WOODWARD

Application Note 51566
(Revision NEW, 1/2017)
Original Instructions

Basic Operation of Woodward GAP PID Blocks, Includes PID, PID 2, PID OPTI, PID DB, and PID SAMPLE

⚠WARNING

Overspeed / Overtemperature / Overpressure

The engine, turbine, or other type of prime mover should be equipped with an overspeed shutdown device to protect against runaway or damage to the prime mover with possible personal injury, loss of life, or property damage.

The overspeed shutdown device must be totally independent of the prime mover control system. An overtemperature or overpressure shutdown device may also be needed for safety, as appropriate.

Revisions

This publication may have been revised or updated since this copy was produced. To verify that you have the latest revision, check manual *26455*, *Customer Publication Cross Reference and Revision Status & Distribution Restrictions*, on the *publications page* of the Woodward website:

www.woodward.com/publications

The latest version of most publications is available on the *publications page*. If your publication is not there, please contact your customer service representative to get the latest copy.

Revisions—Changes in this publication since the last revision are indicated by a black line alongside the text.

Application Note 51566

Contents

Illustrations and Tables

Chapter 1.
General Information

Executive Summary

There are many different PID architectures in use in industry today. This can be confusing for people trying to use and to tune these PIDs. At a high level PIDs can be implemented in a serial manner, in a parallel manner, with PROP_GN distributed (with INT_GN and DERIV multiplied by PROP_GN),with PROP_GN, INT_GN, and DERIV separated, and in various combinations of these. In some PIDs the error between the process and the setpoint is the input to the PID functionality, and in other PIDs the functionality for the process input is different from that for the setpoint or reference input.

In addition, different PIDs have different scaling for PROP_GN, INT_GN, and DERIV, and different scaling for the PID outputs. They have different ways of handling integrator wind-up. They have different limits, on the inputs, and on PROP_GN, INT_GN, and DERIV. A PID user must be aware of the PID form and scaling, plus the scaling of the PID inputs and outputs as they relate to the rest of the system. This can be a lot of information to investigate, utilize, and document.

This Application Note helps users understand some of the Woodward GAP PIDs, without getting into detailed control theory or equations. It introduces PID operation, and details about the operation of some Woodward PIDs, including the PID, PID_2, PID_OPTI, PID_SAMPLE, and PID_DB.

The Woodward GAP PID blocks include the following:

- PID - Developed for turbine control
- PID_2 - Same as the PID, except PROP_GN, INT_GN, and S_D_R can come from other GAP blocks, such as gain tables or curves.
- PID_DB - Similar to the PID and PID2 except the output will not change unless the error is greater than the DB input. This block is used when the output needs to not continuously make corrections, for example, when small output movements cause excessive valve wear.
- PID_SAMPLE - Similar to the PID and PID2 with a “SAMPLE” input. The block executes only when the “SAMPLE” input is TRUE.
- PID_OPTI - This is the new “self-tuning” PID, the PID functionality is very like the PID_2 block.
- PID_ENGA, PID_ENGALG - These PIDs are used primarily for slow speed engines. The PID equations are different from those discussed in this document.
- PID_HYD - This block was created for hydro control and has a separate P, I and D input. The PID equations are different from those discussed in this document.

  Note: the information in this document does not apply to the PID_ENGA, PID_ENGALG, and PID_HYD GAP blocks.

This document is supplemental to the GAP help and associated documents. For the actual PID equations in several different forms, refer to AppNote 51433, and to the GAP help. For information on Woodward provided classes, consulting, and simulation services, please contact Woodward Inc. Contact information is on the last page of this document.

Introduction

This Application Note serves as an overview of the basic operation of some of the Woodward PID GAP blocks, without getting into detailed control theory. This information applies to the PID, PID_2, PID_OPTI, PID_SAMPLE, and PID_DB GAP blocks. All GAP block names, GAP block inputs and GAP block outputs in this AppNote are in UPPERCASE.

At a high level, a PID is a continuous controller, which can be used to control things such as speed, pressure, or temperature. A PID controls a process by moving an actuator, to minimize the difference between the desired setpoint for the process and the actual process.

In general, a PID has a setpoint (SP) input, a process (PROC) input, and an output.

- The SP input is the desired process setpoint, sometimes called the reference.
- The PROC input is a measurement of the actual process.
- The output typically drives an actuator, sometimes called the “demand output” or “command output”.

In addition, PIDs have inputs to adjust the amount and speed of “response”. These are:

- PROP_GN - Proportional gain
- INT_GN - Integral gain, sometimes called “Reset”, though in effect INT_GN is usually proportional to the inverse of Reset or 1/Reset.
- DERIV - Derivative gain, sometimes called “Actuator Compensation”. The Woodward’s PIDs discussed here use an input called S_D_R (Speed Derivative Ratio), rather than Derivative. S_D_R is a ratio of INT_GN to DERIV, and can be translated to Derivative, as described in this document.

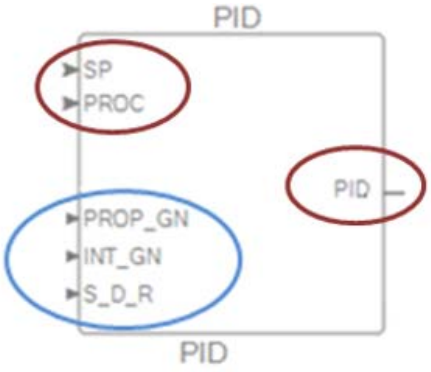


Figure 1. Diagram of a Basic PID

In more detail, a PID is a controller with 3 terms, (P, I, and D) whose effects are added together to make the output. For the purposes of this AppNote, we will call the effects of these three terms, the Proportional Calculations, the Integral Calculations, and the Derivative Calculations.

For these PIDs the output depends on the difference between the setpoint and the process and on the PROP_GN, it does not depend on SP or PROC alone. We call this difference, the “error”.

As shown in Figure 2, PROP_GN effects the error and therefore the Integral and Derivative Calculations. This simplifies tuning of the PID. Looking at Figure 2, the Integral Calculations (in green) are a function of the error (in light brown), PROP_GN (in red), and INT_GN (in green). Similarly, the Derivative Calculations (in blue) are a function of the error, PROP_GN (in red), and DERIV (in blue). The output of the PID is the sum of the Proportional Calculations, the Integral Calculations, and the Derivative Calculations.

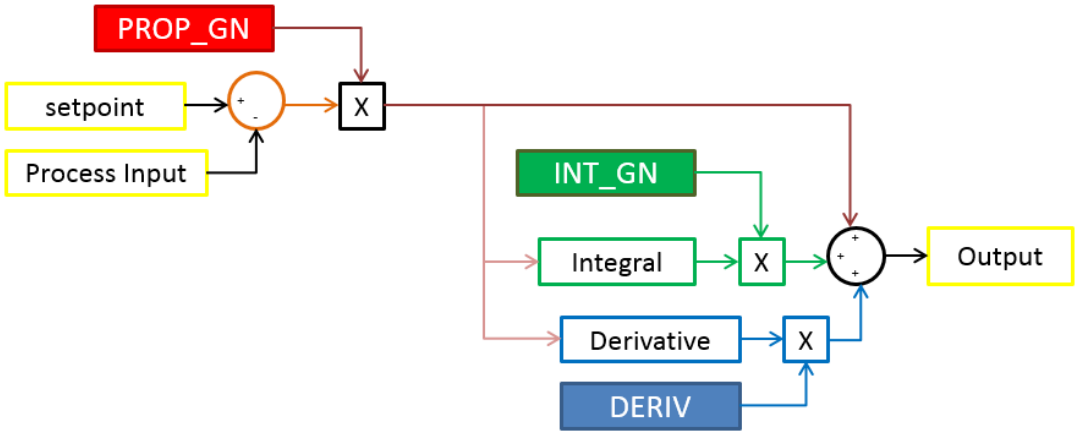


Figure 2. Detailed Diagram of a Parallel PID

Note that in the PID_OPTI GAP block the PROC input equals the PROC input added to the DROOP input.

In general, the Proportional Calculations depend on the instantaneous error, the Integral Calculations depend on the past errors, and the Derivative Calculations depend on the change in the error.

One of the issues with understanding P, I, and D can be that the numbers *seem* to be unit-less. We know what 10 lbs., 25 meters, 60Hz, and 3600 RPM look like because there is a physical dimension attached. The dimensions of P, I, and D (or S_D_R) are less obvious. The PROP_GN, the INT_GN, and S_D_R/DERIV, will be explained in more detail in the following sections.

PROP_GN

PROP_GN is the proportional gain. The Proportional Calculations depend only on the PROP_GN and the instantaneous error, the difference between the setpoint and process input. Remember, errors can be positive and negative. For example, if the SP is 3600 and the PROC is 3595 then the error is 5 RPM. If PROP_GN is 2, the Proportional Calculations will contribute 10 to the PID output. The PID output will be 10 + 10*Integral Calculations + 10*Derivative Calculations.

The units of PROP_GN depend on the units of the PID inputs and outputs. As an example, the SP (setpoint) and PROC (Process Input) are shown in yellow, on the left side of Figures 1 and 2. The Output is shown in yellow on the right side of Figures 1 and 2. The units of each of these are determined by the GAP application. If the SP and PROC input are in RPM, and the Output is in percentage of Actuator travel, then the units of PROP_GN must be:

$$\frac{\%\ \text{of Actuator Travel}}{\text{RPM}}$$

INT_GN

Integral is another term for cumulative, the Integral Calculations are the cumulative error. If you add any error over time, you get an “accumulation” of the previous errors. The Integral Calculations let the PID “remember” or “accumulate” the previous errors. With a small constant error, the Integral Calculations will increase.

The Integral Calculations will increase faster with higher INT_GNs, more slowly with lower INT_GNs.

The effect of INT_GN on the system is to drive the error to zero, so that PROC equals SP, on average.

The units of INT_GN are in Hz, a larger number results in a faster increase or decrease in the Integral Calculations. Some earlier Woodward products (2301, 723) called the integral term “Reset”. In this case, the units were typically seconds. It is important for the user to understand the units of this input before adjusting it.

Figure 3 demonstrates the effect of the PROP_GN, INT_GN, and DERIV.

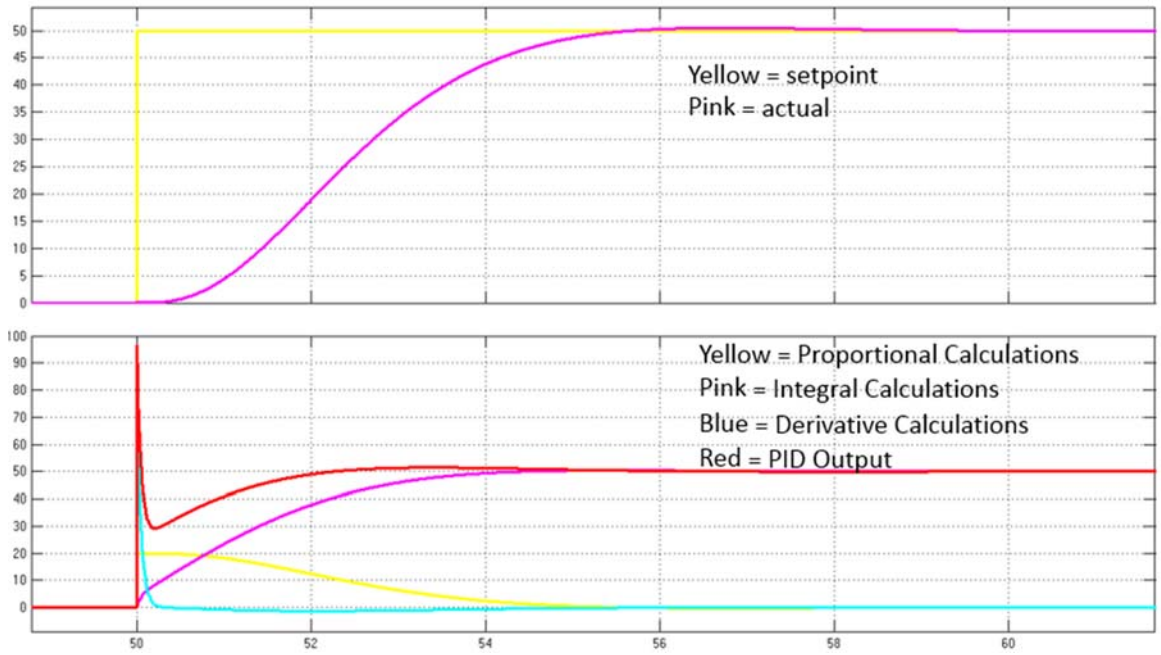


Figure 3. Components of PID Response

In Figure 3, the top plot shows the setpoint (in yellow), and the actual process (in pink), during a step change of the setpoint. The bottom plot shows the effect of the Proportional Calculations (in red), the Integral Calculations (in pink), the Derivative Calculations (in blue), and the PID output (in red). The PID output is the sum of the three Calculations, as shown in Figure 2.

In most of the Woodward GAP PIDs, PROP_GN is used by the Proportional Calculations, and is part of the input used by the Integral Calculations and the Derivative Calculations, again shown in Figure 2. Notice that when the setpoint steps up, the Proportional Calculations step up also, because the error stepped up. The Integral Calculations increase slowly, in proportion to the amount of error. The Derivative Calculations step up quickly, and then decrease quickly as a result of the change in the error.

The Integral Calculations are responsible for the number that the PID outputs, at steady state. Whether the output that is needed is 10% or 50%, the Integral Calculations will increase or decrease the PID output until it reaches the needed value. The Integral Calculations will then stop increasing or decreasing the output, because the error between SP and PROC will be zero.

Again, In the Woodward PID, PID_2, PID_DB, PID_SAMPLE, and PID_OPTI GAP blocks the INT_GN units are in Hz. A *higher* number results in *faster* accumulation in the Integral Calculations.

DERIV or S_D_R

If the Proportional Calculations use the instantaneous error and the Integral Calculations use the past error, then the Derivative Calculations use the future error. Since we cannot know the future, the Derivative Calculations predict the future, based on the change of the error in the past. The derivative is literally the slope of the error curve. Using derivative can be very useful in preventing overshoot in the response of a system because the Derivative Calculations help anticipate the system. One trade-off is that derivative calculations are more sensitive to noise.

In these Woodward PID GAP blocks, S_D_R (Speed Derivative Ratio) is used instead of derivative, to reduce the number of "things" a user has to tune. With S_D_R, DERIV changes automatically when INT_GN is tuned. This typically works well, in most published tuning algorithms, such as the Ziegler Nichols algorithm, DERIV is usually a function of INT_GN.

In most of the Woodward PIDs, DERIV (derivative) is calculated from INT_GN and S_D_R. The user can calculate S_D_R from INT_GN and DERIV, using the equations below. In addition to determining the derivative, S_D_R determines whether the PID is "Input Dominant" or "Feedback Dominant".

If S_D_R > 1 we call this "Feedback Dominant", DERIV = 1/(S_D_R*INT_GN), S_D_R = 1/(INT_GN* DERIV)
If S_D_R < 1 we call this "Input Dominant", DERIV = S_D_R/INT_GN, S_D_R = INT_GN* DERIV

"Feedback Dominant" and "Input Dominant" are discussed in Chapter 2, in the S_D_R section.

If a user would rather tune DERIV than S_D_R, the equations above should be implemented in the GAP using individual GAP blocks or a user block. The resultant PID system will be a PID with PROP_GN, INT_GN, and DERIV, rather than PROP_GN, INT_GN and S_D_R.

Again, the plot in Figure 3 shows the individual contributions from PROP_GN, INT_GN, and DERIV. Note that the Derivative Calculations are related to the change in error.

S_D_R is a ratio and is unit-less. The units of DERIV as discussed in this Application Note are in seconds or 1/Hz.

Tuning the PID

There are many different published PID tuning algorithms. There are also resources for PID tuning help, available in books or on the internet through various universities.

An important system consideration is that before we can tune a PID, we need to understand what we want for a response. If we have an engine or turbine running off line, we might want a fast response, to help prevent overspeed. This might mean that we have a response with overshoot, to increase the system responsiveness. A very fast response is typically required to help the turbine stay on line and to prevent overspeed trip during a load rejection event.

On the other hand, if we have a pressure control loop and we want to be careful not to move the header pressure too much, we might want a turbine that responds slowly, without overshoot.

One of the biggest challenges of PID tuning can be to understand the system that the PID is controlling, including the system scaling, the system requirements, and the other controllers in the system.

The PID_OPTI GAP block will tune the PID for the user, automatically. Refer to the GAP help for more information.

Chapter 2.
Details

Chapter 2 provides more information about the Woodward PID GAP blocks, including information about "Woodward Specific" block inputs. The inputs discussed here are the FEEDBACK, FEEDBACK1, THRESH, RATE_CTRL, P_ONLY, INT_GN_MN, and S_D_R block inputs. This information is supplemental to the GAP help and related documents.

FEEDBACK

The FEEDBACK input to the PID GAP block connects to the HSS/LSS bus output or to one of the downstream GAP blocks that drives the actuator. This is an analog input.

The FEEDBACK input is used to implement the Integral Calculations and Derivative Calculations in the PID equations, to allow the PID to track the actuator demand when it is not in control. This prevents integrator windup. When the PID is not in control) the PID output will be x, where x is defined by:

x = FEEDBACK +/- (SP – PROC)*function of(PROP_GN, INT_GN, S_D_R)

As shown in the equation, when the PID is not in control, its output will differ slightly from the FEEDBACK input, based on the error, and on PROP_GN, INT_GN, and S_D_R. As a result, the FEEDBACK input should come from a GAP block that drives the actuator demand, to prevent integrator windup. If there is scaling between the PID and actuator (for example in an extraction control, due to ratio limiter logic), the FEEDBACK input needs the inverse scaling, so that feedback scaling matches output scaling.

The FEEDBACK input includes a Z^{-1}, to establish the execution order of the loop that is formed when the FEEDBACK input is connected to the PID output, either through other blocks or directly. An additional Z^{-1} is not needed, and if used, limits the dynamic performance of the PID GAP block.

FEEDBACK1

The FEEDBACK1 input connects to the SEL_X of the local LSS (Low Signal Select) or HSS (High Signal Select) bus. This input lets the PID know when it is control of the LSS and or HSS bus. The PID also uses this input to tell whether it is connected to an LSS or to an HSS bus. This is a Boolean input.

The FEEDBACK and FEEDBACK1 inputs are used in conjunction with the THRESH input, to allow the PID to move out of the way of other controls on the LSS or HSS bus. THRESH is then used to tell the PID how much error the PID should see when it is NOT in control, before it drives its output to minimum or maximum.

This is discussed more in the THRESH section. An example of the use of the THRESH and FEEDBACK inputs is shown in Figure 4.

If there is not a downstream LSS or HSS bus the FEEDBACK1 input should be set to TRUE.

> **Note:** The PID_OPTI GAP block uses the LSS_HSS bus input rather than the FEEDBACK1 input to determine whether to go high or low when it is out of control. This allows the GAP application to connect to multiple GAP blocks (such as Analog Name GAP blocks) between the HSS or LSS bus and the FEEDBACK1 input.

THRESH

The THRESH value is used by the user to tell the PID how far out of control it should be before it moves out of the way on the LSS or HSS bus. Setting up a control to work properly with fast and slow PIDs requires understanding how to set the value of THRESH. This input is not needed if there is not an LSS or HSS bus in the application.

The THRESH input lets the user affect the behavior of the PID GAP block when coming into or out of control. It is typically used to prevent a slower PID (a slower Rate Group or slower dynamics) from interfering with a faster PID. THRESH only affects the behavior of the block when it is NOT in control. When set to a low value, the slower PID would not interfere on the LSS or HSS bus unless the PROC is near the SP (within THRESH).

The FEEDBACK1 and THRESH inputs work together. When the error value (SP – PROC) is greater than THRESH, and FEEDBACK1 is FALSE, the PID output will:

- If FEEDBACK1 is connected directly to an LSS (or the LSS_HSS input of the PID_OPTI block is 1), the output will go to MAX_VAL + 1, typically 101%. For the PID_OPTI block, MAX_VAL is fixed at 100.
- If FEEDBACK1 is connected directly to an HSS (or the LSS_HSS input of the PID_OPTI block is 0), the output will go to MIN_VAL - 1, typically -1%. For the PID_OPTI block, MIN_VAL is fixed at zero.
- If FEEDBACK1 is TRUE or connected to neither an LSS or HSS, THRESH has no effect.

It is important to note that even though the PID output is -1 or 101, the Integral Calculations are NOT winding up/down. The GAP diagram in Figure 4 demonstrates the use of FEEDBACK, FEEDBACK1, and THRESH. In the figure, the S_PID is out of control, at 101%, and the T_PID output is 0%. The T_PID has control of the LSS bus, which is also outputting 0%.

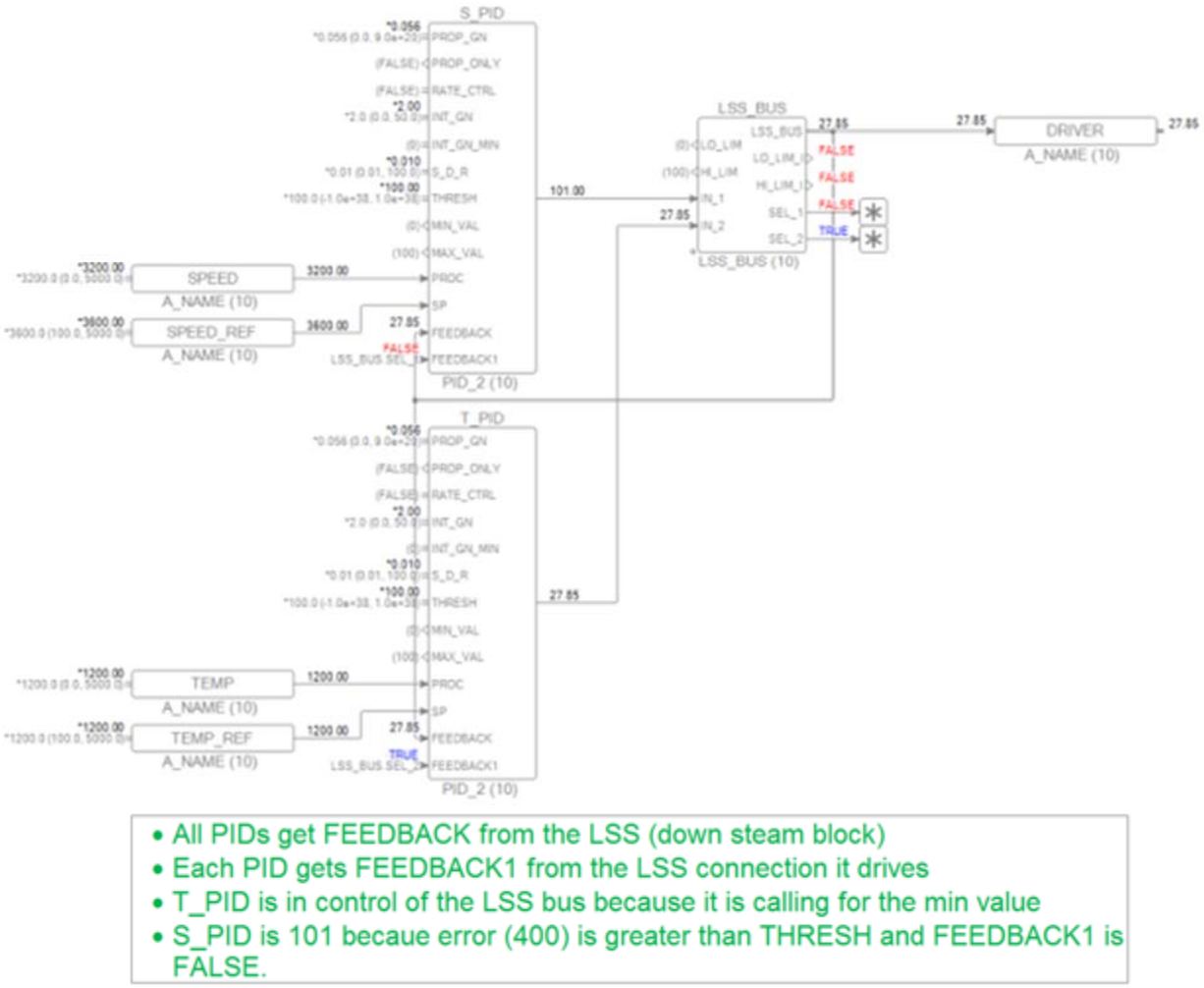
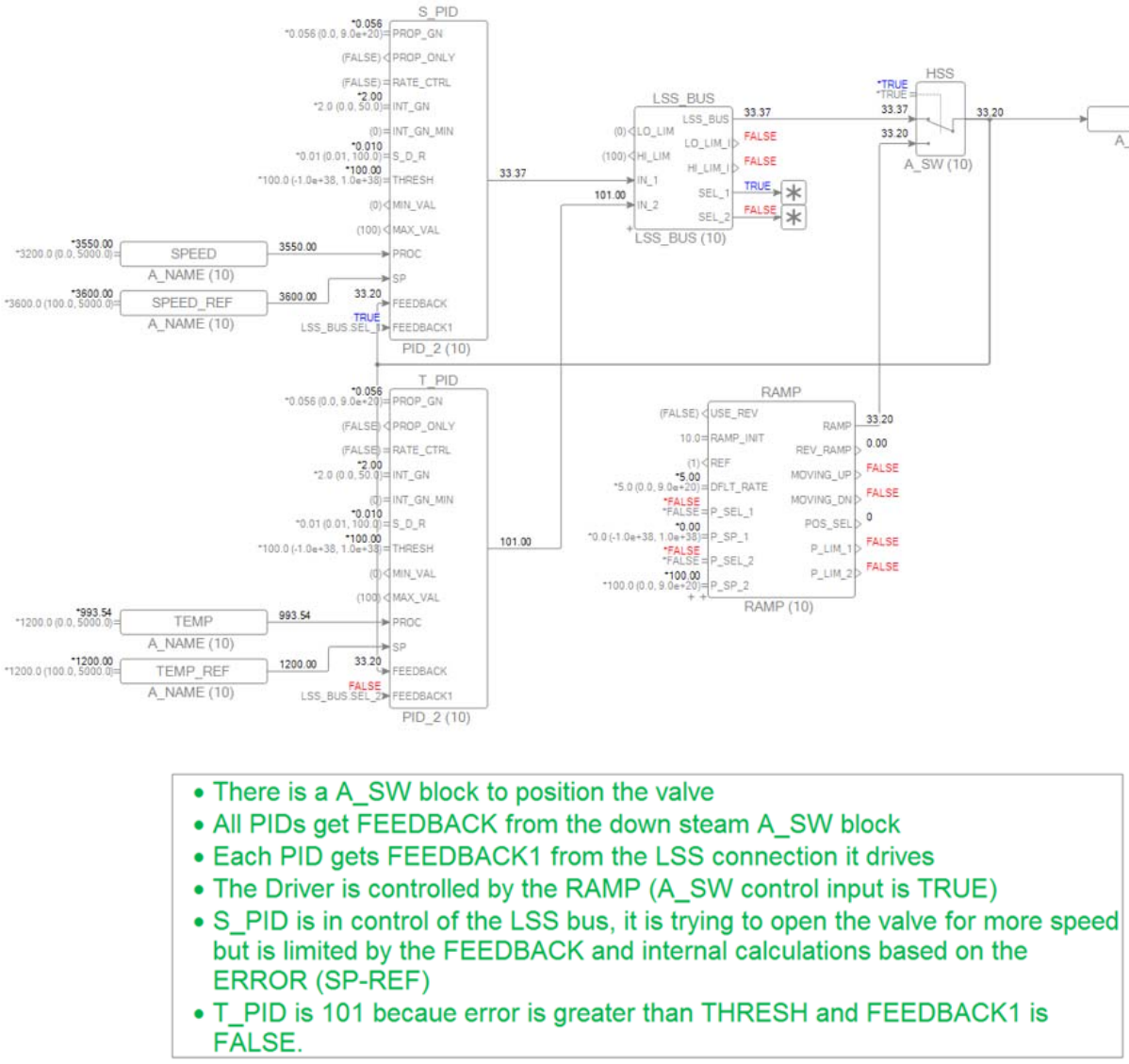


Figure 4. Example GAP Application, 2 PIDs and an LSS block

Again, the reason for the FEEDBACK1 and THRESH functionality is to ensure that slower PIDs do not affect the control of faster PIDs when the slower PIDs are not in control.

For example, if a PID is running in a 10mS rate group and another PID on the same LSS bus is running in an 80mS rate group, the PID running in the 10mS rate group (fast PID) will run 8 times while the slow PID will only run once. When the fast PID is in control of the LSS bus, the slow PID will calculate a slightly higher value based on its error. When the fast PID drives a large change to the output which exceeds the value of the slow PID, the slow PID may take control even though you might expect the fast PID to maintain control. This is a transient situation, when the slow PID runs it will move to a greater value than the fast PID. By moving the slow PID out of the way (to 101), it will not interfere with control unless the PROC exceeds the SP, therefore it will not hinder the response of the fast PID.

Figure 5 demonstrates a system in which the actuator can be switched to a different source. The PIDs track this, through the FEEDBACK inputs. The values in Figure 5 are in the state when the S_PID controls the LSS bus, and the ramp controls the switch.



- There is a A_SW block to position the valve
- All PIDs get FEEDBACK from the down steam A_SW block
- Each PID gets FEEDBACK1 from the LSS connection it drives
- The Driver is controlled by the RAMP (A_SW control input is TRUE)
- S_PID is in control of the LSS bus, it is trying to open the valve for more speed but is limited by the FEEDBACK and internal calculations based on the ERROR (SP-REF)
- T_PID is 101 becaue error is greater than THRESH and FEEDBACK1 is FALSE.

Figure 5. Example GAP Application, 2 PIDs, LSS, Switch, and Ramp

In Figure 5, notice that the FEEDBACK1 inputs to S_PID and T_PID do not reflect the state of the switch, they come from the LSS_BUS SEL_1 and SEL_2 outputs.

The plot in Figure 6 shows the response of the control in Figure 5, as the output is switched between a PID and a ramp.

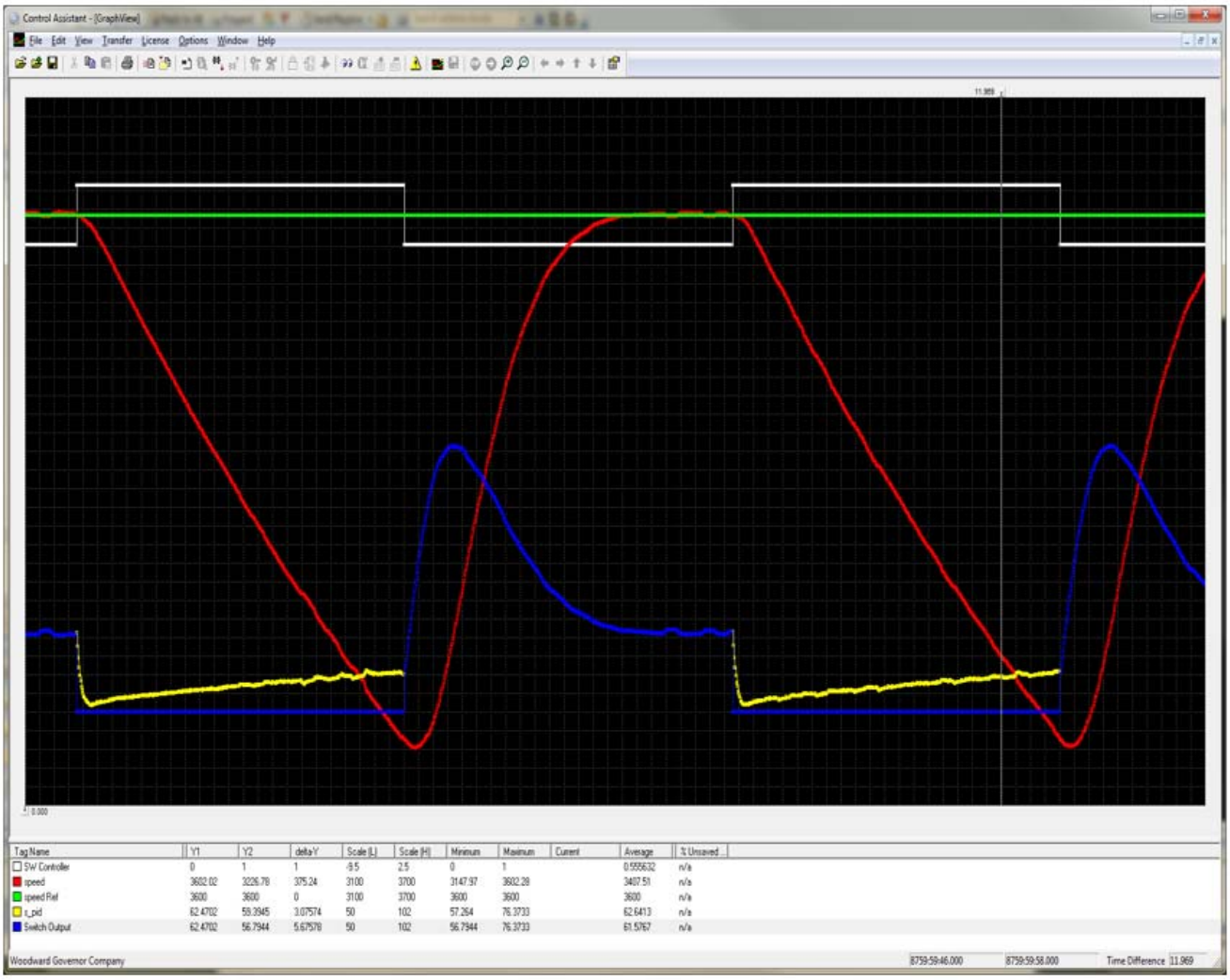


Figure 6 – Plot of the Response of Figure 5

Colors definitions are:

- White is the command to the switch. The switch determines whether the output of the control is driven by the PIDs or set to zero. (Low is PID, high is zero)
- Green is the speed setpoint (SP); note that it is constant for this simulation.
- Red is Speed (PROC). When the PID is in control the speed matches the speed setpoint (PROC = SP). When the actuator is switched to zero, the speed (PROC) decreases. When the switch control becomes low, the PID takes control, its output starts at FEEDBACK, and increases until the PROC matches SP again.
- Blue is the control output, when it is zero the switch is in control, when it is moving the PID is in control.
- Yellow is the PID output. It is the same as the control output (blue trace) when the PID is in control, and moves up based on the error when the PID is not in control. Note that this is not integrator windup.

Again, the THRESH, FEEDBACK1, and LSS_HSS (PID_OPTI only) only affect the behavior of the PID when it is NOT in control (when FEEDBACK1 is FALSE).

RATE_CTRL

The RATE_CTRL input allows the PID to implement a lead term, which is typically necessary when the PID is used for accel (acceleration) or decel (deceleration) control. When RATE_CTRL is used, the system should be analyzed and/or simulated to ensure that the range of responses is acceptable and understood.

An accel or decel control does not have an integrator in the control loop as a speed control has, because the accel/decel signal is the derivative of speed. RATE_CTRL implements an integrator before the PID, giving a “lead” term. An example of accel control is shown in Figure 7.

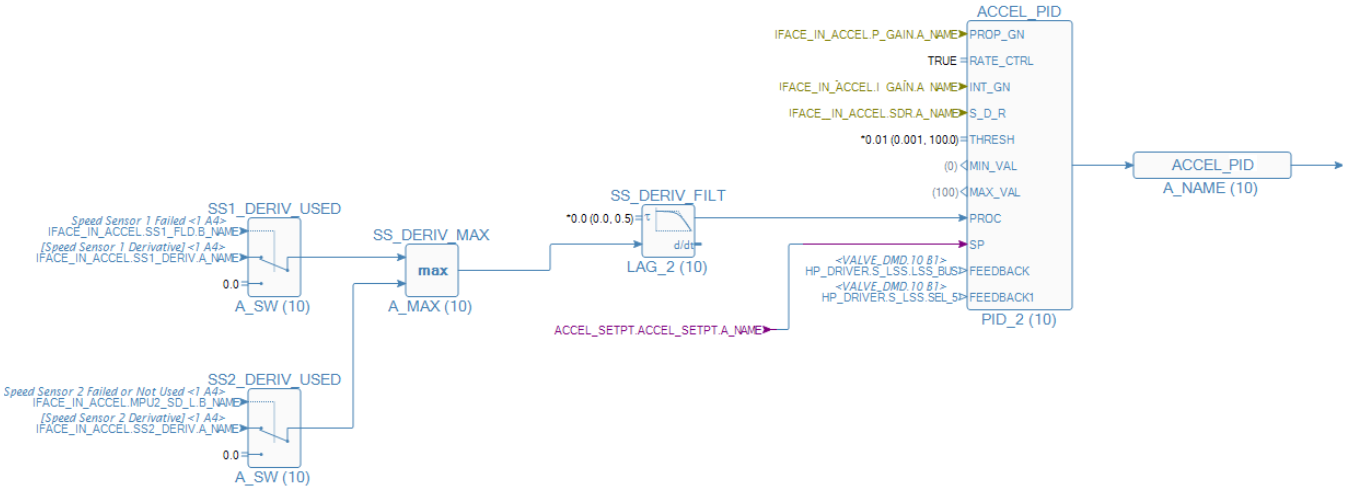


Figure 7. GAP Diagram Example of Acceleration Control

Accel control is similar to speed control, except the process variable is the calculated derivative of the speed signal. As derivative calculations can be noisy, a lag is sometimes used for optional filtering. Again, for acceleration control it is very important to simulate the system before implementing it.

P_ONLY and INT_GN_MN

The P_ONLY input is a way for a user to configure the PID for “Proportional Only” control. INT_GN must be less than INT_GN_MN to use P_ONLY, to minimize process errors.

Proportional control is useful for Ziegler Nichols tuning, the oscillation method of system measurement uses a proportional mode of control. Proportional control is also used for systems which do not need the error correction of an integrator. Note if there is no integrator, a PID will not drive the error to zero. Shifts in the system operating conditions will result in an error between SP and PROC.

Note: the P_ONLY and INT_GN_MN inputs are not available on the PID_OPTI GAP block.

S_D_R

Woodward uses the S_D_R input rather than a DERIV input to try to make “tuning” easier for the user, because the user can tune two variables, rather than three. As INT_GN and DERIV are calculated from the same system variables in most of the published tuning formulas, this typically works well.

A user can tune DERIV, rather than S_D_R, by using the following formulas. Caution should be used in setting DERIV independently of INT_GN.

If S_D_R is <= 0.01 or >= 100 the code is switched to use a PI control, rather than a PID. The effect of the Derivative Calculations is set to zero.
If S_D_R > 1 we call this “Feedback Dominant”, DERIV = 1/(S_D_R*INT_GN)
If S_D_R < 1 we call this “Input Dominant”, DERIV = S_D_R/INT_GN

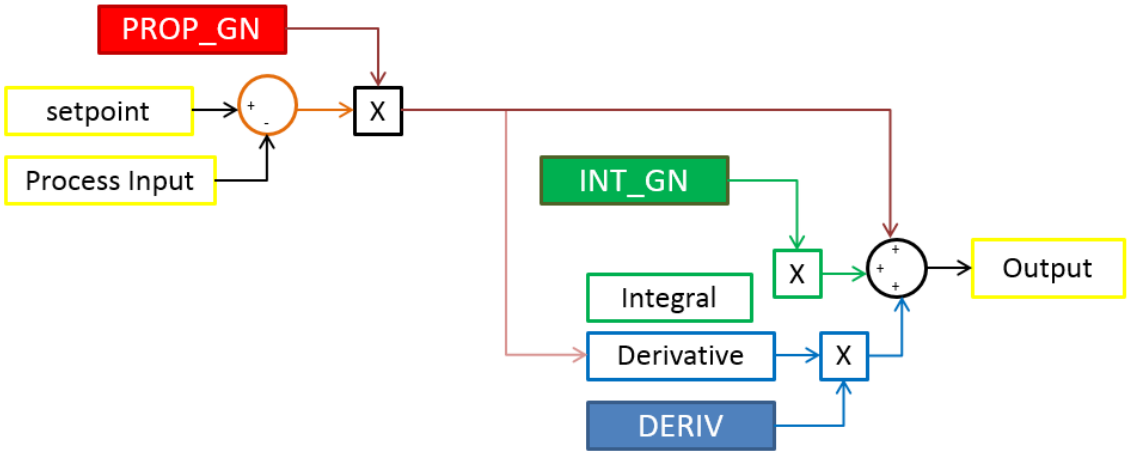


Figure 8. Diagram of a Basic PID

Input and feedback dominant modes result in EXACTLY the same PID equations when the PID is in control.

The difference between input and feedback dominant is in the implementation of the equations, which affects noise susceptibility and behavior when the PID is coming into or out of control.

An “Input Dominant” configuration results in less filtering on the error, letting the PID react to input changes more quickly, before PROC reaches SP. This mode would respond well to PROC disturbances, but would have an increased sensitivity to noise. As a result, input dominant mode might be useful for speed control, because good load rejection performance can be important and difficult to achieve.

A “Feedback Dominant” configuration results in less filtering on the feedback, letting the PID react to feedback changes more quickly. This would be useful for temperature control, when reacting to changes on the LSS bus is important, to allow the temperature controller to take control more quickly. This system would have a decreased sensitivity to noise, but would not respond as well to temperature disturbances.

Again, Input Dominant and Feedback Dominant only affect the PID when it is coming into or going out of control.

Chapter 3.
Integrator Wind-Up

A topic of concern for many control system engineers is integrator wind-up. This refers to the integrator driving the output of the PID to or past its output limits.

Integrator windup typically occurs if the PID is not in control of the process. The effect of the INT_GN is to drive the error to zero. If it cannot, in other words if the control has not moved the process enough, there will be a consistent error. Even with a small consistent error, the Integral Calculations will keep growing. They will grow faster with higher INT_GNs, and slower with lower INT_GNs.

The plot in Figure 3 and again in Figure 9 shows the individual contributions from PROP_GN, INT_GN, and DERIV. Note that the Integral Calculations in pink drive the PID output up, and the error down.

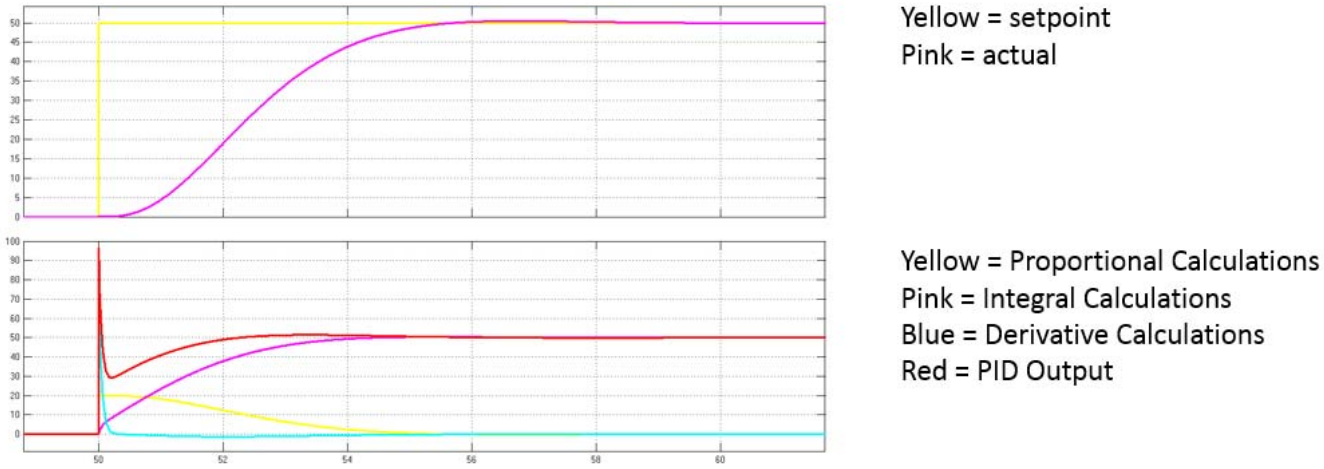


Figure 9. Components of the PID Response

In some other products in the industry, the Integral Calculations can continue to increase or decrease, even when the PID output is at its limit or when the PID is not in control. This is very undesirable because it can take the PID a long time to respond to a change in its inputs, if it has to integrate back up/down from a low/high number. During the delay the system is not well controlled, the process can overshoot or undershoot significantly. In products like this, this problem must be addressed by using additional logic to prevent these conditions.

In Woodward Controls, the FEEDBACK input is used by the integrator. If the FEEDBACK input does not match the PID output, the PID output will track the FEEDBACK input. This prevents the Integral Calculations from diverging from the FEEDBACK input, and from “winding up”.

Chapter 4.
Conclusions

This App Note aids in general understanding of PIDs, and in understanding the details of some of the Woodward GAP PIDs, including the PID, PID_2, PID_OPTI, PID_SAMPLE, and PID_DB GAP blocks.

The Woodward GAP PIDs discussed here have the following characteristics:

- They use the most popular parallel form of the PID with PROP_GN also impacting the Integral and Derivative Calculations, to simplify tuning for the user.
- They use S_D_R instead of derivative to allow tuning of INT_GN to adjust the derivative automatically. Users can still tune DERIV individually, by using the equations provided in this document and in the GAP help.
- They include two different modes for use with different control loops when coming into and going out of control.
  - Input dominant mode, which will help prevent overshoot of the set point, typically in speed loops.
  - Feedback dominant mode, which can keep slower control loops from interfering with faster control loops. This is typically used in temperature topping limit controllers.
- They control integrator windup with overall feedback to/from other control loops.
- They include the THRESH and FEEDBACK1 inputs to help keep slower control loops from influencing faster loops when it is undesirable.
- They include a RATE_CTRL input for control of derivative processes.

We appreciate your comments about the content of our publications.

Send comments to: icinfo@woodward.com

Please reference publication 51556.

B 5 1 5 5 6 : N E W

PO Box 1519, Fort Collins CO 80522-1519, USA
1041 Woodward Way, Fort Collins CO 80524, USA
Phone +1 (970) 482-5811

Email and Website—www.woodward.com

Woodward has company-owned plants, subsidiaries, and branches, as well as authorized distributors and other authorized service and sales facilities throughout the world.

Complete address / phone / fax / email information for all locations is available on our website.