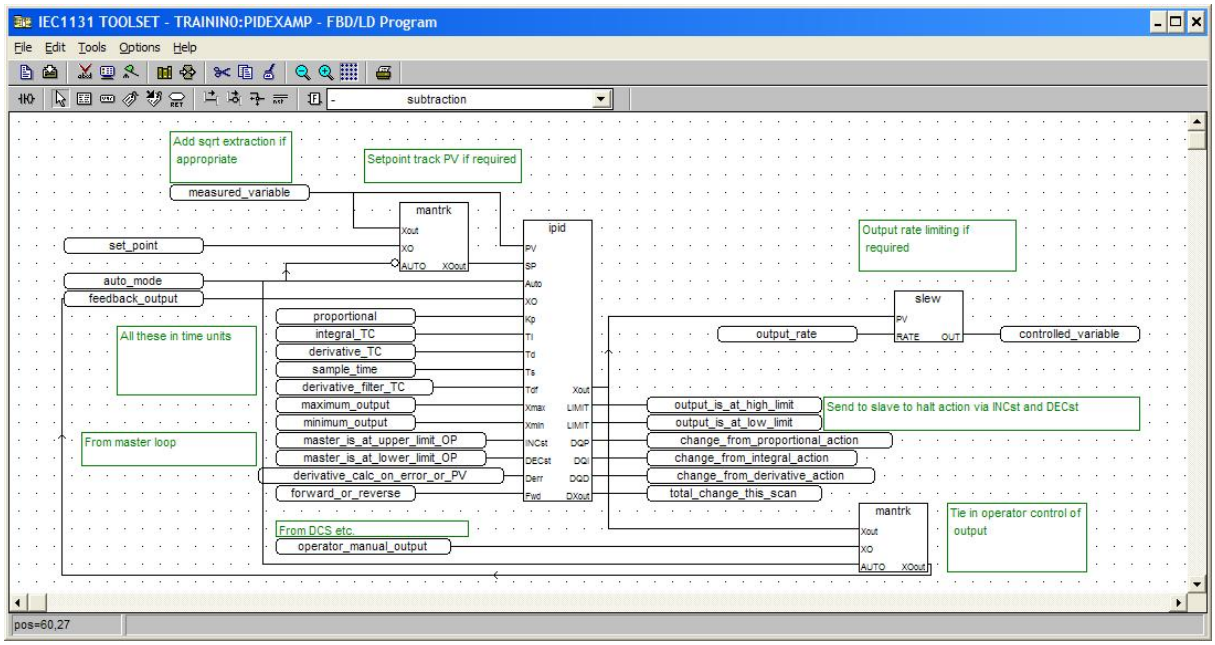


Figure 3 Sample Application Program

Note: Many of the input parameters to the IPID function block will usually be program constants, and the extra outputs are not necessary for a single loop.

The mantrk blocks allow read/write control of setpoint and output. For the setpoint, the mantrk block copies the measured variable to the setpoint input whilst the loop is in manual, to allow a bumpless transfer to automatic mode. On entering automatic mode the setpoint will remain at this value until the setpoint is changed. Note that since the IPID function block uses incremental action, this function is only necessary if the output should not move on entering automatic mode until the setpoint is entered. Without this function, the output will move to control to the setpoint but without a bump to the output.

The second mantrk block allows manual control of the output in manual mode, using the 'operator manual output' tag. In automatic mode, the mantrk block follows the automatic output and writes to the IPID's XO input. On entering manual mode, the IPID block output follows XO, but the operator can now change the output using 'operator manual output'.

The slew function block limits the rate of change of the output, and is optional. Note that although this will smooth the action to the control valve, it will also change the control response in a non-linear fashion.

For cascade (master-slave) pairs of IPID blocks, wire the LIMIT outputs of the slave back to the INCst/DECst inputs of the master. This will stop the master's action if the slave output

has hit its limit. Wire the output of the master to the setpoint of the slave. If the slave is not in auto and reading its setpoint from the master's output ('remote' mode) then the master should be forced into manual mode because its output is not being used. For any loop, if the measured variable is suspect, force the loop into manual mode. If the remote setpoint is suspect, force the loop into local automatic mode with a manually adjustable setpoint. Always wire intermediate signals like remote setpoints and master-slave interactions through variables, so that their state can be monitored.