

2104301-001– rev. AB

PID Control

Application Guide

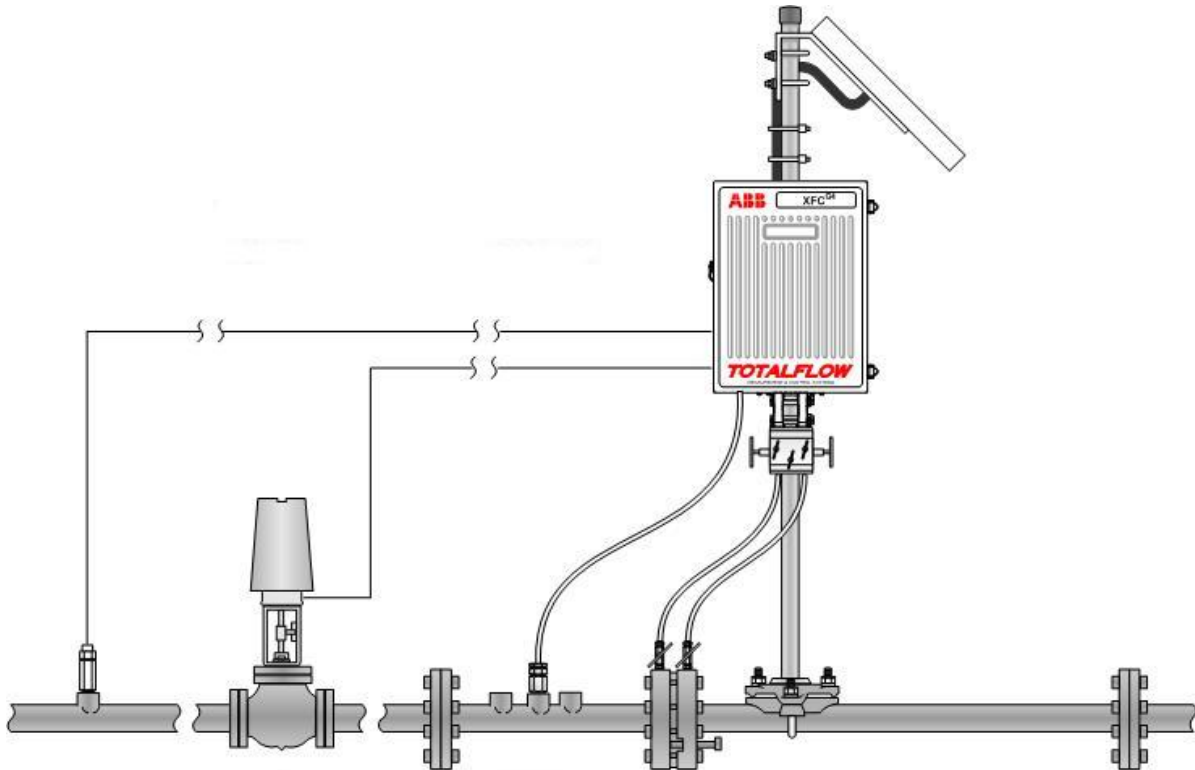


TABLE OF CONTENTS

1.0	INTRODUCTION	3
1.1	Overview	3
1.2	PID Control Example	3
2.0	TOTALFLOW'S IMPLEMENTATION OF THE PID CONTROL.....	4
2.1	Overview	4
2.2	Dead Band	6
3.0	PCCU SCREENS	6
3.1	Operation Tab	7
3.1.1	Override	7
3.1.2	Additional Variable Fields	7
3.2	Advanced Tab	9
3.3	Inputs/Outputs Tab	11
4.0	CONFIGURING PID FOR PCCU32	13
4.1	Typical PID Control Scenario	13
4.1.1	Assumptions.....	14
4.2	Enabling the AGA-3 and PID Controller Applications.....	14
4.3	AGA-3 Setup	14
4.4	Calibrating the Analog Outputs for 4-20 mA Range	15
4.5	Setting Up the PID Application	16
4.5.1	Operation Tab	16
4.5.2	Advanced Tab.....	17
4.5.3	Inputs /Outputs Tab	18
5.0	CONTROLLER INITIAL VALUES AND SYSTEM TUNING	19
5.1	Scale Factor	19
5.2	Derivative Filter	20
5.3	Integral Factor	20
5.4	Derivative Factor	20
6.0	HOLDING REGISTERS	21
6.1	Application Notes	28

1.0 INTRODUCTION

1.1 Overview

The PID controller calculation involves three separate parameters: the Proportional, the Integral and the Derivative. These three parameters form the PID calculation. The proportional value determines the reaction to the current error; the integral value determines the reaction based on the sum of recent errors and the derivative value determines the reaction based on the rate at which the error has been changing. The weighted sum of these three actions is used to adjust the process via a control element such as the position of a control valve or the power supply of a heating element.

By tuning these three constants in the PID algorithm, the controller can provide control action designed for specific process requirements. The response of the controller can be described in terms of the responsiveness of the controller to an error, the degree to which the controller overshoots the set point and the degree of system oscillation. It should be noted that the use of the PID algorithm for control does not guarantee optimal control of the system or system stability.

Some applications may require using only one or two of the parameters to provide the appropriate system control. This is achieved by setting the gain of undesired control outputs to zero. A PID controller will be called a PI, PD, P or I controller in the absence of the respective control actions. PI controllers are particularly common, since derivative action is very sensitive to measurement noise, and the absence of an integral value may prevent the system from reaching its target value due to control action.

1.2 PID Control Example

A common example of a control loop would be the actions taken to keep one's shower at the ideal temperature. This is a process that typically involves the mixing of two process streams: cold and hot water. A person would feel the water to estimate its temperature. Based on this measurement, they then perform a control action that involves using the cold water tap to adjust the process. The person would repeat this input-output control loop, adjusting the hot water flow until the process temperature stabilized at the desired level.

Feeling the water temperature is taking a measurement of the process value or process variable (PV). The desired temperature is called the set point (SP). The output from the controller and input to the process (the tap position) is called the manipulated variable (MV). The difference between the measurement and the set point is the error (e), i.e., too hot or too cold and by how much.

As a controller, the person decides roughly how much to change the tap position (MV) after the temperature is determined (PV), and therefore, the error. This first estimate is the equivalent of the proportional action of the PID controller. The integral actions of a PID controller can be thought of as gradually adjusting the temperature when it is almost at the correct temperature. Derivative action can be thought of as noticing the water temperature is becoming hotter or colder, how fast, anticipating further change and then adjusting for a soft landing at the desired temperature (SP).

Making a change that is too large when the error is small is equivalent to a high gain controller and will lead to overshoot. If the controller were to repeatedly make changes

that were too large and repeatedly overshoots the target, this control loop would be termed unstable, and the output would oscillate around the set point in either a constant, growing or decaying sinusoid. A human would not do this because our adaptive controllers learn from the process history, but PID controllers do not have the ability to learn and must be set up correctly. Selecting the correct gains for effective control is known as tuning the controller.

If a controller starts from a stable state at zero error ($PV = SP$), then further changes by the controller will be in response to changes in other measured or unmeasured inputs to the process that impact on the process, and accordingly, on the PV. Variables that impact on the process, other than the MV, are known as disturbances. Generally, controllers are used to reject disturbances and/or implement set point changes. Changes in feed water temperature constitutes a disturbance to the shower process.

In theory, a controller can be used to control any process which has a measurable output (PV), a known ideal value for that output (SP) and input to the process (MV) that will effect the relevant PC. Controllers are used in the industry to regulate temperature, pressure, flow rate, chemical composition, speed and practically every other variable for which a measurement exists. Automobile cruise control is an example of a process which utilizes automated control.

Due to the long history, simplicity, well grounded theory, simple setup and maintenance requirements, PID controllers are the controllers of choice for many of these applications.

2.0 TOTALFLOW'S IMPLEMENTATION OF THE PID CONTROL

2.1 Overview

In the following discussion, several terms are used interchangeably:

- Proportional and Gain
- Integral and Reset
- Derivative and Rate
- Manipulated Variable (MV) and Individual PID Output
- Xout and selected output
- Process Variable (PV) and Measured Value (m)
- Scale and Normalizing Factor

The user's particular control situation may require a single PID controller that controls a single process variable; however, Totalflow has implemented PID control in such a way as to allow multiple controllers to manipulate a single process variable. A single PID application supports a Primary and Override controller.

For example, if the user wishes to control on flow rate only, a single PID (Primary) can perform the task sufficiently. In many situations, the user may be required to control not only on flow rate but on a down stream pressure limit (i.e., a down stream sales point that cannot allow pressures higher than 600psi). In this situation, the user may want to implement two PID controllers (Primary and Override): one controlling on flow rate (Primary) and one controlling on the down stream pressure (Override).

Totalflow's PID implementation requires a few more variables. These variables will decide which PID controller's (Primary or Override) manipulated variable (MV) will be

selected to drive the process (valve position). The value of the selected MV is passed back into the calculation of both controllers as a tracking variable (TK). Passing the tracking variable back into calculations keeps the unselected PID output (MV) in the range of the selected PID output (MV). This prevents the unselected PID output from "winding up". If and when the unselected PID becomes the selected PID, its MV will be closer to the value it is trying to achieve.

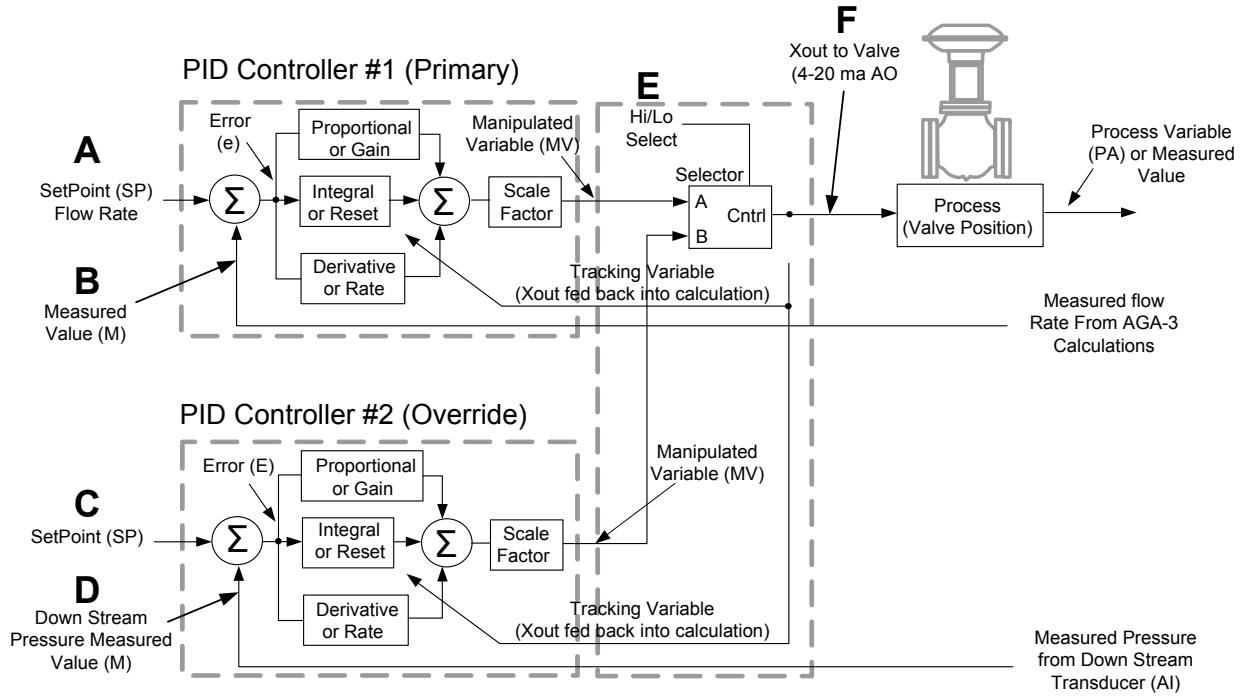


Figure 1 PID Diagram

Table 1 Key Points in the PID Diagram

ID	Descriptor	Definition
A	Flow Rate Set Point	Desired flow rate is set by user.
B	Measured Flow Rate	Actual flow rate comes from AGA-3 calculation.
C	Down Stream Pressure Set Point	Desired limit for down stream pressure is set by user.
D	Measured Down Stream Pressure	Actual down stream pressure comes from an external transducer.
E	Lo/Hi Select	Desired controlling output is selected by user.
F	Xout to Valve	4-20 mA AO to control valve position to maintain desired flow rate and not exceed down stream pressure limit.

2.2 Dead Band

Totalflow's PID controllers implement a dead band. The dead band represents the range over which the primary controller will allow the process variable (PV) to deviate without exerting any correction. For example, imagine the user is attempting to maintain a flow rate of 400MCF. If that flow rate were to fluctuate between 395 and 405MCF, the controller would exert no correction. The dead band would be 10MCF.

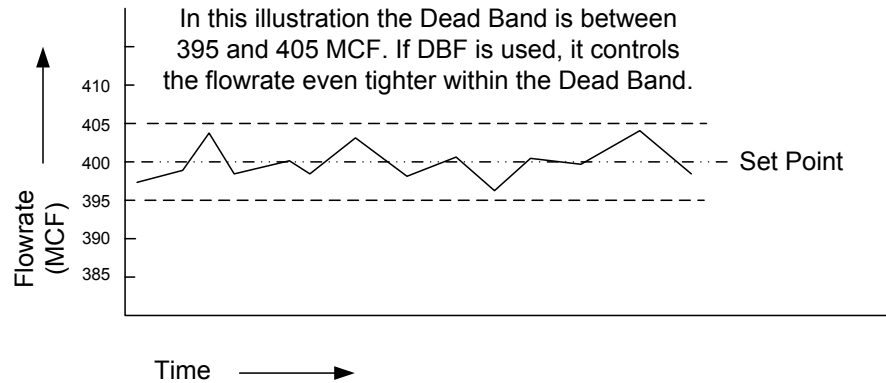


Figure 2 Dead Band

In an effort to afford even tighter control over the process variable (i.e., flow rate), Totalflow has incorporated a dead band gain factor (DBF). The DBF (if set to a non-zero value) allows the operator to have some control over operation within the dead band. If the DBF is used, it is multiplied by the scale factor (KN or normalizing gain) to determine the controller output (MV). If DBF is set to zero, no control action is exerted within the dead band.

3.0 PCCU SCREENS

The Operation tab displays three “scales” and a table with three columns. The first column lists the variables by name. The second column lists the variable values for the Primary PID. The third column lists the variable values for the Override PID. The grayed out variables (Process Variable and Output) are read only. They monitor those variables and cannot be “set” by the operator. The three “scales” (PV, SP and Output) provide a live, visual indication of those three variables.

3.1 Operation Tab

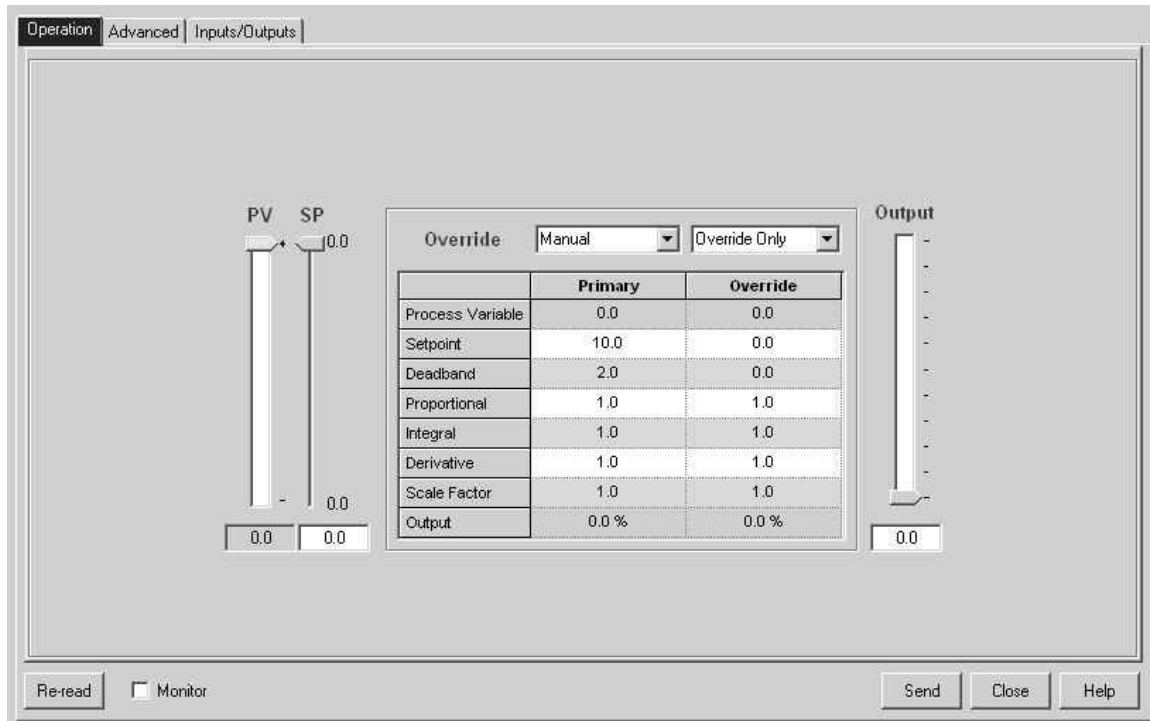


Figure 3 Operations Tab

The following will provide details on the assorted variables that are encountered within the tab.

3.1.1 Override

- **Manual** - In Manual mode, the user can set the PID output manually using the Output slider. To achieve a bump-less transfer between Auto and Manual mode, the set point is set equal to the measured value (PV). The error ($SP - m$) would be zero, and therefore, no “bump”. If the user wants the Primary and the Override PIDs to track during Manual mode operation, Manual Mode Tracking should be enabled.
- **Auto** - Auto is the default and the normal mode of operation for the PID application. In Auto mode, the PIDs will determine the PID outputs not the user.
- **Primary Only** - Only one controller (Primary) is being implemented. In the example, the user would be controlling strictly on flow rate with no pressure override.
- **Override Only** - Only one controller (Override) is being implemented. In the example, the user would be controlling strictly on down stream pressure with no control of flow rate.
- **Override Control** - Both PIDs are being implemented (Primary and Override). In the example, the user would be controlling the flow rate with the Primary PID and using the Override PID to control down stream pressure.

3.1.2 Additional Variable Fields

The following are the additional variable fields that the user will encounter in the Operations tab (see Table 2).

Table 2 Operations Tab Variable Fields

Variable	Description
Process Variable (read only)	The Process Variable is the parameter being controlled. In the illustration above, PID Controller #1 is controlling flow rate. In the example, the PID controller controls the Process Variable flow rate by opening and closing a valve. As the valve opens, flow rate is increased. As the valve is closed, flow rate decreases.
Set Point (read/write)	The Set Point is the value at which the user is trying to control the Process Variable. In the example above, PID Controller #1 is maintaining the Process Variable (flow rate) at the Set Point of 400 MCF.
Deadband (read/write)	<p>The Deadband creates a “window” in which the PID controller maintains the system output. Generally, as long as the Process Variable (PV) is within this Deadband “window”, no corrective measures are taken by the PID controller. However, if the Dead Band Factor (DBF) is employed, the controller will continue to exercise control over the PV while in the Deadband “window”. The Deadband defaults to zero (0), and the DBF (Deadband gain) defaults to 1. These default values are equivalent to not having a Deadband.</p> <p>Setting the DBF to a value of 0.0 is the same as not changing the output while operating inside of the Deadband. Use a value between 0 and 1 to reduce the change effect but still limit valve changes. Setting this factor to a value greater than 1 will increase the total gain and make output response larger than normal (high gain preset action). This factor only effects the controller output when the Process Variable is within the user-specified Deadband.</p>
Proportional Gain (read/write)	Proportional term, sometimes referred to simply as Gain, makes a change to the output that is proportional to the current error value. The larger the error signal (PV-SP), the larger the correction that will be made to the manipulated variable. Large gains can cause a system to become unstable. Low gains can cause the system to respond too slowly to system disturbances.
Integral or Reset (read/write)	The Integral, or Reset Term, accumulates the instantaneous error value over a defined period of time and provides corrections based on this accumulated error. The amount of correction is based on the Integral value. When used in conjunction with the Gain factor, the Integral value helps to drive toward the set point, eliminating steady-state error. Since the Integral value is based on past error values, it can cause overshoot of the present value.
Derivative or Rate (read/write)	The rate of change of the error signal is the basis of the Derivative (or Rate) factor. The Derivative factor tends to slow the rate of change of the process variable. This behavior tends to reduce the possibility of overshoot due to the Integral factor.
Scale or Normalizing Factor (read/write)	A positive value of Scale Factor results in ‘direct’ action. A negative value results in ‘reverse’ action. For additional information, please see 5.1 Scale Factor.
Output (read only)	The Output value is the signal controlling the controlled process. In the example, the controlled process is the valve controlling flow rate. The Output is the signal driving that valve open and closed to control flow rate.

3.2 Advanced Tab

	Description	Value
11.253.0	Control Loop Description	
11.252.10	Primary SP Low Limit	1.0000
11.252.11	Primary SP High Limit	0.0000
11.252.14	Primary SP Ramp Rate	0.0000
11.252.12	Override SP Low Limit	0.0000
11.252.13	Override SP High Limit	0.0000
11.252.15	Override SP Ramp Rate	0.0000
11.252.28	Override Threshold	0.0000
11.250.5	Override Type	Low
11.252.16	Output Low Limit	0.0000
11.252.17	Output High Limit	100.0000
11.251.0	Minimum Control Time	00:00:00
11.250.8	Manual Mode Tracking	Disabled
11.250.7	Controller Reset Mode	Manual

Figure 4 Advanced Tab

The following are the fields that the user will encounter when they are in the Advanced tab.

Table 3 Advanced Tab Variable Fields

Variable	Description
Primary SP Low Limit	This user-defined field enables the user to establish the low limit that the setpoint cannot be set outside of. The entered value also provides scaling which is used to display the scales on the Operation tab.
Primary SP High Limit	This user-defined field enables the user to establish the high limit that the setpoint cannot be set outside of. The entered value also provides scaling which is used to display the scales on the Operation tab.
Primary SP Ramp Rate	This user-defined field enables the user to establish the setpoint ramp rate that allows for the ramping of the setpoint from one value to the next.
Primary Derivative Filter	The Derivative filter is used to filter noise out of the process variable. This allows for better derivative control and reduces the effects of system noise. Without this, the Derivative Factor can be adversely affected by the noise. The default for this value is set to 1.0.
Primary Dead Band Factor	This user-defined field is the gain factor to be used within the Deadband of the controller. A value of 1.0 is equivalent to not having a Deadband. Setting the value to 0.0 is equivalent to not having gain control within the Deadband. A value between 0 and 1 reduces the overall gain and limits process (i.e., valve)

Variable	Description
	changes. Setting this to a factor greater than 1 increases the total gain and output response. This factor only affects the controller output when the process variable is within the user-specified Deadband.
Primary Output Scale Factor	In this user-defined field, a positive value will result in a 'direct' action. A negative value results in a 'reverse' action. For additional information, please see 5.1 Scale Factor.
Override SP Low Limit	This user-defined field enables the user to establish the limit that will override the established setpoint low limit.
Override SP High Limit	This user-defined field enables the user to establish the limit that will override the established setpoint high limit.
Override SP Ramp Rate	This user-defined field enables the user to establish the limit that will override the established setpoint ramp rate.
Override Derivative Filter	The Derivative filter is used to filter noise out of the process variable. This allows for better derivative control and reduces the effects of system noise. Without this, the Derivative Factor can be adversely effected by the noise. The default for this value is set to 1.0.
Override Dead Band Factor	This user-defined field is the gain factor to be used within the Deadband of the controller. A value of 1.0 is equivalent to not having a Deadband. Setting the value to 0.0 is equivalent to not having gain control within the Deadband. A value between 0 and 1 reduces the overall gain and limits process (i.e., valve) changes. Setting this to a factor greater than 1 increases the total gain and output response. This factor only effects the controller output when the process variable is within the user-specified Deadband.
Override Output Scale Factor	This user-defined field enables the user to establish a value that will override the established Override Output Scale Factor. For additional information, please see 5.1 Scale Factor.
Override Threshold	Override Threshold is an override setpoint adjustment that allows for early detection of the output select. Also, this setting allows the Override PID controller to stay in control longer before switching back to the Primary PID control. This parameter is only used during Override Control of the Primary PID.
Override Type	<p>The user has two selections that can be made within this field:</p> <p>High - After both PIDs have calculated their next value and presented that value to the selector, the selector will pass the higher value to X-out and the process being controlled (i.e., a valve).</p> <p>Low - After both PIDs have calculated their next value and presented that value to the selector, the selector will pass the lower value to X-out and the process being controlled (i.e., a valve).</p>
Loop Interval	This user-defined variable represents the number of seconds between processing runs on the PID controller.
Minimum Control Time	This parameter represents the minimum amount of time that one of the PIDs will have exclusive rights to the output value. The time is restarted each time that the output switches selection. No output switches can take place until the timer expires.
Manual Mode	The user has two selections that can be made within this field:

Variable	Description
Tracking	Enabled - Manual Mode Tracking is used to establish the setpoint of the control loop to the process variable when the loop is in Manual mode. The output of the control loop can be manually changed by the user when the controller is in Manual mode. Since the setpoint equals the process variable, there is no "bump" when the loop is placed in Automatic mode. This provides a "bump-less" transfer when changing the loop from Manual to Automatic mode. Disabled - This selection disables Manual Mode Tracking.
Controller Reset Mode	This parameter defines the operating mode that will be initiated subsequent to any power loss and restart of the PID controller. The two options available to the user are: Manual and Auto.
Full Closed Step Off	This variable is only used when the Output Type is Digital. This variable represents the time it takes to move a fully closed valve to some minimum opened condition. The units of this variable are milliseconds per count.
Valve Rest Interval	This variable is only used when the Output Type is Digital. This value is used to give a fixed pause between outputs after the valve position has been changed.
Digital Output Gain	This variable is only used when the Output Type is Digital. This is a scaling factor that converts the PID output value to an active time period, in milliseconds. It is recommended that a value of 1 be used in this field. The Output Low Limit and Output High Limit should be set (in units of milliseconds) to provide the amount of time the valve needs to travel from full-closed to full-open.

3.3 Inputs/Outputs Tab

	Description	Value
11.254.0	Primary PV	0.0.0
11.253.1	Primary PV Units	
11.254.1	Override PV	0.0.0
11.253.2	Override PV Units	
11.250.6	Output Type	Analog
11.252.16	Output Low Limit	0.0000
11.252.17	Output High Limit	100.0000
11.253.3	Output Units	%
11.254.2	Analog Output	0.0.0
11.250.9	TFIO VC Module	TFIO VC Module 0
11.254.3	Operator Chain	0.0.0

Figure 5 Inputs/Output Tab

Table 4 Input/Output Variable Fields

Term	Definition
Primary PV	This user-defined field enables the user to establish the registers used for the measured value (PV) controlled by the Primary, and if used, Override PID.
Primary PV Units	This user-defined field is a free-form text value used for display beside the process value.
Override PV	This user-defined field enables the user to establish the limit that will override the set measured value (PV).
Override PV Units	This user-defined field is a free-form text value used for display beside the process value.
Output Type	<p>There are two options that the user can select within this field:</p> <p>Analog - Analog specifies a continuous position type of controller used by 4-20 mA valves. The controller output would drive a 4-20 mA analog output.</p> <p>Digital - Digital specifies a stepper motor output. A positive (forward direction) or negative (reverse direction) DC voltage is applied for a specified period of time. Limit switches are employed to determine when the valve is fully open or fully closed.</p>
Output Low Limit	<p>This value clamps the output (X-out or the manipulated variable) at this limit. It prevents the output from going any lower. If the Output Type is set to Digital, this value should be set to the amount of time (in milliseconds) required to fully close the valve.</p> <p>Example: If 20 seconds are required to fully close the valve, this value should be set to 20,000.</p>
Output High Limit	<p>This value clamps the output (X-out or the manipulated variable) at this limit. It prevents the output from going any higher. If the Output Type is set to Digital, this value should be set to the amount of time (in milliseconds) required to fully open the valve.</p> <p>Example: If 20 seconds are required to fully open the valve, this value should be set to 20,000.</p>
Output Units	This variable is a free-form text value used for display beside the output value. If the Output Type is set to Digital, set this field to "msec".
Analog Output	This is the register (AO) used for the X-out (or manipulated variable) value being presented to the valve when the Output Type is set to Analog. For an Output Type of Digital, set this field to "0.0.0".
TFIO VC Module	This variable allows the user to select the TFIO Valve Control Module to be used for the Digital Output when Output Type is set to Digital. The drop-down box allows for the selection of TFIO VC modules 0-7.

4.0 CONFIGURING PID FOR PCCU32

Totalflow's XSeries (XFC and XRC) is used as a method of providing support for the PID application.

4.1 Typical PID Control Scenario

A common implementation of the Totalflow PID controller involves a simultaneous control of flow rate and down stream static pressure. PID #1 (Primary PID) is controlling flow rate at 400MCF, while PID #2 (Override PID) is being used to prevent down stream pressure from exceeding 1000PSI.

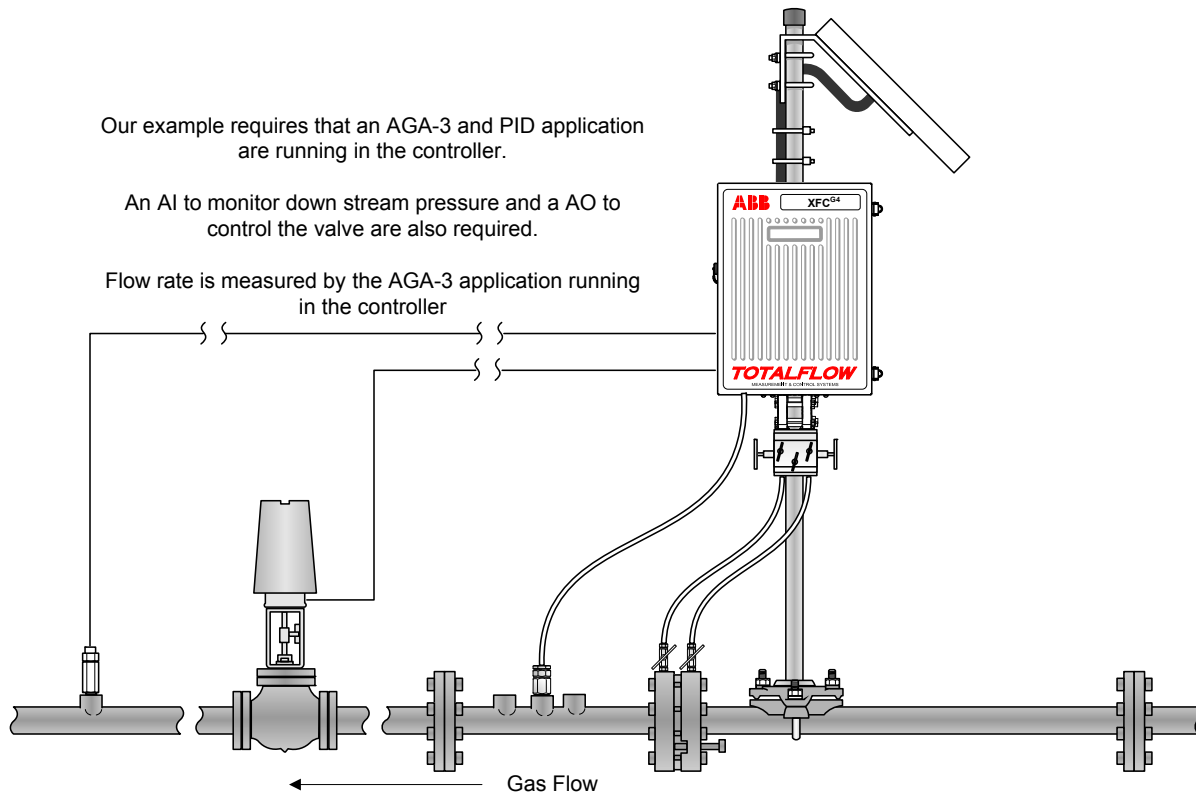


Figure 6 PID Scenario

For the scenario, the Totalflow controller is running an AGA-3 application for flow rate and the PID application for system control. An AI is used to monitor a down stream pressure transducer and an AO is used to control the main valve. Refer to the above illustration for a graphical representation of this scenario.

Each PID (Primary and Override) goes through the following steps:

- Calculates the difference between the actual value and the setpoint.
- The error signal is processed per the various gain factors (proportional, integral, derivative and normalizing gain) and sent to the selector switch.
- The selector switch selects between the Primary and Override output, depending upon the user setting of Override Type (high or low).
- The selector output (X-out) is sent to the valve.

- The manipulated variable (MV) of the PID whose PID was NOT selected tracks the value of the PID whose MV was selected. This prevents "winding up" of the PID whose output was not selected. It also produces a bump-less transfer to the other PID when that transfer is required.

For the scenario that is being established, the Primary PID is controlling flow rate at 400MCF. The Override PID is watching down stream static pressure and is going to prevent it from going over 1000PSI. Each PID, through the selector switch, has control over the valve position, when appropriate.

As the flow rate drops below 400MCF, the MV of the Primary PID will be re-calculated based on the error signal (flow setpoint–flowrate PV). This new MV is presented to the selector. At the same time, the Override PID will recalculate its MV. If its upper limit (1000PSI) has not been met, it will try to open the valve all the way. If the Override Type has been set to 'low', the selector will select the lower of the two MVs to be output to the valve. Which MV is the lower of the two PIDs will depend upon the error signal and the various gain settings of the PIDs. Assuming that the Override PID is higher than the Primary PID, the Primary PID's MV (being the lower of the two) would continue to open the valve.

When (and if) the down stream pressure limit of 1000PSI is reached, the Override PID would become the controlling PID. Again, both PIDs would recalculate their manipulated variables and present them to the selector. The lower MV (now the Override PID) would be selected to pass onto the valve, and it would begin to close. The Primary MV will track the Override PID until the Override MV becomes un-selected or lower than the Primary MV. At this point, the Primary MV would again control the valve, and the Override MV will track the Primary MV.

4.1.1 Assumptions

It is assumed within this scenario that both the transducer and control valve are installed on the pipe. It is also assumed that the user is using an XFC^{G4}.

The subsequent information will detail the steps necessary to establish this scenario.

4.2 Enabling the AGA-3 and PID Controller Applications

- 1) Move to slot 11 within the Applications tab.
- 2) From the drop-down menu, select AGA-3 Measurement.
- 3) Click the Send button, and then click the Re-read button. AGA3-1 will display in the tree-view under the Flow Measurement main menu.
- 4) Next, move to slot 101 thru 119 within the Applications tab.
- 5) From the drop-down menu, select PID Controller.
- 6) Click the Send button, and then click the Re-read button. PID Control will display in the tree-view under the Flow Measurement main menu.

4.3 AGA-3 Setup

- 1) Within the tree-view, the user can move to the AGA-3 application under the Flow Measurement main menu. Move to the Current tab.
- 2) Within the Current tab, hover the mouse over the Flow Rate amount. A small information box will display that details the register number for the value. Write down the register number. For the sake of these instructions, the register number is 11.7.19. This value will be used for the Primary PID.

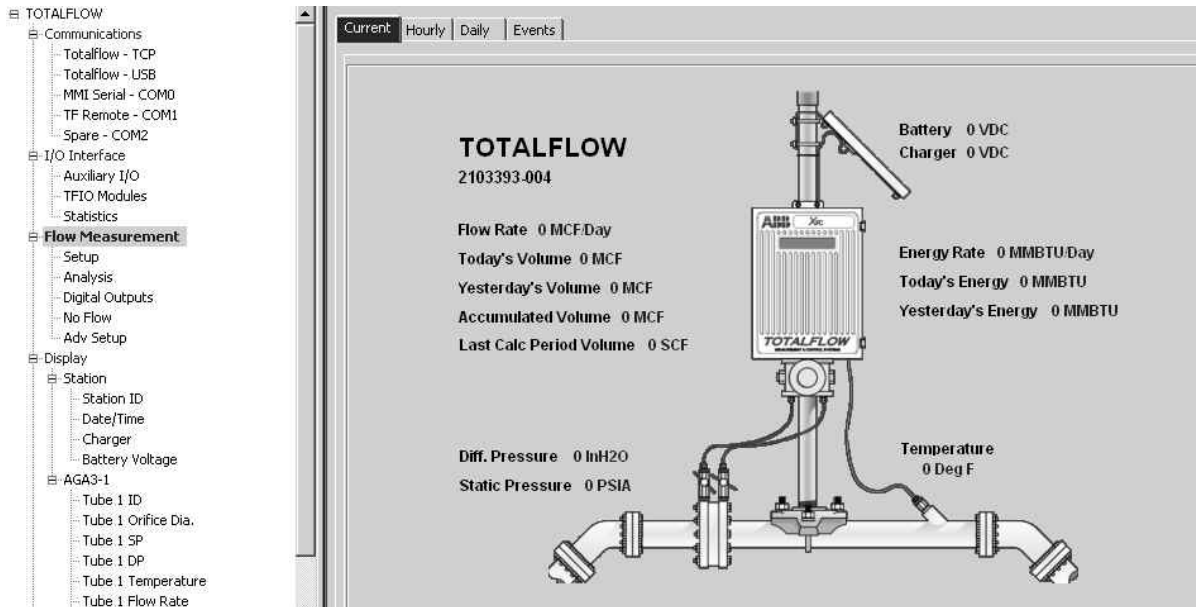


Figure 7 AGA-3 Current Tab

4.4 Calibrating the Analog Outputs for 4-20 mA Range

- 3) From the PCCU main window tool bar, click on the Tools icon. This will take the user to the Calibrate screen.
- 4) From the tree-view, click on the corresponding TFIO module that is to be calibrated.
- 5) Connect a current meter in series with the TFIO module, and load. This can be any one of the four terminals on the TFIO module. Ensure that the terminal selected goes directly to the load.
- 6) Click the Manual Mode radio button.
- 7) Click on the Low button in the Calibration (ma) section of the window. Enter the reading displayed by the current meter. Click OK. The ammeter reading should be slightly less than 4 mA.
- 8) Click on the High button on the Calibration (ma) side of the screen, and enter the reading displayed by the current meter. Click OK. The ammeter reading should be slightly above 20 mA.
- 9) The user will see a dialog box stating that the calibration is complete. Acknowledge the message by clicking the OK button. The Current Values windows (Eng. Units, %FS and mA) should be showing full scale values.

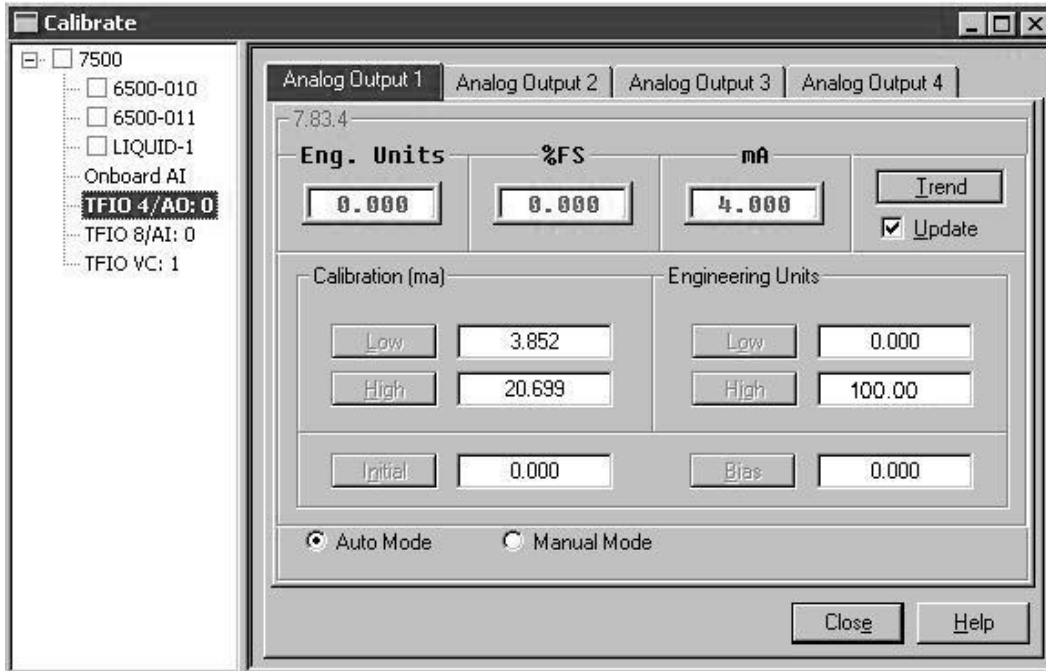


Figure 8 Calibrate Window

4.5 Setting Up the PID Application

4.5.1 Operation Tab

- 1) The user will need to navigate to Entry mode within PCCU. Upon completing this task, click on PID Control within the tree-view. By default, the user is taken to the Operation tab.
- 2) Within the Operation tab, the user will see three scales and an information table with three columns. The first column is a list of each PID variable by name. The second column lists the PID values for the Primary PID. The third column lists the PID values for the Override PID (see).
- 3) In keeping with the established scenario, the user will click into the Setpoint field in the Primary column, and enter a value of 400.
- 4) Next, the user will move to the adjacent Override column, and type in the Setpoint value as 1000.
- 5) Next, move to the Deadband field under the Primary column, and type in .5. Move to the adjacent Override column, and set the Deadband value to 2.0.
- 6) When finished, click the Send button.

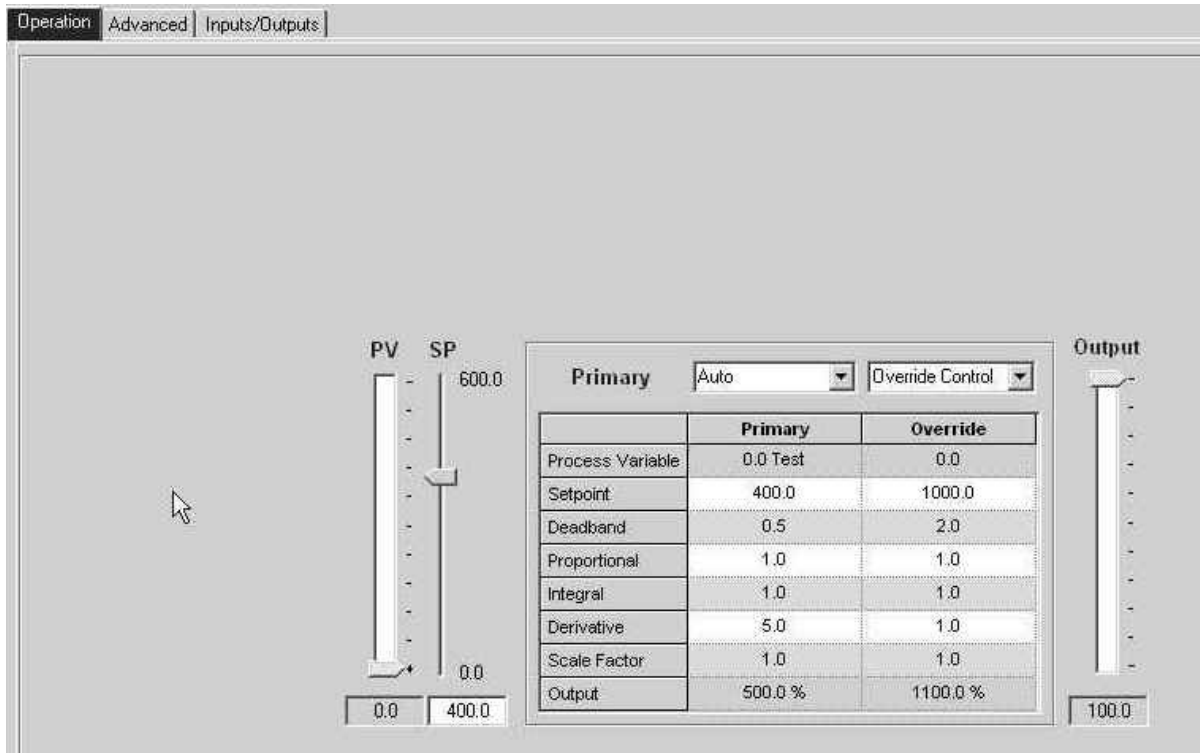


Figure 9 Operation Tab

4.5.2 Advanced Tab

- 1) Click the Advanced tab within the PID Control application. The user will need to set the following parameter fields:
 - a. Primary SP High Limit - 600.0
 - b. Override SP High Limit - 1500.0
 - c. Minimum Control Time - 00:00:10
 - d. Manual Mode Tracking - Enabled
- 2) The remaining parameter fields can stay at their default settings.
- 3) Click the Send button.
- 4) Click the Re-read button. The user can then click on the Operation tab, and verify that the values entered within the Advanced tab have taken effect.

Operation Advanced Inputs/Outputs		
	Description	
14.253.0	Control Loop Description	
14.252.10	Primary SP Low Limit	0.0000
14.252.11	Primary SP High Limit	600.0000
14.252.14	Primary SP Ramp Rate	0.0000
14.252.31	Primary Derivative Filter	1.0000
14.252.32	Primary Dead Band Factor	1.0000
14.252.30	Primary Output Scale Factor	1.0000
14.252.12	Override SP Low Limit	0.0000
14.252.13	Override SP High Limit	1500.0000
14.252.15	Override SP Ramp Rate	0.0000
14.252.34	Override Derivative Filter	1.0000
14.252.35	Override Dead Band Factor	1.0000
14.252.33	Override Output Scale Factor	1.0000
14.252.28	Override Threshold	0.0000
14.250.5	Override Type	Low
14.252.29	Loop Interval	1
14.251.0	Minimum Control Time	00:00:10
14.250.8	Manual Mode Tracking	Enabled
14.250.7	Controller Reset Mode	Manual
14.251.2	Full Closed StepOff	2000
14.251.1	Valve Rest Interval	1000
14.252.18	Digital Output Gain	100.0000

Figure 10 Advanced Tab

4.5.3 Inputs /Outputs Tab

- 1) Click the Inputs/Outputs tab within the PID Control application. The user will need to set the following parameter fields:
 - a. Primary PV - 11.7.19
 - b. Primary PV Units - MCF/Day
 - c. Override PV - 7.3.3
 - d. Output Type - Analog
 - e. Analog Output - 7.81.0
- 2) The remaining parameter fields can stay at their default settings.

- 3) Click the Send button.
- 4) Click the Re-read button. The user can then click on the Operation tab, and verify that the values entered within the Advanced tab have taken effect.

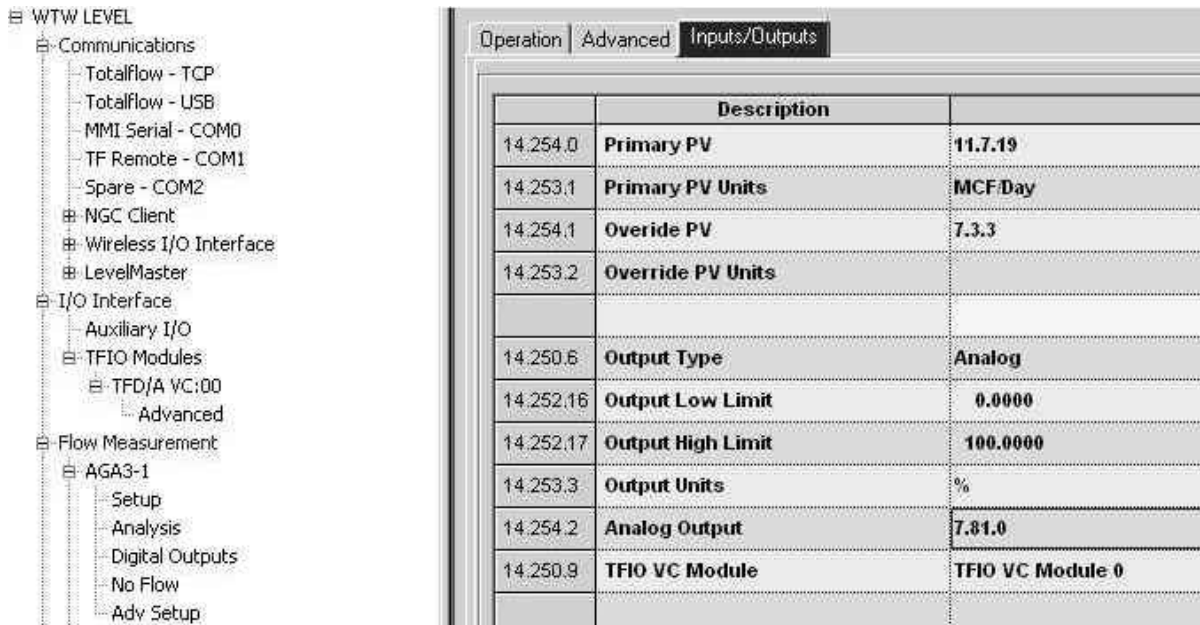


Figure 11 Inputs/Outputs Tab

5.0 CONTROLLER INITIAL VALUES AND SYSTEM TUNING

If the PID controller needs to be more responsive (act more quickly), increase the proportional factor and/or decrease the integral factor. The derivative factor may need to be increased in order to dampen any overshoot or undershoot in the controller output. The actual values selected for these parameters will depend upon the process that is under control.

The following information should also be considered when setting initial values for the PID controller.

5.1 Scale Factor

Scale factor is a mapping between the process value and the output value.

The general equation is: **output/process value = Scale Factor**, where the numbers used are the full scale values of their ranges.

Example: The process variable is flow rate and its measurement is coming from an AGA-3 application. The flow is being controlled by a 4-20ma control valve. The maximum flow rate when the valve is fully open is 800MCF. The output value used to control the 4-20ma valve is 0-100%. Then a good initial scale factor is 100/800 or 0.125.

As experience is acquired with the use of this algorithm, the following equation may be used to provide a better initial value:

$$(\text{Proportional gain} \times \text{output}) / \text{process value} = \text{Scale Factor}$$

5.2 Derivative Filter

This value should never be set to a value less than the loop interval. This parameter acts as a derivative amplifier when the value is less than the loop interval. This will cause severe oscillations in the output value. A value equal to the loop interval provides the means of disabling the derivative filter.

This parameter is used to reduce the noise reading of the derivative factor. Noisy process readings can cause the controller derivative factor to cause “chattering” of the control process (i.e. the valve). Process variables such as flow rate or flow volume will probably not benefit from using the derivative filter, but pressure or temperature used for the process variable reading will benefit from the use of the derivative filter.

5.3 Integral Factor

The Integral factor is primarily thought of as the ‘output to input’ phase changing parameter. A value of less than 1.0 produces preset action, or anticipatory action. Where as, an integral factor value greater than 1.0 produces a delayed action. The default value of 1.0 provides no phase change and a value of 0.0 provides zero effect in the output value.

5.4 Derivative Factor

The derivative factor is primarily the emergency operation when the process variable starts to deviate from the set point. This parameter is responsible for reducing or eliminating the overshoot or undershoot of the set point when the set point is changed, or some disturbance takes the process variable away from its controlled position. It is important that the derivative factor, as it is applied to the process variable, is not fooled by noise. The response of the control valve is the primary concern for the value of the derivative factor (how much output is needed to get the valve moving in the right direction). A highly filtered process value (using the derivative filter) will need a much larger derivative factor to produce the desired effect. (i.e. a derivative filter value of 8 will need a derivative factor multiplied by 2 to restore the losses of the filter).

6.0 HOLDING REGISTERS

The following information will describe the Totalflow registers for the G4 PID/Operations/Holding applications.

The Totalflow G4 XFC devices provide access to application data by numeric tags. These numeric tags consist of three values: application number, array number and register number. These three values uniquely identify any accessible application data item. These values are commonly referred to as "app.array.register".

It should be noted the following format for the register list is array#.register#. The application can be instantiated in any application slot.

Register	Description	Type	Access	Note
0.X	Periodic Operations X Trigger Register	Register	Read/Write	1
1.X	Periodic Operations X Trigger Type	Byte	Read/Write	1
2.X	Periodic Operations X Trigger Interval	Uint32	Read/Write	1
3.X	Periodic Operations X Operation	Byte	Read/Write	2
4.X	Periodic Operations X Register 1	Register	Read/Write	2
5.X	Periodic Operations X Register 2	Register	Read/Write	2
6.X	Periodic Operations X Output Register	Register	Read/Write	2
7.X	Math Operation X Result	Float	Read Only	3
8.X	Math Operation X Operation	Byte	Read/Write	
9.X	Math Operation X Register 1	Register	Read/Write	
10.X	Math Operation X Register 2	Register	Read/Write	
11.X	Bit Operation X Result	Uint32	Read Only	4
12.X	Bit Operation X Operation	Byte	Read/Write	
13.X	Bit Operation X Register 1	Register	Read/Write	
14.X	Bit Operation X Register 2	Register	Read/Write	

15.X	Compare Operation X Result	Float		6
16.X	Compare Operation X Operation	Uint16		
17.X	Compare Operation X Register 1	Register		
18.X	Compare Operation X Register 2	Uint16		
19.X	Array Operation X Result	Float	Read Only	6
20.X	Array Operation X Operation	Uint16	Read/Write	
21.X	Array Operation X Register	Register	Read/Write	
22.X	Array Operation X Size	Uint16	Read/Write	
23.X	Periodic Operation X Descriptor	String	Read/Write	
24.X	Math Operation X Descriptor	String	Read/Write	
25.X	Bit Operation X Descriptor	String	Read/Write	
26.X	Compare Operation X Descriptor	String	Read/Write	
27.X	Array Operation X Descriptor	String	Read/Write	
28.X	Select Operation X Descriptor	String	Read/Write	7
29.X	Select Operation X Switch	Register	Read/Write	
30.X	Select Operation X Input 1	Register	Read/Write	
31.X	Select Operation X Input 2	Register	Read/Write	
32.X	Select Operation X Output	Float	Read/Write	
33.X	Lag Operation X Description	String	Read/Write	8
34.X	Lag Operation X Interval	Uint32	Read/Write	
35.X	Lag Operation X Input Register	Register	Read/Write	
36.X	Lag Operation X Output	Float	Read/Write	

37.X	Lead Operation X Description	String	Read/Write	9
38.X	Lead Operation X Interval	Uint32	Read/Write	
39.X	Lead Operation X Input Register	Register	Read/Write	
40.X	Lead Operation X Output	Float	Read/Write	
41.X	Queue Operation X Descriptor	String	Read/Write	10
42.X	Queue Operation X Interval	Uint32	Read/Write	
43.X	Queue Operation X Array Register	Register	Read/Write	
44.X	Queue Operation X Array Size	Uint16	Read/Write	
45.X	Queue Operation X Input Register	Register	Read/Write	
46.X	Queue Operation X Output	Float	Read/Write	
47.X	Ramp Operation X Descriptor	String	Read/Write	11
48.X	Ramp Operation X Slope	Float	Read/Write	
49.X	Ramp Operation X Input Register	Register	Read/Write	
50.X	Ramp Operation X Output	Float	Read/Write	
51.X	Pulse Operation X Descriptor	String	Read/Write	12
52.X	Pulse Operation X Interval	Uint32	Read/Write	
53.X	Pulse Operation X Trigger Register	Register	Read/Write	
54.X	Pulse Operation X Trigger Flags	Byte	Read/Write	
55.X	Pulse Operation X Output	Float	Read/Write	
56.X	Limit Operation X Descriptor	String	Read/Write	13
57.X	Limit Operation X High Limit	Float	Read/Write	
58.X	Limit Operation X Low Limit	Float	Read/Write	

59.X	Limit Operation X Velocity	Float	Read/Write	
60.X	Limit Operation X Input Register	Register	Read/Write	
61.X	Limit Operation X Output	Float	Read/Write	
62.X	Scale Operation X Descriptor	String	Read/Write	14
63.X	Scale Operation X Input High Limit	Float	Read/Write	
64.X	Scale Operation X Input Low Limit	Float	Read/Write	
65.X	Scale Operation X Output High Limit	Float	Read/Write	
66.X	Scale Operation X Output Low Limit	Float	Read/Write	
67.X	Scale Operation X Input Register	Register	Read/Write	
68.X	Scale Operation X Output	Float	Read/Write	
69.X	PID Operation X Descriptor	String	Read/Write	15
70.X	PID Operation X Input Register	Register	Read/Write	
71.X	PID Operation X Setpoint	Float	Read/Write	
72.X	PID Operation X Dead Band	Float	Read/Write	
73.X	PID Operation X Proportional Gain	Float	Read/Write	
74.X	PID Operation X Integral Reset	Float	Read/Write	
75.X	PID Operation X Derivative Rate	Float	Read/Write	
76.X	PID Operation X Mode	Uchar	Read/Write	
77.X	PID Operation X Action	Uchar	Read/Write	
78.X	PID Operation X Output	Float	Read/Write	
79.X	Playback Operation X Descriptor	String	Read/Write	16
80.XYY	Playback Operation X Variable	Float	String	

	Y			
81.X	Playback Operation X Filename	String	Read/Write	
82.X	Playback Operation X Interval	Uint32	Read/Write	
83.X	Playback Operation X Size	Uint8	Read/Write	
84.X	Playback Operation X Trigger Register	Register	Read/Write	
85.X	Playback Operation X Trigger Type	Uint8	Read/Write	
86.X	Playback Operation X Wrap Option	Uint8	Read/Write	
87.X	Playback Operation X Pause	Uint8	Read/Write	
88.X	Playback Operation X Input Line #	Uint32	Read	
89.X	Playback Operation X File Error Code	Uint32	Read	
90.XYY	Playback Operation X Variable Y Name	String	String	
250.0	PID status (Primary/Override)	Byte	Read/Write	
250.1	PID Type	Byte	Read/Write	17
250.2	PID Mode (Auto/Manual)	Byte	Read/Write	
250.3	PID Action 1 (Direct/Reverse)	Byte	Read/Write	
250.4	PID Action 2 (Direct/Reverse)	Byte	Read/Write	
250.5	PID Override Type (High/Low)	Byte	Read/Write	
250.6	PID Output Type (Analog/Digital)	Byte	Read/Write	
250.7	PID Reset (Auto/Manual)	Byte	Read/Write	
250.8	PID Track (Enable/Disable)	Byte	Read/Write	
250.9	PID TFIO Valve Control Module	Byte	Read/Write	
251.0	PID Hysteresis	Uint32	Read/Write	
251.1	PID Valve Rest Interval	Uint32	Read/Write	
251.2	PID Full Close Step Off	Uint32	Read/Write	

251.3	PID VC Module Timer	Uint32	Read/Write	
252.0	PID Primary Set Point	Float	Read/Write	
252.1	Primary Dead Band	Float	Read/Write	
252.2	Primary Proportional Gain	Float	Read/Write	
252.3	Primary Integral Reset	Float	Read/Write	
252.4	Primary Derivative Rate	Float	Read/Write	
252.5	Override Set Point	Float	Read/Write	
252.6	Override Dead Band	Float	Read/Write	
252.7	Override Proportional Gain	Float	Read/Write	
252.8	Override Integral Reset	Float	Read/Write	
252.9	Override Derivative Rate	Float	Read/Write	
252.10	Primary SP Low Limit	Float	Read/Write	
252.11	Primary SP High Limit	Float	Read/Write	
252.12	Override SP Low Limit	Float	Read/Write	
252.13	Override SP High Limit	Float	Read/Write	
252.14	Primary SP Ramp Rate	Float	Read/Write	
252.15	Override SP Ramp Rate	Float	Read/Write	
252.16	Output Low Limit	Float	Read/Write	
252.17	Output High Limit	Float	Read/Write	
252.18	Digital Output Gain	Float	Read/Write	
252.19	Primary Controller PV	Float	Read/Write	
252.20	Override Controller PV	Float	Read/Write	
252.21	Primary Controller Output	Float	Read/Write	
252.22	Override Controller Output	Float	Read/Write	
252.23	Controller Position	Float	Read/Write	
252.24	Controller Change	Float	Read/Write	
252.25	Previous Output	Float	Read/Write	
252.26	Ramped Primary SP	Float	Read/Write	
252.27	Ramped Override SP	Float	Read/Write	

252.28	Override Threshold	Float	Read/Write	
252.29	Execution interval seconds units	Float	Read/Write	
252.30	PID 1 Normalization Scale	Float	Read/Write	
252.31	PID 1 Derivative filter seconds units	Float	Read/Write	
252.32	PID 1 Normalization Scale factor inside dead band	Float	Read/Write	
252.33	PID 2 Normalization Scale factor	Float	Read/Write	
252.34	PID 2 Derivative filter seconds units	Float	Read/Write	
252.35	PID 2 Normalization Scale factor inside dead band	Float	Read/Write	
253.0	PID Controller Description	String	Read/Write	
253.1	PID Primary Units	String	Read/Write	
253.2	PID Override Units	String	Read/Write	
253.3	PID Output Units	String	Read/Write	
254.X	PID Primary PV	Register	Read/Write	
254.X	PID Override PV	Register	Read/Write	
254.X	PID Analog Output	Register	Read/Write	
254.X	PID External Controller	Register	Read/Write	
255.0	Number of Periodic Operations	Uint16	Read/Write	
255.1	Number of Math Operations	Uint16	Read/Write	
255.2	Number of Bit Operations	Uint16	Read/Write	
255.3	Number of Compare Operations	Uint16	Read/Write	
255.4	Number of Array Operations	Uint16	Read/Write	
255.10	Number of Select Operations	Uint16	Read/Write	

255.12	Number of Lag Operations	Uint16	Read/Write	
255.14	Number of Lead Operations	Uint16	Read/Write	
255.16	Number of Queue Operations	Uint16	Read/Write	
255.18	Number of Ramp Operations	Uint16	Read/Write	
255.20	Number of Pulse Operations	Uint16	Read/Write	
255.22	Number of Limit Operations	Uint16	Read/Write	
255.24	Number of Scale Operations	Uint16	Read/Write	
255.26	Number of PID Operations	Uint16	Read/Write	
255.28	Number of Playback Operations	Uint16	Read/Write	
20X.Y	Holding Array X Register Y Value	Programmable	Read/Write	101
22X.Y	Holding Array X Register Y Description	String	Read/Write	102
241.X	Holding Array X Description	String	Read/Write	103
242.X	Holding Array X Data Type	Byte	Read/Write	104
243.X	Holding Array X Persistence	Byte	Read/Write	104
244.X	Holding Array X Capacity	Uint16	Read/Write	105
244.255	Number of Holding Arrays	Uint16	Read/Write	106

6.1 Application Notes

- 5) Period operations can be executed at a cyclic interval (such as every five seconds), at a specific time of day (such as 8:00) or they can be triggered by another register. If the operation trigger type in array 1 is 0 (Interval), the operation is executed at the interval specified by the corresponding register in array 2. If the operation trigger type is 1 (Time), the operation is executed at the time of day specified by the corresponding register in array 2. If the operation trigger type is 2 (Triggered), the operation is executed each second if the register specified by the corresponding register in array 0 is non-zero.
- 6) Periodic operations combine the capabilities of Math, Bit, Compare and Array operations. The registers in array 3 contain operation codes specifying the operation to be performed. The registers in arrays 4 and 5 contain the operands of the operations. The registers in array 6 specify where the results of the operations will be placed. The following operation codes are supported:
 - 0 - No Operation
 - 1 - R1 + R2 (Addition)

```

2 - R1 - R2 (Subtraction)
3 - R1 * R2 (Multiplication)
4 - R1 / R2 (Division)
5 - R1 ^ R2 (Power of)
6 - sqrt(R1) (Square Root)
7 - sin(R1) (Trig Sine)
8 - cos(R1) (Trig Cosine)
9 - tan(R1) (Trig Tangent)
10 - ABS(R1) (Remove sign)
11 - Log(R1) (Logarithm)
12 - 1 / R1 (Reciprocal)
13 - R1 % R2 (Modulo)
20 - R1 & R2 (Bitwise AND)
21 - R1 | R2 (Bitwise OR)
22 - R1 ^ R2 (Bitwise XOR)
23 - R1 ~ R2 (Bitwise Complement)
24 - R1 << R2 (Bitwise Shift Left)
25 - R1 >> R2 (Bitwise Shift Right)
26 - R1 = !R2 (Bitwise NOT)
30 - (R1 & R2) == 0 (True if Equal to Zero)
31 - (R1 & R2) != 0 (True if Not Equal to Zero)
32 - (R1 | R2) == 0 (True if Equal to Zero)
33 - (R1 | R2) != 0 (True if Not Equal to Zero)
34 - (R1 ^ R2) == 0 (True if Equal to Zero)
35 - (R1 ^ R2) != 0 (True if Not Equal to Zero)
40 - R1 == R2 (True if Equal)
41 - R1 != R2 (True if Not Equal)
42 - R1 > R2 (True if Greater than)
43 - R1 >= R2 (True if Greater than or equal to)
44 - R1 < R2 (True if Less than)
45 - R1 <= R2 (True if Less than or equal to)
46 - R1 == 0 (True if Equal to zero)
47 - R1 != 0 (True if Not Equal to zero)
48 - R1 > 0 (True if Greater than zero)
49 - R1 < 0 (True if Less than zero)
60 - Integer R1 + R2 (Addition)
61 - Integer R1 - R2 (Subtraction)
62 - Integer R1 * R2 (Multiplication)
63 - Integer R1 / R2 (Division)
64 - Integer abs(R1) (Remove sign)
65 - Integer R1 % R2 (Modulo)
70 - Array Addition
71 - Array Linear Average
72 - Array Square Root Average
73 - Array Minimum
74 - Array Maximum
80 - Copy Array
81 - Move Array
254 - Out = R2 = R1
255 - Out = R1

```

- 7) Math operations are executed when the corresponding register in array 7 is read. The registers in array 8 contain operation codes specifying the math operation to be performed. The registers in arrays 9 and 10 contain the operands of the operations. The result of the math operation is returned in the corresponding register in array 7. The following math operation codes are supported:

- 0 - No Operation
- 1 - $R1 + R2$ (Addition)
- 2 - $R1 - R2$ (Subtraction)
- 3 - $R1 * R2$ (Multiplication)
- 4 - $R1 / R2$ (Division)
- 5 - $R1 ^ R2$ (Power of)
- 6 - $\text{sqrt}(R1)$ (Square Root)
- 7 - $\text{sin}(R1)$ (Trig Sine)
- 8 - $\text{cos}(R1)$ (Trig Cosine)
- 9 - $\text{tan}(R1)$ (Trig Tangent)
- 10 - $\text{ABS}(R1)$ (Remove sign)
- 11 - $\text{Log}(R1)$ (Logarithm)
- 12 - $1 / R1$ (Reciprocal)
- 13 - $R1 \% R2$ (Modulo)

- 8) Bit operations are executed when the corresponding register in array 11 is read. The registers in array 12 contain operation codes specifying the bit operation to be performed. The registers in arrays 13 and 14 contain the operands of the operations. The result of the bit operation is returned in the corresponding register in array 11. The following bit operation codes are supported:

- 20 - $R1 \& R2$ (Bitwise AND)
- 21 - $R1 | R2$ (Bitwise OR)
- 22 - $R1 \wedge R2$ (Bitwise XOR)
- 23 - $R1 \sim R2$ (Bitwise Complement)
- 24 - $R1 \ll R2$ (Bitwise Shift Left)
- 25 - $R1 \gg R2$ (Bitwise Shift Right)
- 26 - $R1 = !R2$ (Bitwise NOT)
- 30 - $(R1 \& R2) == 0$ (True if Equal to Zero)
- 31 - $(R1 \& R2) != 0$ (True if Not Equal to Zero)
- 32 - $(R1 | R2) == 0$ (True if Equal to Zero)
- 33 - $(R1 | R2) != 0$ (True if Not Equal to Zero)
- 34 - $(R1 \wedge R2) == 0$ (True if Equal to Zero)
- 35 - $(R1 \wedge R2) != 0$ (True if Not Equal to Zero)

- 9) Compare operations are executed when the corresponding register in array 15 is read. The registers in array 16 contain operation codes specifying the compare operation to be performed. The registers in arrays 17 and 18 contain the operands of the operations. The result of the compare operation is returned in the corresponding register in array 15. The following compare operation codes are supported:

- 40 - $R1 == R2$ (True if Equal)
- 41 - $R1 != R2$ (True if Not Equal)
- 42 - $R1 > R2$ (True if Greater than)
- 43 - $R1 >= R2$ (True if Greater than or equal to)
- 44 - $R1 < R2$ (True if Less than)
- 45 - $R1 <= R2$ (True if Less than or equal to)
- 46 - $R1 == 0$ (True if Equal to zero)
- 47 - $R1 != 0$ (True if Not Equal to zero)
- 48 - $R1 > 0$ (True if Greater than zero)
- 49 - $R1 < 0$ (True if Less than zero)

- 10) Array operations are executed when the corresponding register in array 19 is read. The registers in array 20 contain operation codes specifying the array operation to be performed. The registers in arrays 21 and 22 contain the

operands of the array operations. The result of the operation is returned in the registers in array 19. The following operation codes are supported:

- 70 - Array Addition
- 71 - Array Linear Average
- 72 - Array Square Root Average
- 73 - Array Minimum
- 74 - Array Maximum
- 80 - Copy Array
- 81 - Move Array

- 11)** Select operations are executed when the corresponding register in array 32 is read. The registers in array 29 contain the value that is used to switch the output between two input values. The registers in arrays 30 and 31 contain the inputs to the operation. The result of the select operation is returned in the corresponding register in array 32.
- 12)** Lag operations are executed when the corresponding register in array 36 is read. The registers in array 34 contain the value that is used as the lag interval. The register in array 35 contains the input to the operation. The result of the lag operation is returned in the corresponding register in array 36.
- 13)** Lead operations are executed when the corresponding register in array 40 is read. The registers in array 38 contain the value that is used as the lead interval. The register in array 39 contains the input to the operation. The result of the lead operation is returned in the corresponding register in array 40.
- 14)** Queue operations are executed when the corresponding register in array 46 is read. The registers in array 42 contain the value that is used as the queue interval. The register in array 45 contains the input to the operation. The register in array 43 specifies the starting register address for the array. The register in array 44 specifies the number of registers in the array. The result of the queue operation is returned in the corresponding register in array 46.
- 15)** Ramp operations are executed when the corresponding register in array 50 is read. The registers in array 48 contain the value that is used as the slope. The register in array 39 contains the input to the operation. The result of the ramp operation is returned in the corresponding register in array 50.
- 16)** Pulse operations are executed when the corresponding register in array 55 is read. The registers in array 52 contain the value that is used as the pulse interval. The register in array 53 contains the trigger input. The result of the pulse operation is returned in the corresponding register in array 55.
- 17)** Limit operations are executed when the corresponding register in array 61 is read. The registers in array 60 contain the input to the operation. The registers in array 57 are used as the high limit values for the input data. The registers in array 58 are used as the low limit values for the input data. The registers in array 59 specify the value in engineering units per second that the output is allowed to change in response to the input. The result of the limit operation is returned in the corresponding register in array 61.
- 18)** Scale operations are executed when the corresponding register in array 68 is read. The registers in array 67 contain the input to the operation. The registers in array 63 are used as the high limit values for the input data. The registers in array 64 are used as the low limit values for the input data. The registers in array 65 are used as the high limit values for the output data. The registers in array 66 are used as the low limit values for the output data. The result of the scale operation is returned in the corresponding register in array 68.

- 19) PID operations are executed when the corresponding register in array 78 is read. The registers in array 70 contain the input to the operation. The registers in array 71 are used as the set points for the operation. The registers in array 72 are used as the dead band values for the set points and the inputs. The registers in array 73 are used as the proportional gain of the operation. The registers in array 74 are used as the integral reset values of the operation. The registers in array 75 are used as the derivative rate of the operation. The registers in array 76 are used for the auto (1) / manual (0) mode of the operation. The registers in array 77 are used for the action (direct =0, reverse=1) of the operation. The result of the PID operation is returned in the corresponding register in array 78.
- 20) Playback operations can be executed at a cyclic interval specified in array 82 and Pause (array 87) is non-zero. Pause is set by the Trigger Type in array 85. When the end-of-file is reached, the Wrap option, array 86, controls the next action. A non-zero will cause starting over from the start of the file; else the values are left unchanged (Pause). The size parameter, array 83, sets the number of columns of data read from the file. Position, array 88, show the line number of the last read values. File Error, array 89, will show if there is an error in the File open. The value of 2 indicates: "no such file exists". The Output values in array 80 are read from the filename, in array 81, which is a CSV (comma separated values) file. The first line contains the names of the columns and is placed in the variable descriptions, array 90. Array 79 contains the name of the playback block.

21) PID Type

- 0 – Primary
- 1 – Override
- 2 – Primary & Override

Sample File:

```
#Tubing,Casing,Case_Tube,Line,Tube_Line,Flowrate
170.574000,176.892000,6.318000,128.626000,41.948000,0.00000
```

Trigger Type:

0	Level Control	Pause on Non-Zero
1	Level w/ Rewind	Pause on non-Zero, Reset Position to Zero on Zero
2	Rising Edge	Playback on changing to Non-Zero
3	Falling Edge	Playback on changing to Zero
4	Change of State	Playback on changing value

Wrap – End of File action

0	Freeze	Keep returning last value read from file.
1	Wrap	Restart reading at beginning of file.

Pause

0	Run	Read file on interval
1	Pause	Stop reading file
2	STOP	Pause and close file

- 100) The number, type and size of the Holding Register arrays are user-programmable.
- 101) 255.x registers reserved for G3 compatibility (old descriptor enable):
5-9,11,13,15,17,19,32,23,25,27
- 102) Each Holding Register Array can be assigned a 24-character name.
- 103) The data type of each Holding Register array is user-programmable. The following data types are supported:

0	32-Bit Floating Point
1	16-Bit Integer
2	16-Bit Unsigned Integer
3	32-Bit Integer
4	32-Bit Unsigned Integer
5	8 Bit Byte
6	24-Character String
7	4 Byte Register (app.array.register)

- 104) Each Holding Register array can be made persistent through a warm start. If the array persistence flag is set to 1, each Write operation to a register in the array causes the array data to be written to file. This can be very slow. It will also increase processor usage considerably. If registers in an array are being updated frequently, the array should not be made persistent.
- 105) The number of registers in each Holding Register array is limited to 65,535 and by available memory.
- 106) The number of holding arrays is limited to 20 and by available memory.



©Copyright 2011 ABB, All rights reserved

Document Title

Application Information, (PID Control)

Document No.	Date & Rev. Ind.	No. of Pages	
2104301-001	AB	34	