

Chapter 19 Programming the PID Algorithm

Introduction

The PID algorithm is used to control an analog process having a single control point and a single feedback signal. The PID algorithm controls the output to the control point so that a setpoint is achieved. The setpoint may be entered as a static variable or as a dynamic variable that is calculated from a mathematical operation.

For many years, the PID algorithm was not accepted as a function suitable for a PLC. It was included in a DCS (Distributed Control System) or configured from a number of stand-alone PID controllers. However, as PLC prices continued to fall during the 1980's and later and more economical HMI systems were developed for the PLC, PLCs became more accepted as PID controllers. In fact, because PLCs have undercut the cost of competing systems, DCSs and other PID controllers have been forced to drop prices dramatically or no longer remain competitive. An early hybrid design was introduced into the Allen-Bradley 1771 I/O family including 2 PID stand-alone controllers attached to a single I/O slot and executing the PID algorithm from the controller in the I/O slot. Newer control schemes have the PID algorithm executing in the PLC with other programs and controlling complicated processes with good success.

Chapter 19 uses the PID block to control a simple process. Then, it discusses more complex operations capable of being programmed by the PID control block. The chapter describes the SLC PID block followed by the CompactLogix processor as well as the Siemens 1200 and their implementations of the PID function. Using these various PLC configurations demonstrates differences between the newer PID blocks and the SLC PID block. The SLC processor uses an integer-based PID block. Integer-based blocks have the disadvantage that scaling must be used to convert numbers to more meaningful real values. Scaling adds complexity to the program that becomes transparent with a floating-point PID block. More sophisticated PID blocks such as is available in the PLC/5 and ControlLogix processors as well as Siemens allow floating-point calculations. These more robust PID blocks also provide more sophistication in their functionality. All PID blocks are not created equal.

Fundamentals of Closed Loop Control

Closed Loop Control Tasks

"Closed loop control is a process where the value of a variable is established and maintained continuously through intervention based on measurements of this variable. This generates a sequence of effects that takes place in a closed loop -the control loop- because the process runs based on measurements of a variable that is influenced in turn by itself." This variable that is to be controlled is measured continuously and compared with another specified variable of the same type. Depending on the result of this comparison, an adaptation of the variable to be controlled to the value of the specified variable is performed by the control process.

Proportional Controller (P-Controller)

In the case of P-controllers, the manipulated variable is always proportional to the recorded system deviation. The result is that a P-controller reacts without a delay to a deviation and generates a manipulated variable only if the deviation (error) is present. The proportional pressure regulator sketched in the figure below compares the power F_S of the setpoint spring with

the power F_B that the pressure P_2 generates in the spring-elastic metal bellows. If the forces are off balance, the lever rotates around the pivot point D. The valve position changes and accordingly the pressure P_2 to be regulated until a new balance of forces is established.

The behavior of the P-controller if a system deviation suddenly occurs is shown in the figure below. The amplitude of the manipulated variable jump y depends on the level of the deviation e and the amount of the proportional coefficient K_p :

To keep the deviation low, a proportionality factor as large as possible has to be selected. Increasing the factor causes the controller to respond faster. However, a value that is too high may cause overshooting and a large hunting tendency on the part of the controller.

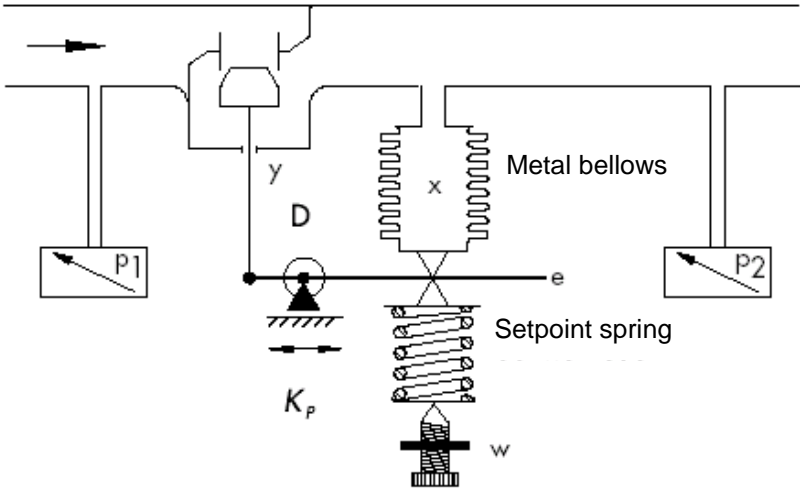


Fig. 19-1

$$Actual\ Flow \approx \sqrt{P_2 - P_1}$$

$$e(error) = Flow (Actual) - Flow (Set Point)$$

$$y(output) = K_p \cdot e$$

The diagram below shows the behavior of the P-controller:

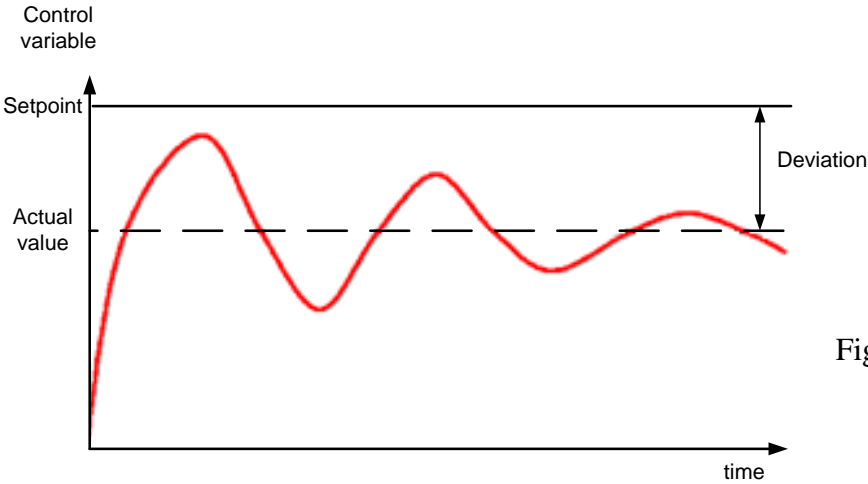


Fig. 19-2

The advantages of this type of controller consist on the one hand of its simplicity (electronic implementation can, in the simplest case, consist of merely a resistor), and on the other hand its prompt response in comparison to other controller types. The main disadvantage of the P-controller is the continuous deviation; the setpoint is never completely attained, even long term. This disadvantage as well as the not yet ideal response speed can be minimized only insufficiently with a larger proportionality factor, since otherwise the controller will overshoot. In the most unfavorable case, the controller will enter a state of continuous oscillation. This causes the controlled variable to be periodically moved away from the setpoint, not by the influencing variable but by the controller.

The problem of continuous deviation is solved best with an integral controller.

Integral Controller (I-Controller)

Integrating controllers are used to completely correct system deviations at each operating point. As long as the deviation is unequal to zero, the manipulated variable continues to change. Only when the reference variable and the controlled variable are equal is the control system in a steady state.

The mathematical formulation of this integral behavior is as follows:

$$y = K_i \int (e) \text{ with } K_i = \frac{1}{T_n}$$

How fast the manipulated variable rises (or falls) depends on the deviation and the integration time.

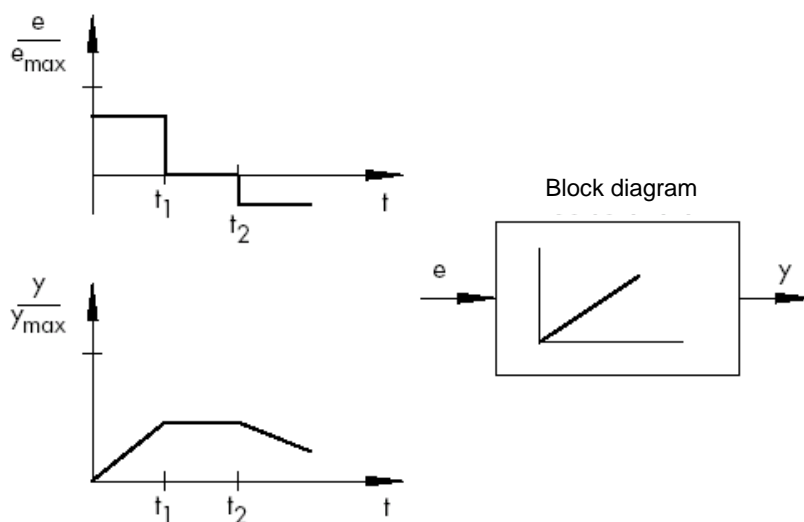


Fig. 19-3

PI-Controller

The PI-controller is a type often used in practice. It results from connecting a P-controller and an I-controller in parallel. When laid out correctly it unites the advantages of both controller types (stable and fast, no permanent system deviation), so that their disadvantages are compensated at the same time.

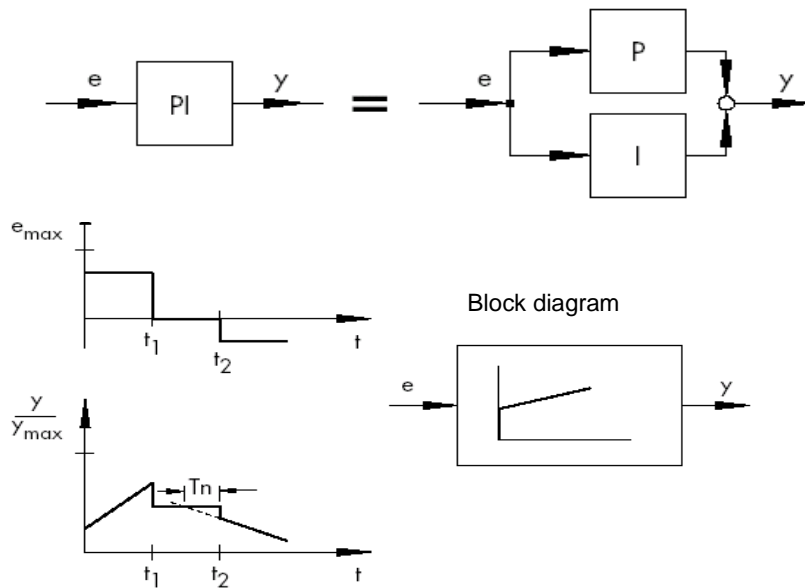


Fig. 19-4

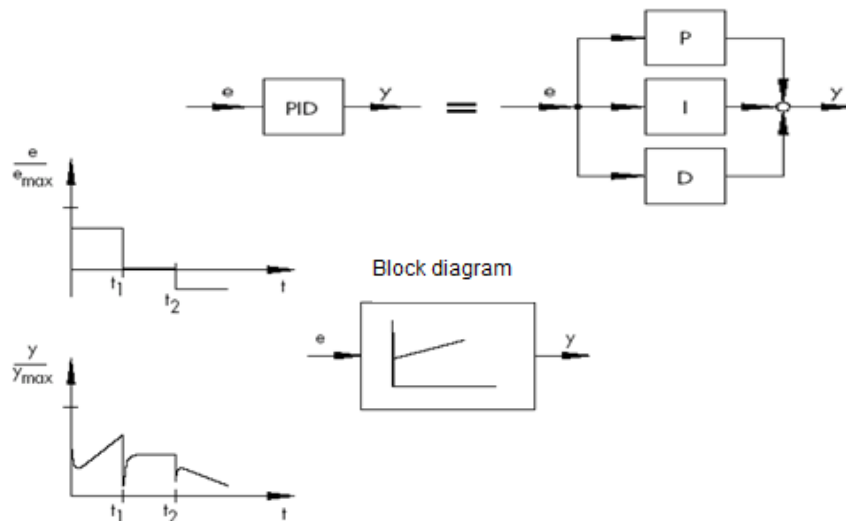
The behavior with respect to time is identified by the proportional coefficient K_p and the reset time T_n . Because of the proportional component, the manipulated variable responds immediately to every system deviation e , while the integral component takes effect only in the course of time. T_n represents the time that passes until the I-component generates the same amplitude of flow as occurs immediately because of the P-component (K_p). As in the case of the I-controller, the reset time T_n has to be decreased if we want to increase the integral component.

Differential Controller (D-Controller)

The D-controller generates its manipulated variable from the rate of change of the system deviation, and not, as the P-controller, from its amplitude. For that reason, it responds considerably faster than the P-controller. Even if the deviation is small, it generates (looking ahead) large amplitudes of flow as soon as an amplitude change occurs. However, the D-controller does not detect permanent deviations, because no matter how large it is, its rate of change equals zero. For that reason, the D-controller is used only rarely by itself in practice. Rather, it is used jointly with other control elements, usually in connection with a proportional component.

PID Controller

If we expand the PI controller with a D-component, the universal PID controller is created. As in the case of the PD controller, adding the D-component has the effect that, if laid out correctly, the controlled variable reaches its setpoint sooner and its steady state faster.



$$y = K_p \cdot e + K_i \int e \cdot dt + K_D \frac{de}{dt} \quad \text{with} \quad K_i = \frac{K_p}{T_n}, K_D = K_p \cdot T_v$$

Fig. 19-5 PID Diagrams and Equations

Objectives of Control System Setting

For the control result to be satisfactory, selecting a suitable controller is an important aspect. However, even more important is setting the suitable controller parameters K_p , T_n and T_v , that have to be adjusted to the controlled system behavior. Usually, we have to compromise between a very stable but slow control system or a very dynamic, more unsettled control performance which under certain circumstances has a tendency to oscillate and can become unstable.

In the case of non-linear systems that are always to process at the same operating point -such as fixed setpoint control- the controller parameters have to be adjusted to the controlled system behavior at this working point. If, as in the case of servo controls, a fixed working point cannot be defined, a controller setting has to be found that supplies a sufficiently fast and stable control result over the entire working range.

In practice, controllers are usually set based on values arrived at through experience. If these are not available, the controlled system behavior has to be analyzed exactly, in order to subsequently -with the aid of theoretical or practical layout procedures - specify suitable controller parameters.

An Example SLC PID Function

In its simplest form, the SLC PID block is used as a single block with no input contacts and surrounded by only two SCP blocks. This PID instruction is located in Ladder 2. The SCP block is configured to retrieve a numerical value from the analog input channel, linearly scale the input and move the resultant value to the PID block. The input is a 4-20 mA signal from a flow transmitter. The output is a 4-20 mA signal to a variable flow valve.

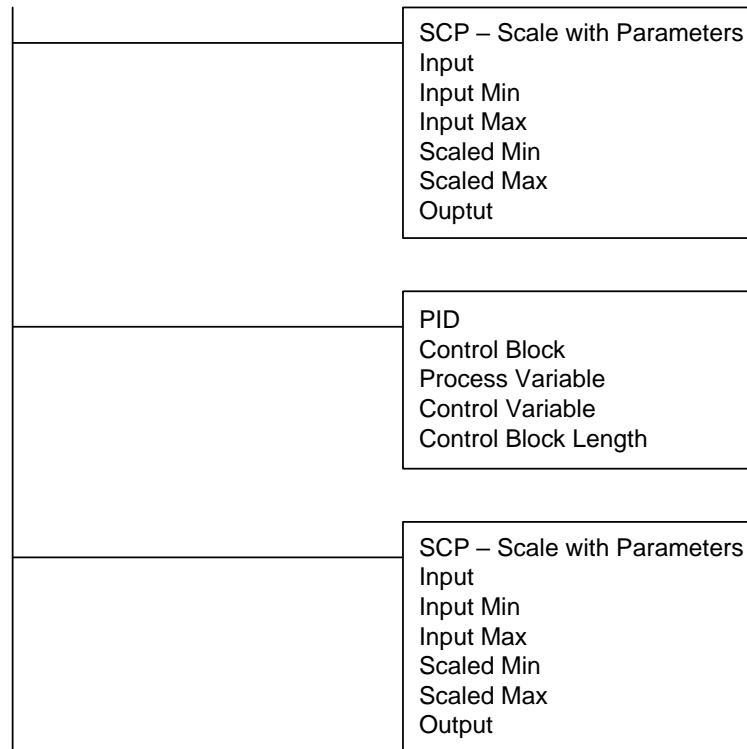


Fig. 19-6 Simple Program of PID for SLC Processor

In the first SCP instruction, values found in the Input Min and Input Max of the SCP instruction are from the I/O card. The engineer must first decide which I/O card to use and then find the proper lower and upper limits from the literature on the card to enter values in the SCP instruction.

In this case, the analog card selected is the 1746-NIO4I Ser. A. This card is a combination card with 2 analog inputs and 2 analog outputs. From the web, select I/O Analog Modules, Analog I/O Modules for SLC 500 Programmable Controllers – Technical Data. Then select 4 Channel Module Configuration, 4 Channel Module Wiring, and 4 Channel Module Specifications to find the choices available for Analog Inputs and Analog Outputs.

In the section describing 4 Channel Module Specifications are found the following Channel Data sheets:

Input Type	Signal Range	Engineering Units	EU Scale
+/- 10 Vdc	-10.25 to + 10.25 Vdc	-10250 to + 10250	1 mV/step
0 to 5V dc	-0.5 to +5.5 Vdc	-500 to +5500	1 mV/step
1 to 5V dc	0.5 to 5.5 Vdc	500 to 5500	1 mV/step
0 to 10 Vdc	-0.5 to +10.25 Vdc	-500 to +10250	1 mV/step
0 to 20 mA	-0.5 to +20.5 mA	-500 to +20500	1.0 uA/step
4 to 20 mA	3.5 to 20.5 mA	3500 to 20500	1.0 uA/step
+/- 20 mA	-20.5 to +20.5 mA	-20500 to +20500	1.0 uA/step
0 to 1 mA	-0.05 to 1.05 mA	-50 to + 1050	1.0 uA/step

Channel Data Word Values for Engineering Units

Input Type	Signal Range	NI4 Data Format
+/- 10Vdc	-10.00 to +10.00 Vdc	-32768 to +32767
0 to 5Vdc	0.0 to 5.00 Vdc	0 to 16384
1 to 5 Vdc	1.00 to 5.00 Vdc	3277 to 16384
0 to 10 Vdc	0.0 to 10.00 Vdc	0 to 32767
0 to 20 mA	0.0 to 20.0 mA	0 to 16384
4 to 20 mA	4.0 to 20.0 mA	3277 to 16384
+/- 20 mA	-20.0 to +20.0 mA	-16384 to +16384
0 to 1 mA	0.0 to 1.00 mA	0 to 1000

Channel Data Word Values for Scaled Data

Using the value 4 to 20 mA from the Input Type column, the value in Engineering Units is 3277 min to 16384 max. These values are entered in the SCP instruction to scale the variables correctly.

Fig. 19-7

SCP – Scale with Parameters	
Input	
Input Min	3277
Input Max	16384
Scaled Min	
Scaled Max	
Ouptut	

The scaled min and max values that are sent to the PID's process variable are found in the setup documentation of the PID block. The min value is 0 and the max value is 16383. A location must be selected. In this case, the process variable or PV is selected to be N10:28. It is advisable to keep the PID block data separated from other integer data. In order to do keep the data for the PID separated, the data file N10 was created to handle the PID data.

The input address may also be selected. Remember the value is I:s.w where s is the slot number and w is the relative word address down the card. In this case, the slot address chosen is 1 and the w or word address is 0, the first analog input point on the card. The other option for the input in slot 1 is I:1.1.

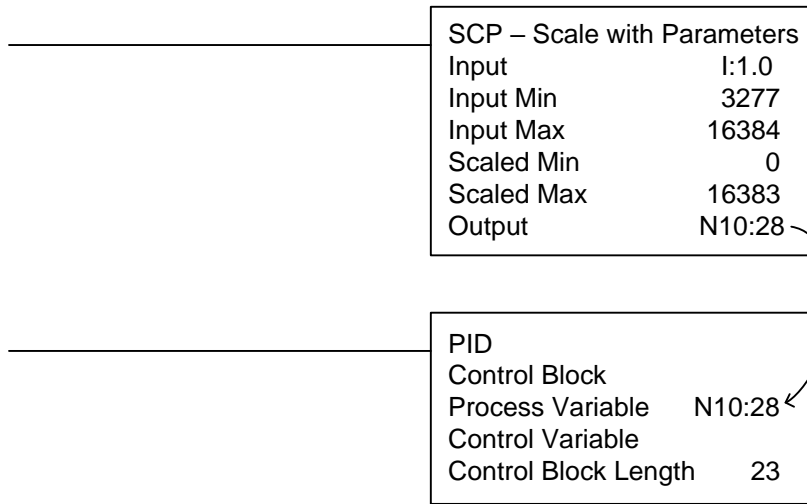


Fig. 19-8 Moving the Process Variable into the PID Block

The control block address is chosen. This address requires 23 contiguous words reserved in an integer table. The block N10:0 (through N10:22) was chosen. Also reserve a location for the control variable or output of the PID function. N10:29 was chosen.

This control variable or output is then sent to the analog output card. Scaling again must be chosen. The min for the PID output is 0 and the max is 16383. These are the same values as are used for the PID input. To use the entire range of values for a PID input or output, choose the range 0 to 16383. Always strive to use the entire range of the PID block when programming an integer PID block. This gives the greatest accuracy.

The scaled output must be ranged to fit a 4 to 20 mA analog output card. Use the values as were found in the reference manual, 6,242 min and 31,208 max. Use the first output point on the same card as the input. Its slot number is 0:1.0. Now, the PID and two SCP blocks can be finished.

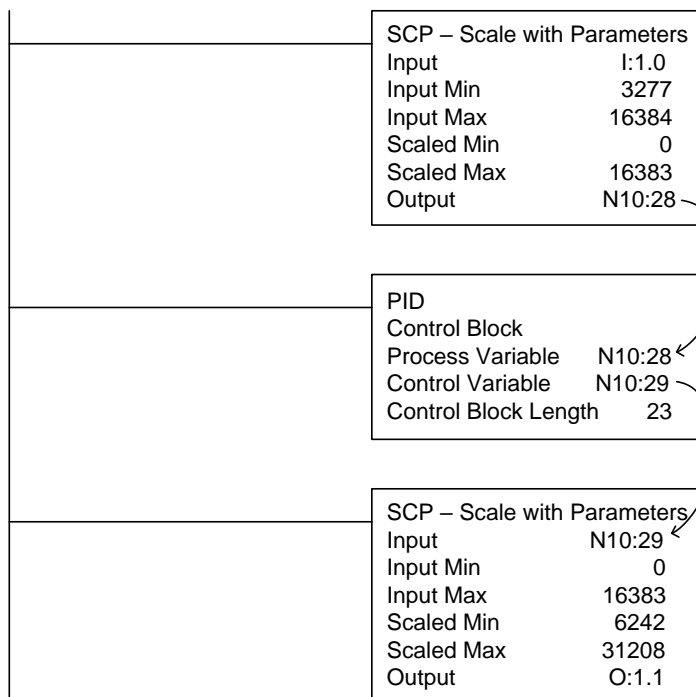


Fig. 19-9 Moving the Variables Into and Out of the PID

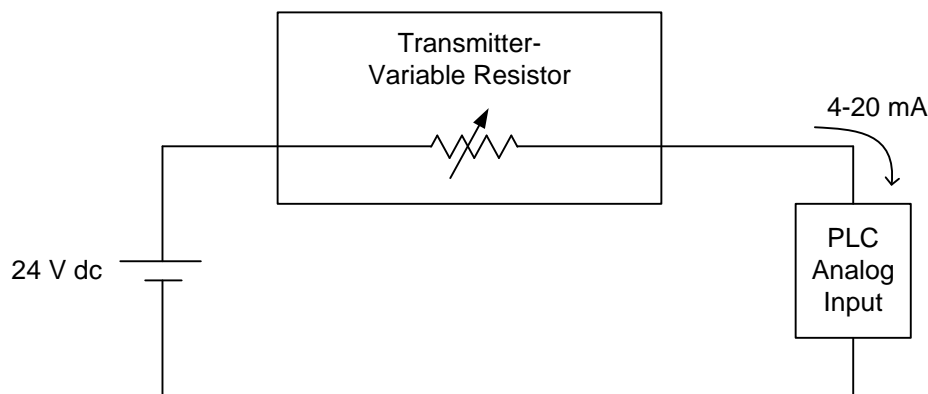
Wiring a 4-20 mA Current Loop

Handling wiring and other hardware issues is found from information in the instruction manual for the module. In the case above, the card used was the 1746-NI04I module from Allen-Bradley. Look specifically in the chapter on installation and wiring.

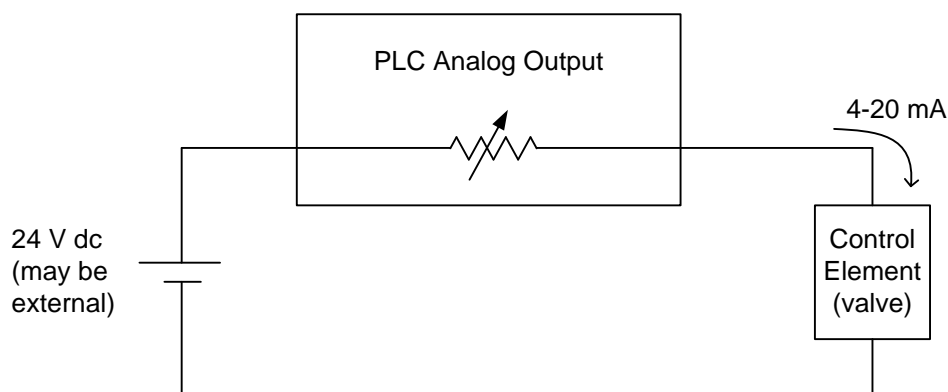
In addition to the actual wiring diagram for the application, important information including dip switch settings should be noted. If possible, all dip switch settings should be copied to the installation drawing for the card or added as notes to the schematic drawings. In the case of the 1746-NI04I card, no dip switches were found.

To wire a 4-20 mA control circuit for a PLC input, wire a loop with the power supply, transmitter, and PLC input. To wire a 4-20 mA PLC output, wire a power supply, valve and output. From the manufacturer's diagram, it should be noted whether the 4-20 mA output requires loop power or the analog output card provides loop power.

For the analog input, the transmitter varies the resistance to the PLC input so that the current ranges from 4 mA for no flow to 20 mA for maximum flow. The transmitter “borrows” enough voltage from the 24 V dc to activate electronics inside the transmitter. The voltage drop across the transmitter does not affect the current range of the loop. The PLC analog output varies the resistance to the control valve in a similar manner.



4-20 mA Analog Input – Current Loop



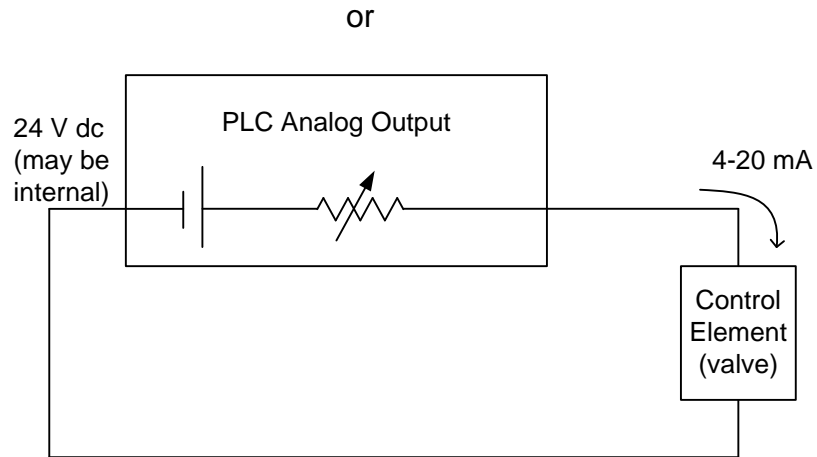


Fig. 19-10 4-20 mA Analog Output – Current Loop

In the case of output cards, care must be taken to find whether or not the 24V dc power supply should be added to the loop. The drawing from the installation manual provides direction here. From the figure below, note that there is no power supply needing to be added in the output current loop diagram for this specific card (NI04I).

The figure below shows the catalog information for wiring this card. In fact, the analog output does not need a power supply since the output furnishes this power internally. The term "analog source" for the input implies inclusion of the 24V power supply. Load for the output implies no external power supply. Note the jumpers installed for inputs not used.

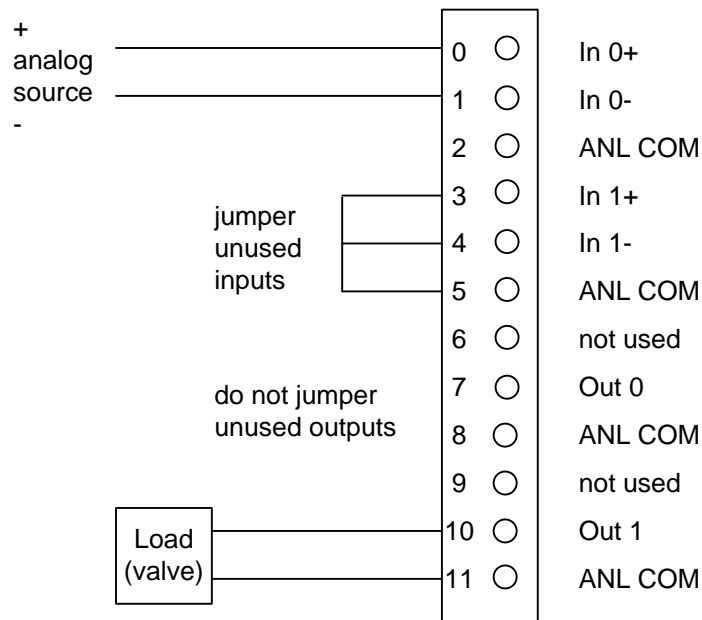


Fig. 19-11 4-20 mA Analog I/O – Current Loop (NI04I)

Configuring the SCP and PID Instructions for the SLC

The description of the SCP instruction mentions that the inputs may be integer, floating point, immediate data values, or indirect referenced values. The minimum and maximum values for both input and output form a range over which the variables are scaled. The instruction solves the equation $y = mx + b$ without the user responsible to calculate actual values for 'm' and 'b'.

Care must be taken to keep the program performing in an acceptable manner if the input value is less than the card minimum value. The scaled output value should continue to solve the equation and the output value should scale to less than the minimum value of the instruction. The same result should also occur if the value exceeds the maximum.

In the Instruction Help description, the PID block is described:

"This output instruction is used to control physical properties such as temperature, pressure, liquid level, or flow rate of process loops.

The PID instruction normally controls a closed loop using inputs from an analog input module and providing an output to an analog output module as a response to effectively hold a process variable at a desired setpoint."

The PID instruction can be chosen to be operated in either the timed mode or the STI mode. In the timed mode, the instruction updates the output algorithm periodically at a rate selected in the block. In the STI mode, the PID instruction is placed in an STI (Software Timed Interrupt) subroutine. The PID block updates the PID algorithm each time the STI subroutine is called. A-B points out that the STI time interval and the PID loop update rate must be equal in order for the equation to perform properly. The suggested time duration for the STI or timed mode is .1 second.

A Setup screen is provided on the PID instruction.

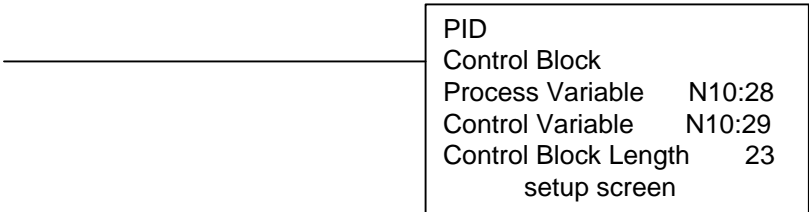


Fig. 19-12 Example PID Instruction

From the A-B Text and the Instruction Help Screen is shown the Block Layout of the PID Instruction:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 0	EN		DN	PV	SP	LL	UL	DB	DA	TF	SC	RG	OL	CM	AM	TM
Word 1	PID Sub Error Code (MSB)															
Word 2	Setpoint SP															
Word 3	Gain Kc															
Word 4	Reset Ti															
Word 5	Rate Td															
Word 6	Feed Forward Bias															
Word 7	Setpoint Maximum (Smax)															
Word 8	Setpoint Minimum (Smin)															
Word 9	Deadband															
Word 10	INTERNAL USE – DO NOT CHANGE															
Word 11	Output Max															
Word 12	Output Min															
Word 13	Loop Update															
Word 14	Scaled Process Variable															
Word 15	Scaled Error SE															
Word 16	Output CV% (0-100%)															
Word 17	MSW Integral Sum															
Word 18	LSW Integral Sum															
Word 19	Altered Derivative Term (Low word)															
Word 20	Altered Derivative Term (High word)															
Word 21	Time of Last Update															
Word 22	Setpoint Old Value															

The table above corresponds to N10:0 through N10:22 found in our example above. Word 0 (N10:0) is used for bit control storage. For example, bit 1 is the AM or Auto/Manual bit. When bit 1 is on, the block is in manual. When bit 1 is off, the PID block is in auto. The address for AM is N10:0/1. Words 1 through 22 are used for constants and variables used in the solution of the PID algorithm.

The PID Setup Screen shown below describes variables found in the table above that may be changed from the programming software.

Solving the PID Block and Adding the HMI

Once the analog value of the process variable is mapped from the SCP instruction to the PID block, the PID block solves the equation for the Control Variable (CV) or Output. A more thorough explanation of how the output is achieved may be found in a text on control systems. Equations vary but the three most common equations are given later in the chapter.

The PID block has two analog inputs. One is the PV or process variable and the other is the SP or setpoint. The setpoint is manually entered into the PID block. This may be done through the PID Setup screen, through an HMI such as PanelView, or through a program statement (a MOV). If the SP is entered manually through the program, the SP is considered static and should never be changed by operator control since an operator is not generally considered reliable enough to enter variables through the RSLogix500 Setup Screen.

The PID Setup screen is pictured below. The setup screen allows the engineer or technician full capability of modifying the PID block.

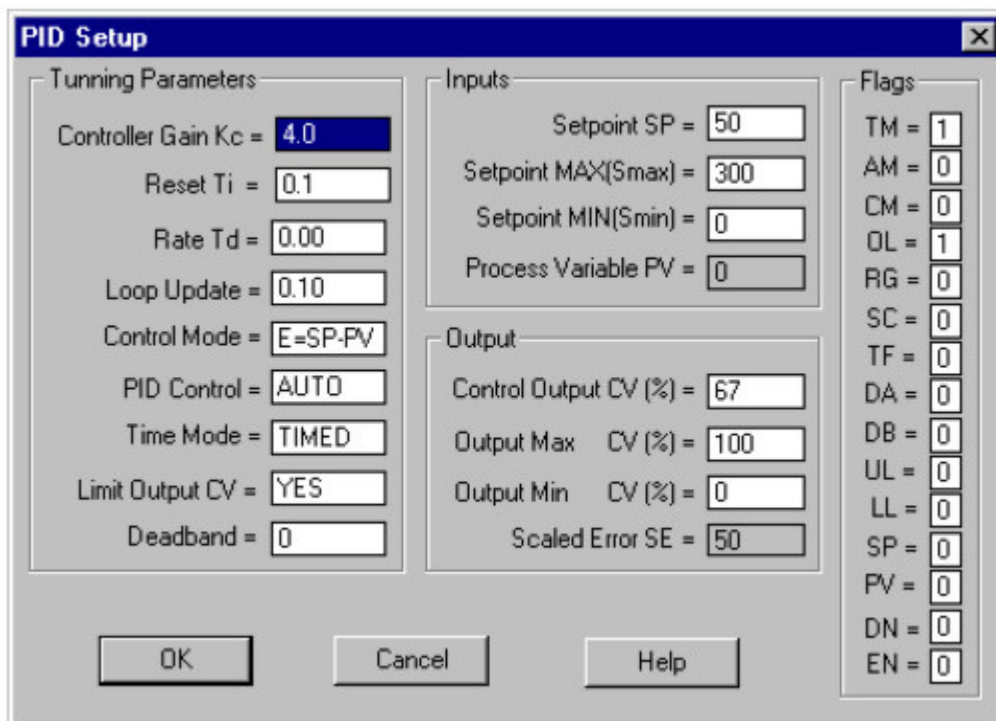


Fig. 19-13 SLC PLC Startup Block in RSLogix 500

The SP may be entered through the PID Setup screen. The PV is entered using the SCP instruction.

From the A-B Instruction Reference Manual:

“Process Variable PV is an element address that stores the process input value. This address can be the location of the analog input word where the value of the input A/D is stored. This value could also be an integer if you choose to pre-scale your input value to the range 0 to 16383.”

The output is referred to as the CV or Control Variable. It is described in the same manual as:

“Control Variable CV is an element address that stores the output of the PID instruction. The output value ranges from 0 to 16383, with 16383 being the 100% ‘on’ value. This is normally an integer value, so that you can scale the PID output range to the particular analog range your application requires.”

The PID block is very much like a black box function with inputs entering and outputs leaving the block. The block diagram for the PID block in auto is:

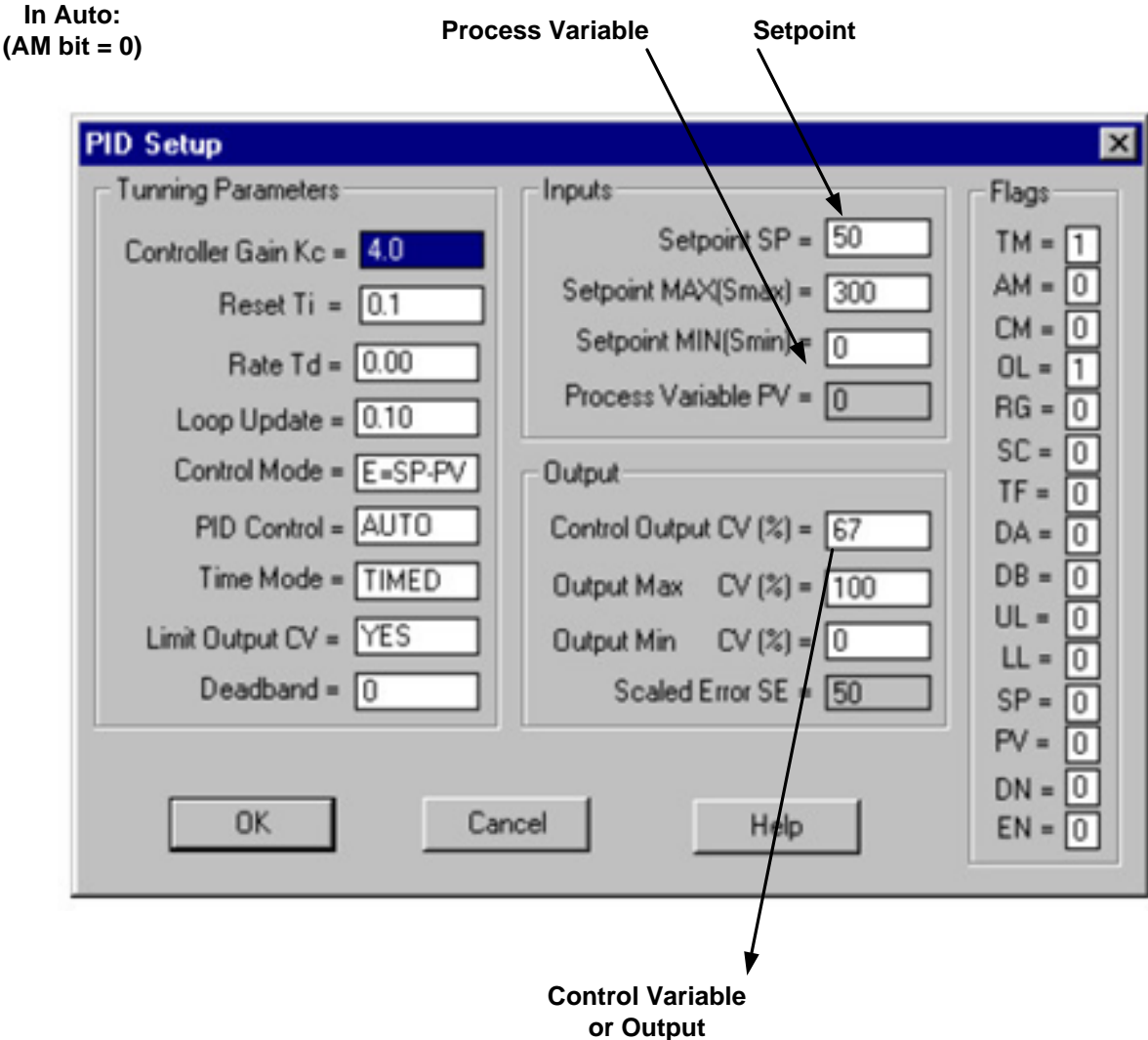


Fig. 19-14 Using Setup Screen

The PID algorithm is solved while the block is in auto. Auto is determined by the status of the AM bit. When $AM = 0$ the operation is automatic. When $AM = 1$, the operation is manual.

The PID algorithm does not output a value for the PID block if the block is in manual. It is as if the block has been manually disengaged. The PV or SP may change and the output stays at its last value unless a new value is written into the CV location. The CV location may be overwritten in manual. In auto, the PID block constantly writes the value to the CV. The range of the CV is from 0 to 16383. Writing to the CV allows the user to manipulate the valve in the manual mode.

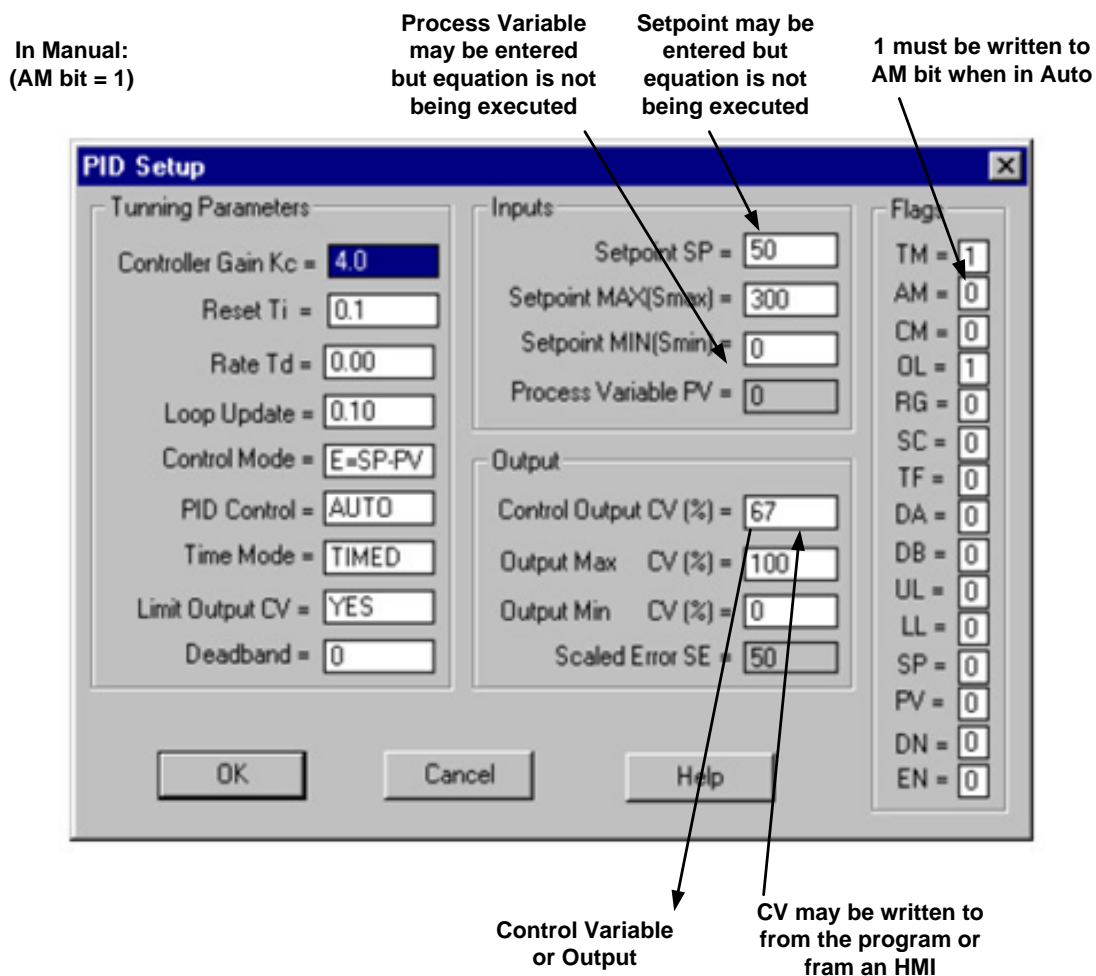


Fig. 19-15 Additional Use of Setup Screen

Another bit that must be set correctly for the PID block to work is the Control (CM) bit. It determines whether the error term $E = SP - PV$ or $E = PV - SP$. If the CM bit is set incorrectly, the valve will quickly go to full on (100%) or full off (0%). This bit is never to be set by an operator. Use the PID Setup screen to set it. The bit is not to be changed after it is set in the initial configuration of the auto mode.

The simple PID algorithm from the SLC processor demonstrates many important steps in implementing the PID block successfully. First, the input must be correctly signal conditioned and the output signal conditioned as well. The wiring must be correct. The PID block must be correctly configured including all min and max values plus all tuning parameters. Then the engineer can control the program either in manual or auto from the programming helps menu. The PID block must be placed in a block that executes on a clocked interrupt or the PID block itself must be programmed to execute on a timer in the main or OB1 block. Either method works but the preferred method is to program the PID algorithm in a separate timed interrupt block. Also the data must be guaranteed to be 'fresh'. That is, the data that is used for the algorithm must have been gathered recently. This may be as recent as an immediate read or from a scanned card that reports to the main CPU on a regular basis. This data must be guaranteed to have been read less than 10% of the time since the last execution of the PID block. This is a rule of thumb – 10%.

We will also discuss the development of a faceplate, the operator interface used for input of data as well as manipulation of the mode of the PID block. The operator is not to be given control of all the parameters including tuning parameters as well as max and min values for the signal and

alarms. Faceplate design can emulate the manufacturers' PID control panel or enhance the data shown on the panel. They are to be intuitive and easy to use. The PID operator interface design can vary from very simple to complex. The best faceplate design incorporates what the operator needs to know without too many alarms or blinking lights. Historical data plots are a plus.

Design of a Faceplate for PID Block

Faceplates of some stand-alone PID controllers are shown below. These include the Red Lion stand-alone TCU controller and the Honeywell stand-alone controller faceplates.



Fig. 19-16

Stand-alone PID controllers such as the Red Lion TCU controller solve the PID equation in a manner similar to the PID equation solved in the PLC. The Red Lion display is referred to as the faceplate. HMI displays are used to allow the operator to run the process from a display in a manner similar to the Red Lion faceplate. To run the PID successfully in the PLC, several parameters should be available on the display to adjust the process of controlling the PID equation.

Commonly used tags in the HMI are:

- Auto/Manual
- Setpoint
- Process Variable
- Output (CV)
- Error (Deviation) (May be on restricted access page.)
- Deadband (May be on restricted access page.)
- Gain, Reset, Rate (May be on restricted access page.)

Mode switches such as Auto/Manual are included in the SLC PID block. Other modes normally used but not part of the SLC PID block include:

- Local/Remote
- Maintenance

In Local, the operator is able to change the setpoint manually and verify the output's response while the PID loop is in auto.

In Remote, the process (program) sets the SP and the PID loop responds to the changes. The PID

loop is in auto mode in both local and remote modes. Remote mode is referenced as Cascade mode by some PID controller manufacturers.

In Maintenance mode, the loop is in manual and any variable can be changed from the operator station. This mode should be **password** protected.

A faceplate may be drawn on the HMI similar to the one below. This faceplate is typical for a system of PID loops controlling a process.

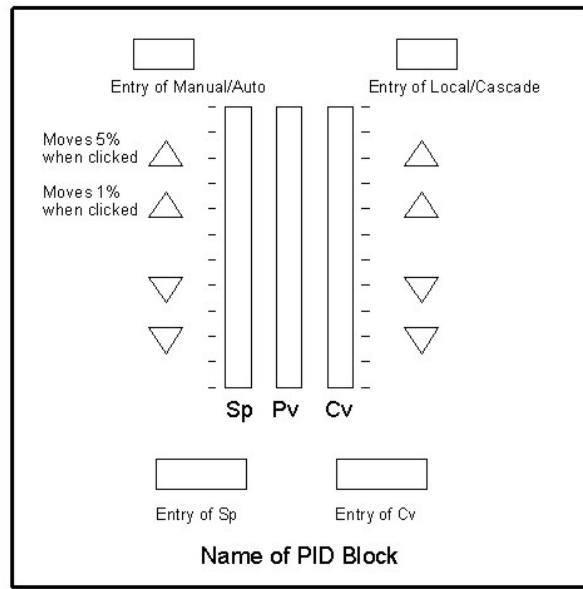


Fig. 19-17

The triangles on the left and right side of the bar graphs are used to add or subtract 5% or 1% of the SP or CV. They provide a quick method to adjust SP or CV to get to a desired number. The more exact approach is to enter a number in the data box for either SP or CV. This approach is slower to implement than the method of touching a triangle when making small changes.

From the example of the PID Block for the SLC controller, to implement a PID Block successfully, the PID Block must be programmed with some provision for scaling, whether through a programming block or other means. The analog input or PV must be in an appropriate range for the block to calculate an error based on the difference between the PV and a setpoint or SP. In addition, the output or CV must be correctly scaled to an output.

Also, the PV and CV must be wired to analog points correctly.

Processes in Lab

Two processes in the lab are pictured on the following page in Fig. 19-18. The one on the left is the water valve. The one on the right is the ball-in-tube. Fig. 19-17 shows the flow sensor for the water valve. Information on the laser, the ball's feedback sensor, is found in the instructions for the laser and the setup of the analog output in Fig. 19-19.

The two processes are controlled by the two plc's in the lab, A-B's Compact Logix processor and the Siemens S7-1200 processor. The feedback devices are both 4-20 mA input devices. The valve requires 4-20 mA from the CompactLogix processor to set the position while the fan motor is controlled by a pulsed 24 V output from the Siemens PLC.

The following is a bill of material to construct the flow valve system shown below in Fig. 19-18.

Quantity	Item	Description	Unit Price	Amount
1.00	FLO-7104 3/4"	FLOW-TEK BALL VALVE 316SS FULL PORT, NPT ENDS WITH MOUNTING BRACKET AND COUPLING		
1.00	MAX-UT-26-DA	DOUBLE ACTING PNEUMATIC ACTUATOR P/N UT26	1,268.00	1,268.00
1.00	ACC-A51236AT	ACCORD 4-20mA POSITIONER P/N A51236AT SAME AS A51136AT W/BEACON COMPLETELY ASSEMBLED AND TESTED		
1.00	MAR-M1FR2NHFM	0-120 PSI REGULATOR 5 MICRON P/N M1FR2NHFM COMPLETE CONTROL VALVE ASSEMBLY MOUNTED AND TESTED WITH PRESURE GUAGES PER		



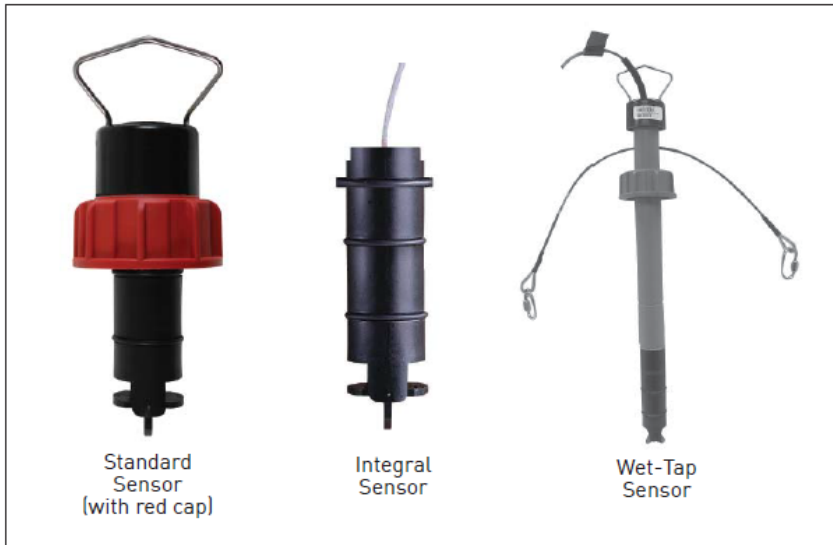
Fig. 19-18 Water Valve Hardware



Ball in Tube Hardware

The valve on the wall was a first lab for EET to activate. It had been available for students from about academic 2004. We had used it over the years with good success.

Signet 515 Rotor-X Paddlewheel Flow Sensors



Features

- Operating range 0.3 to 6 m/s (1 to 20 ft/s)
- Wide turndown ratio of 20:1
- Highly repeatable output
- Simple, economical design
- Installs into pipe sizes DN15 to DN900 (½ to 36 in.)
- Self-powered/no external power required
- Test certificate included for -X0, -X1

Fig. 19-19 The Flow Sensor Input

The flow sensor is a paddle wheel placed in the flow of water. There is a calibrated readout for the flow meter that displays the flow in gallons per minute. Included with the flow sensor is a flow instrument read-out. This read-out is separate from the PLC and HMI and is used by personnel in the field to read the 4-20 mA reading from the transmitter to the PLC input. It is a useful instrument in that it verifies externally from the PLC a value that can be seen in the program.

System Overview

Panel Mount Signet 8550 Flow Instrument (Includes mounting bracket and panel gasket) 	Pipe, Tank, Wall Mount Signet 8550 Flow Transmitter 	Integral Mount Signet 8550 Flow Transmitter
Signet Universal Adapter Kit (3-8050) (sold separately) 	Signet Integral Adapter Kit (3-8051) (sold separately) 	
Signet Flow Sensor (sold separately) 515 2507 2540 525 2536 2551 2000 2552 2100 	Signet Flow Sensor (sold separately) 515 2507 2540 525 2536 2551 2000 2552 2100 	Signet Integral Mount Flow Sensor (sold separately) 3-8510-XX 3-8512-XX
Signet Fittings (sold separately) 		



Fig. 19-20 Signet Flow Instrument as seen in Lab

The valve has been discontinued as an active lab due to the possibility of water flooding the downstairs. What had been a good lab is no more.

The ball-in tube lab was built over the 2013-14 academic year. It has served students well.

The laser is the feedback device for the ball-in-tube experiment. The laser gives an accurate position of the top of the ball. Specifications for the laser are given in the following figure.

Installation Instructions 45LMS Laser Measurement Sensor

IMPORTANT: SAVE THESE INSTRUCTIONS FOR FUTURE USE.

Description

The 45LMS family of long distance laser sensors is available in a variety of measuring ranges. The 8 m diffuse and 50 m retroreflective models use a Class 1 visible red laser and the 15 m diffuse models use a Class 2 visible red laser. The discrete and analog outputs can be easily set using the 5-step rotary switch and the push button. Potential applications include object position (analog output) and object detection (background suppression with discrete output).

This sensor utilizes the Time of Flight (ToF) principle and has a relatively small beam spot even at 15 m away. The sensor is completely self-contained and does not require any external control devices which add cost and require additional mounting space.

The 45LMS is easily set up by mounting the sensor such that the target is within the operating range of the sensor and teaching in the appropriate set-points required for the application. All sensors in this family have one discrete output with one analog output. The discrete output can be wired for either Light Operate (L.O.) or Dark Operate (D.O.) and the analog output is automatically scaled between the selected set-points with either a positive or negative slope.

The 45LMS is an excellent solution for long range detection and measurement applications including: distance measurement, verifying material position, stack level, thickness measurement, roll diameter, positioning fixtures, error proofing inspection, long standoff distance, level monitoring, crane crash protection and other difficult applications that exceed the capabilities of standard diffuse or background suppression photosensors.

Features

- Eye Safe Class 1 or Class 2 laser (by model)
- 8 m (26 ft.), 15 m (49 ft.) or 50 m (164 ft.) sensing range, dependent on model
- One discrete output (1 x NPN/PNP) and one analog output (1 x 4...20 mA)
- Easy setup of switch points or analog scaling using programming buttons
- IP65 enclosure
- Self-contained sensor

ATTENTION



This installation instruction should be read and understood before operating the sensor.
The 45LMS sensor should only be installed by qualified personnel.
The 45LMS is not a safety component as described by the EU machinery directives.

General Specifications

Certifications	UL, cULus, and CE marked for all applicable directives
Operation	
Sensing Beam	Class 1 laser, visible red 660 nm (for 8 m & 50 m models) Class 2 laser, visible red 660 nm (for 15 m model)
Spot Size	< 10 mm (0.39 in.) at a distance of 8 m (26 ft) at 20 °C (68°F) < 15 mm (0.59 in.) at a distance of 15 m (49 ft) at 20 °C (68°F) < 50 mm (2 in.) at a distance of 50 m (164 ft) at 20 °C (68°F)
Sensing Distance	0.2...8 m (0.66...26.25 ft) diffuse 0.2...15 m (0.66...49.21 ft) diffuse 0.2...50 m (0.66...164.04 ft) retroreflective
Absolute Accuracy	± 25 mm (± 0.98 in.)
Repeatability	< 5 mm (0.20 in.)
Angle Deviation	± 2° max.
Reference Target	Kodak white (90%)
Temperature Influence	≤ 0.25 mm/K typ.
Electrical	
Operating Voltage	10...30V DC (18...30V DC when operating in IO-Link mode)
Current Consumption	≤70 mA @ 24V DC
Discrete Output Type	1 NPN/PNP output, short-circuit protected, reverse polarity protected
Discrete Output Rating	30V DC max. / 100 mA max.
Analog Output Type	1 analog output 4...20 mA, short-circuit/overload protected
Switching Frequency	50 Hz
Response Time	10 ms
Mechanical	
Housing Material	Plastic ABS
Optical Face Material	Plastic pane
Control Inputs	5-step rotary switch for operating modes selection Push button for set-point teach
LED Indicators	Green: Power Yellow: Output switching states Green/Yellow Flashing 2.5 Hz: Teach indication Green/Yellow Flashing 8.0 Hz: Teach error
Connection Type	4-Pin DC Micro (M12)
Supplied Accessories	None
Environmental	
Operating Environment	IP65
Vibration	10...55 Hz, 0.5 mm amplitude; 3 planes; meets or exceeds IEC 60068-2-6
Shock	30 g; 11 ms; 3 planes: meets or exceeds IEC 60068-2-27
Operating Temperature	-30...50 °C (-22...122 °F)
Storage Temperature	-30...70 °C (-22...158 °F)

Fig. 19-21 Instructions for Laser for Ball-in-Tube Lab

Calibration of the analog output for the laser is described in the following figure. This calibration may be ignored, however, due to the programming techniques used.

Setting the analog output: Q2

The 4...20 mA output can be defined as any range within 200 mm to the maximum range of the sensor, as either a rising or falling slope, as described below. The default analog output setting for Q2 is A = 200 mm (8 in.) and B = 5,000 mm (16 ft) for all sensor models. Minimum window for setting the analog span is 21 mm (0.83 in.)

Positive Slope

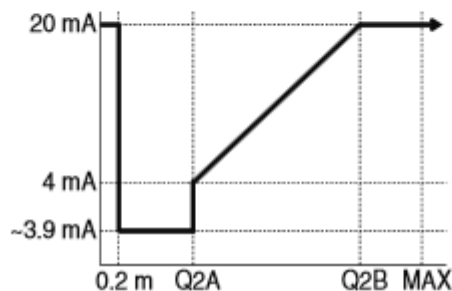


Fig. 19-22

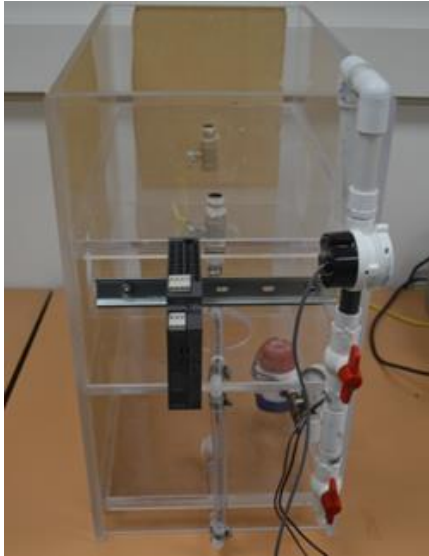
In the Positive Slope mode (also called Rising Slope) a target positioned at the closer set-point results in an analog output of 4 mA while a target at the farther set-point results in an output of 20 mA, with the analog output scaled linearly in between. In this mode, the sensor will output 20 mA when the target is outside of the operating range, which is 0...200 mm (0...8 in.) and anything greater than the maximum sensing range.

1. Place a target at the minimum Teach-point.
2. Move the Rotary Switch to position Q2-A.
3. Press and hold the SET button until the Green and Yellow LEDs flash simultaneously¹.
4. Place a target at the maximum Teach-point.
5. Move the Rotary Switch to position Q2-B.
6. Press and hold the SET button until the Green and Yellow LEDs flash simultaneously¹.
7. If the Teach is successful, move the Rotary Switch to RUN.

The calibration can be ignored with the program as written. When the program initializes, the ball is sent to the top of the tube and stays there for a short period of time. The laser output at this value becomes the 100% value. When at rest at the bottom, this value is the 0% value. While the calibration may vary for different lasers installed, this process sets the limits of the process from which values for the high and low limits may be set. If this kind of self-calibration is possible with your process, the calibration set-up becomes very easy.

Tank over Tank Level Control Lab

This lab was an effort to mimic a lab from a major educational equipment manufacturer. The first attempt is pictured below. The later design is pictured further below. The number of different sensors used in the design is significant. What first seemed to work may not work in the final design. This was found to be the case in both the level and flow sensors.



Level Control of Upper Tank with Multiple Drains and Feedback from Level Sensor

Fig. 19-23

The first system used a cheap level sensor before settling on the sonic sensor (yellow) seen below. The flow sensor changed from a cheap \$10 sensor to a \$110 sensor and finally a better \$160 sensor. These changes were seen as necessary to control the process accurately.

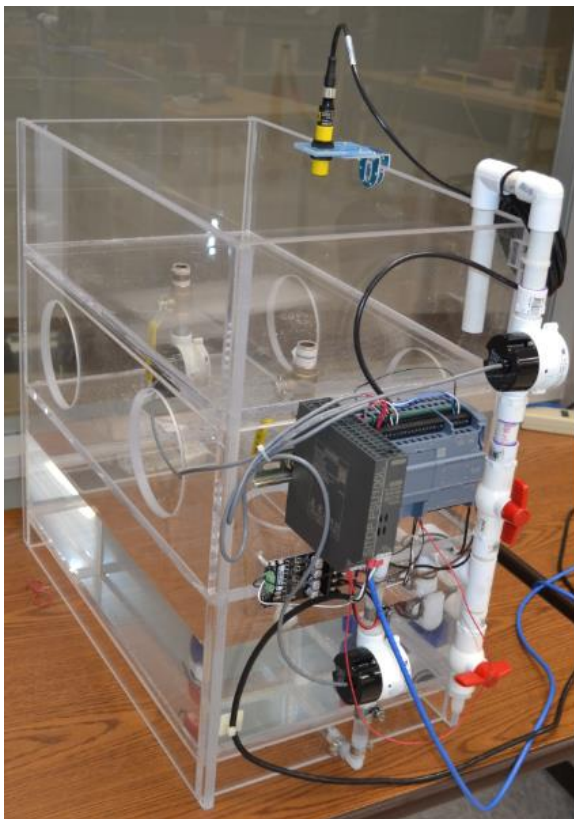


Fig. 19-24

The pump control was from a digital output to a drive control module and finally to the pump motor. The drive control module is shown below as attached to the system. The actual device is shown below as well. The pump is a submersible bilge pump selected by the plastics manufacturer known by him since his experience had been with boats and boat construction.

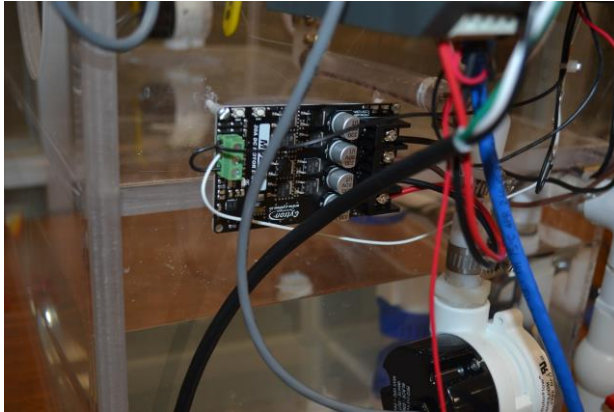


Fig. 19-25



Cytron 20Amp Bi-Directional 6V-30V DC Motor Driver Speed Controller 60A Peak

★★★★★ 12

\$19⁸⁰

FREE Shipping

Only 14 left in stock - order soon.

Fig. 19-26

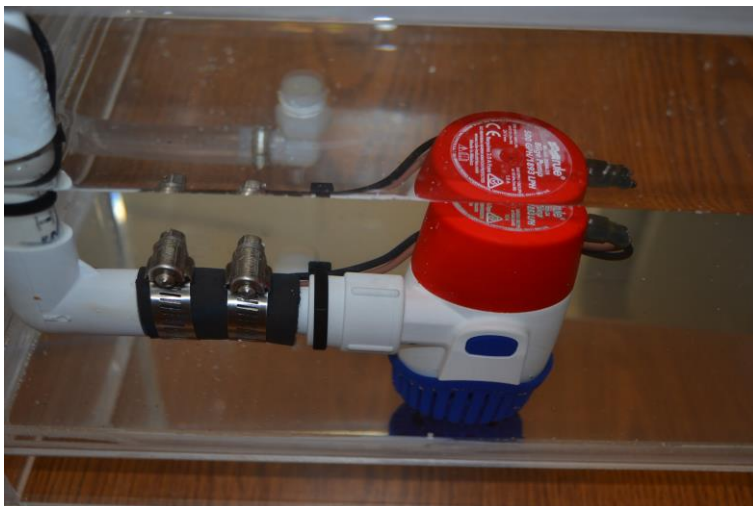


Fig. 19-27

The level control selected first had been one that was attached to the Arduino microprocessor. That level control failed. It is not even on the pages of Arduino sensors at this time. Seems as if more than one discovered that it didn't work. This is a common story with low-cost sensors. Many will work for a while. Some do not work at all. The level sensor below is an industrial sensor and is guaranteed to work long-term. It does cost

significantly more but is worth the money.

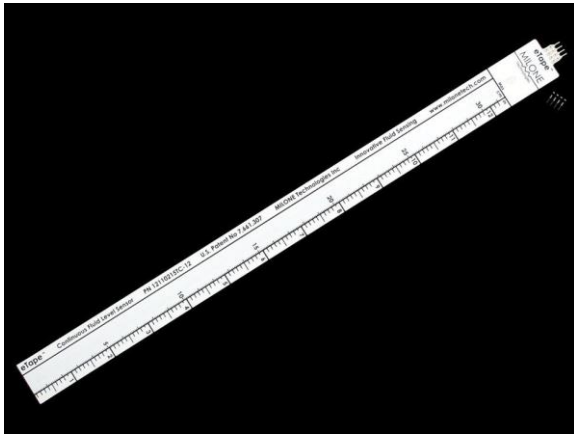
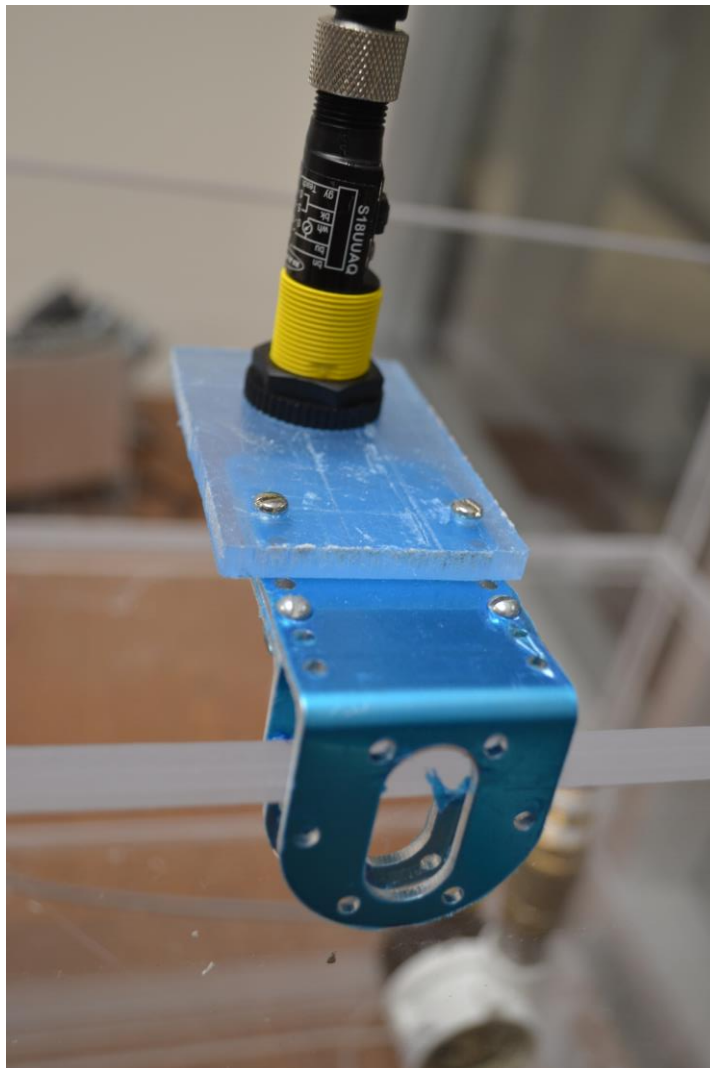


Fig. 19-28

The price of the sensor is approximately \$250.



Compact ultrasonic sensor in straight or right-angle housing.

- Senses from 30 to 300 mm
- Available in analog or discrete models
- Features minimal dead zone and eliminates dead zone if used in retrosonic mode
- Ideal for material handling and packaged goods applications, such as bottling or liquid level detection and control for small containers
- Available in straight or right-angle versions with a wide variety of mounting hardware for enhance sensing versatility
- Offers programmable background suppression
- Compensates for temperature, for greatest sensing accuracy
- Simplifies setup with push-button and remote TEACH-mode programming
- Shows status during setup and operation, using highly visible LEDs indicators

Fig. 19-29

S18UUAQ BANNER ENGINEERING ULTRASONIC, VOLTAGE QD SENSOR



S18UUAQ

Price: \$285.00

Manufacturer SKU: S18UUAQ

Manufacturer ID: 02700

This item qualifies for
FREE SHIPPING over \$300!

Availability: Usually Ships in 3 to 5 Business Days

Part Number: S18UUAQ

Qty:

ADD TO CART

The yellow ultrasonic level transmitter is shown above. It worked very well and gives a stable accurate signal to the PLC from the tank level. The output of this device is 4-20 mA.

We now look at the flow sensors tried. The first again was a low-cost sensor. It worked for a little while (about an hour or so) only to fail. We purchased a number of these and they all failed in a short while. The electronics was not robust and the signal stopped shortly after initially running.



DIGITEN G3/4" Water Flow Sensor, Food-Grade Hall Effect Sensor Flow Meter Flowmeter Counter 1-60L/min - Arduino, Raspberry Pi,...

★★★★☆ ~ 10

\$11.99

FREE Delivery by **Wed, Apr 13** for Prime members

Only 8 left in stock - order soon.

The flow sensor shown here is the second. The third device is shown further below. At the bottom is a fourth which was held in reserve but may be used in the future. This sensor worked (but was not accurate). We looked at it because we wanted something that would work. It worked but if we want an accurate signal across the range, it lacked a great deal.



Gems Sensors 155481 RFO Series Flow Rate Monitoring Sensor

Brand: Gems Sensors

★★★★☆ ~ 1 rating

Price: \$110.00 ✓prime & FREE Returns

- Body Material: Polypropylene
- Port Size: 1/2" NPT Female
- Flow Range (GPM): 1.5-20.0
- Low Flow Accuracy: +/- 7%
- Standard Flow Accuracy: +/- 15%

\$110.00

✓prime & FREE Returns

FREE delivery: **Tuesday, April 13**

Order within 3 hrs and 22 mins
Details

Only 13 left in stock
(more on the way).

Qty: 1

The sensor below is the third flow sensor and is the best so far. It is more costly but is accurate across the range.

FLOW SWITCHES & SENSORS

FT-110 SERIES TURBINE FLOW SENSOR

Part Number: 173934-C



PRODUCT DESCRIPTION:

FT-110 Series Turbine Flow Sensors are ideal for OEM applications involving low flow liquid monitoring. FT-110 mount in any orientation, have accuracy $\pm 3\%$ of reading and low flow rates of 0.5 to 30 L/Min (0.1 to 8 GPM).

- Low Flow Rates of 0.5 to 30 L/Min (0.1 to 8 GPM)
- Accuracy $\pm 3\%$ of Reading
- Mounts in Any Orientation

This meter was found and is possibly a useful flowmeter for this project. Its cost is significantly less than the two above but has not been validated yet. The cost is significantly less. The one above is about \$160 and this one is about \$60.



Vision Turbine Meter

Models BV1000, BV2000 and BV3000

for Low Viscosity and Non-Aggressive Liquids NSF/ANSI Standards 61 and 372 Certified

DESCRIPTION

The Vision Turbine Meters comply with the lead-free provisions of the Safe Drinking Water Act. Available models include meters that are:

- Bisphenol A (BPA)-free
- Certified to NSF/ANSI Standards 61 and 372

The meters are designed for flow measurement of low-viscosity, aggressive and non-aggressive liquids alike, including demineralized water, alkaline solutions, oils, salad oil, fuel/fuel consumption, beverages, water solutions and coolants.

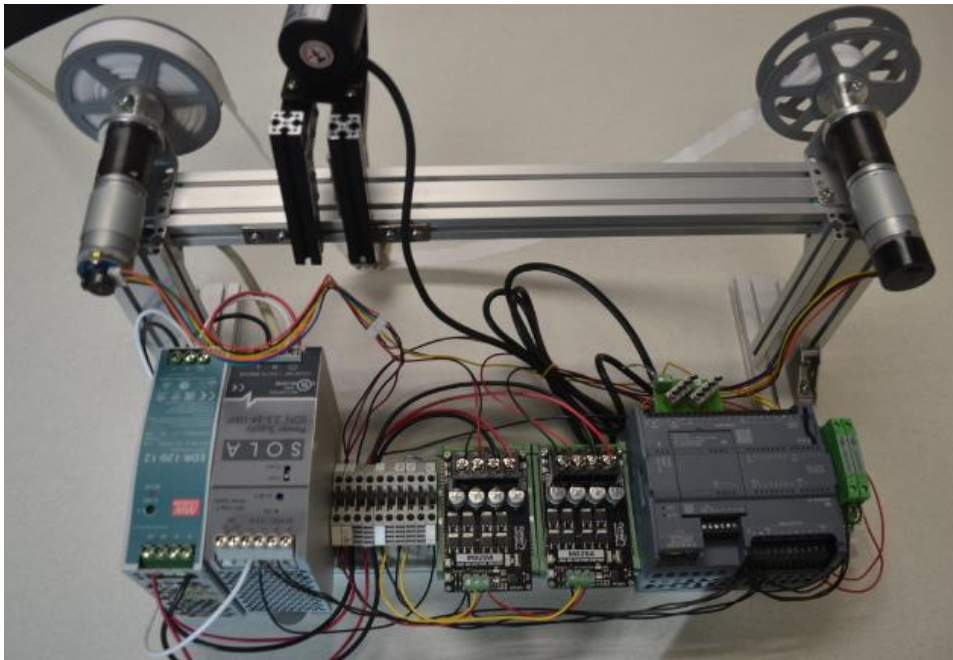
- The BV1000 flow range is 0.026...0.65 gpm (0.1...2.5 lpm)
- The BV2000 flow range 0.13...9.2 gpm (0.5...35 lpm)
- The BV3000 flow range 1.32...17.17 gpm (5...65 lpm)

The meter is especially suitable for washing machines, dishwashers, coffee machines, laser cooling plants, solar solutions, bakery machines, steam cooking machines in large kitchen plants, and CD or DVD cleaning.



Tape Rewind Machine

The design shown below gives speed control for the two dc motors with tension control between the two.



Later Motor Speed and Position Control Design

Fig. 19-34

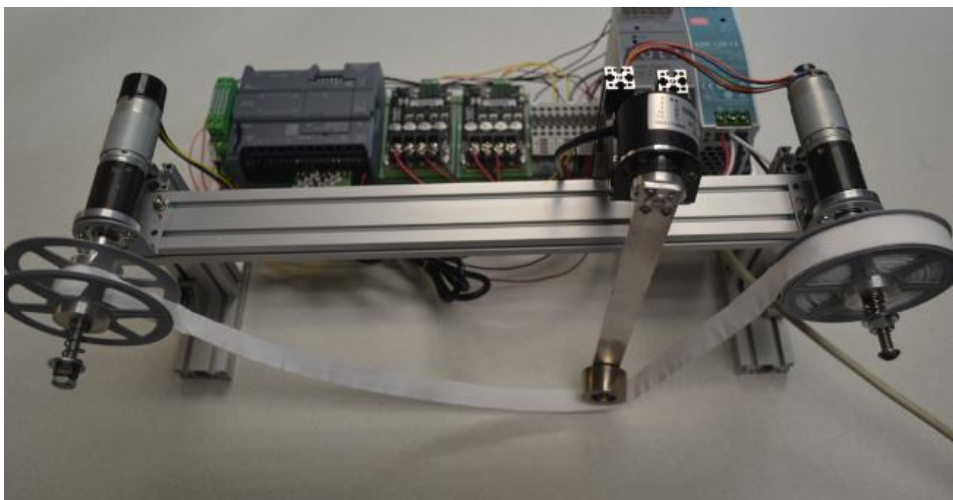


Fig. 19-35

In this design, two motors are involved with a tensioner between. The motors cannot both run at constant speed. One can run at constant speed or at a ramped speed. The second follows the first based on the angle of the dancer roll between the two. The tension on the second can be changed based on the angle of the tension arm. Weight can be added to the arm if additional tension is desired.

Combining of two speed/position-controlled motors results in a lab similar to the one above in Fig 19-83. This lab is inexpensive and provides a pair of PID loops to control the two dc motors and a third PID loop to control the tension between the two. The third loop uses the dancer roll potentiometer as a feedback device. This lab concentrates on loop-in-loop control. Also important are start-up control issues. The lab also asks the question of which loop is the master.

For instance, should the right loop be constant speed? Should the left loop be constant speed? Or should the speed be constant across the dancer roll? The program is written differently for each. Also, a sensor must be added if the dancer roll is to be constant speed. This project has many different possible results depending on where the design starts. The advances from the earlier toilet paper lab to the present design are many and include the addition of 80-20 extruded aluminum instead of the cheaper erector-set metal construction. This one addition gave added stability to the machine from the earlier design.

All present labs use the Siemens PLC due to the flexibility of the I/O to control analog quantities.

Siemens Analog Inputs and Outputs

The Siemens' PID implementation is used in all the active applications shown above. First, the address of all I/O is required as well as the wiring diagram for each analog point. The S7-1200 has two analog inputs located on the controller.

Addressing for the two analog input channels is found below: IW64 and IW66. The two analog inputs are wired to these two points and programmed with these addresses.

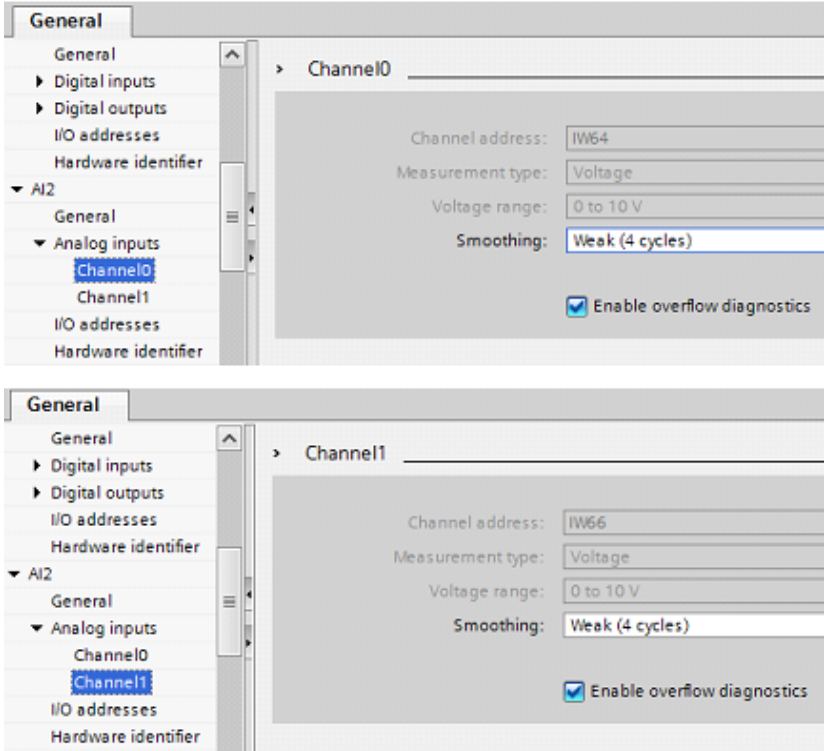


Fig. 19-36

To read or write an analog value, use the immediate read or write instruction as shown below:

Table 4-4 Memory areas

Memory area	Description	Force	Retentive
I Process image input	Copied from physical inputs at the beginning of the scan cycle	No	No
I_:P1 (Physical input)	Immediate read of the physical input points on the CPU, SB, and SM	Yes	No
Q Process image output	Copied to physical outputs at the beginning of the scan cycle	No	No
Q_:P1 (Physical output)	Immediate write to the physical output points on the CPU, SB, and SM	Yes	No
M Bit memory	Control and data memory	No	Yes (optional)
L Temp memory	Temporary data for a block local to that block	No	No
DB Data block	Data memory and also parameter memory for FBs	No	Yes (optional)

¹ To immediately access (or to force) the physical inputs and physical outputs, append a ":P" to the address or tag (such as I0.3:P, Q1.7:P, or "Stop:P").

Use a cyclic interrupt event to house the PID function. The event is defined as an OB or Object Block. We will use OB 30 for the program containing the PID Block for the present applications.

Analog values are available from high-speed digital input pulses. Analog output values may be realized through PTO or PWM signals from digital outputs. An example is the Tank over Tank problem discussed in Chapter 25 of the Hybrid Lab Text. The configuration of the pulse input is as follows:

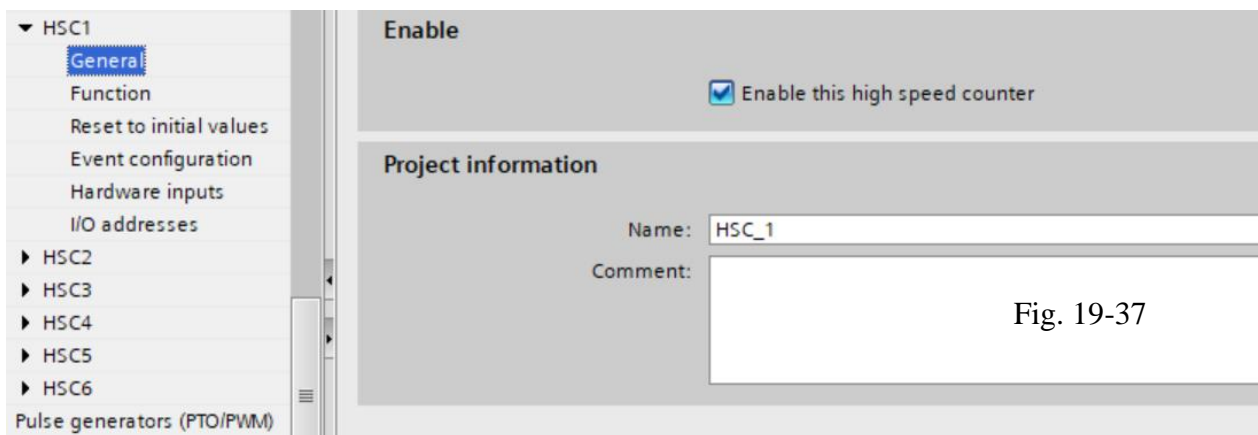
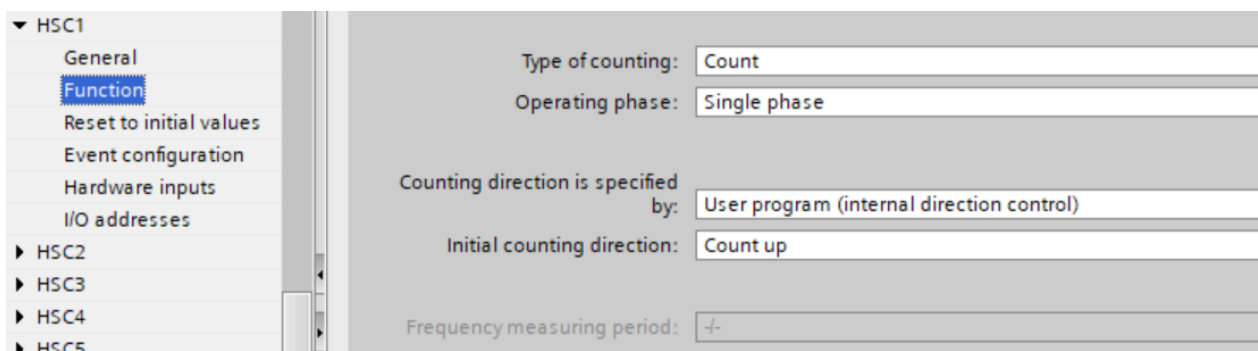
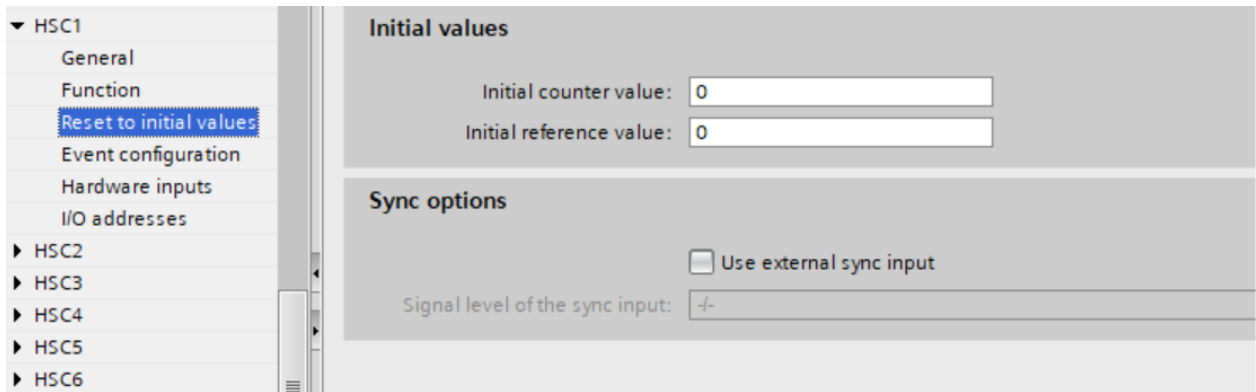


Fig. 19-37

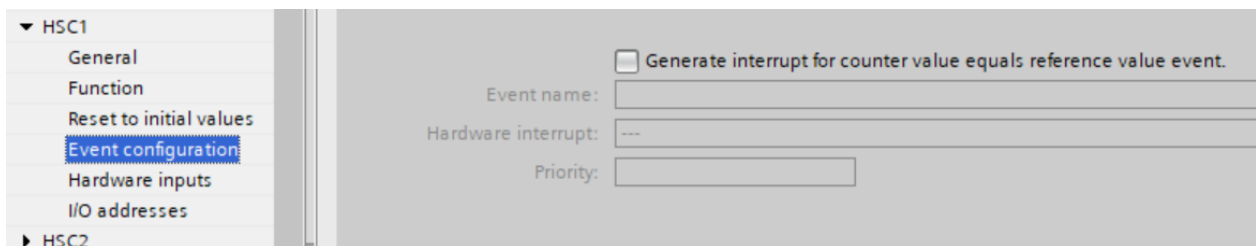
Under the Function tab, choose single phase unless quadrature is to be used:



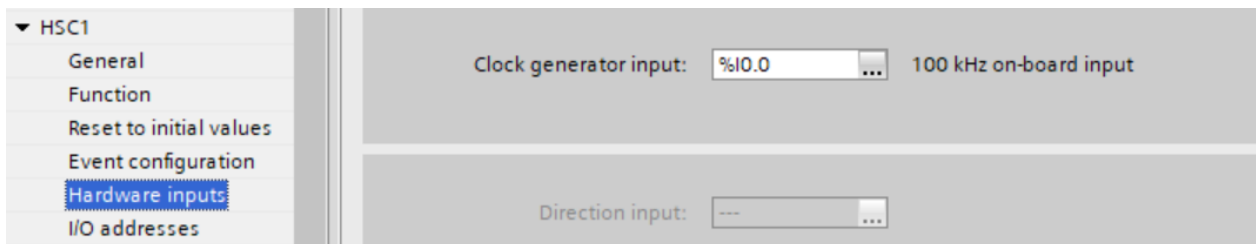
Here, 0 and 0 are fine:



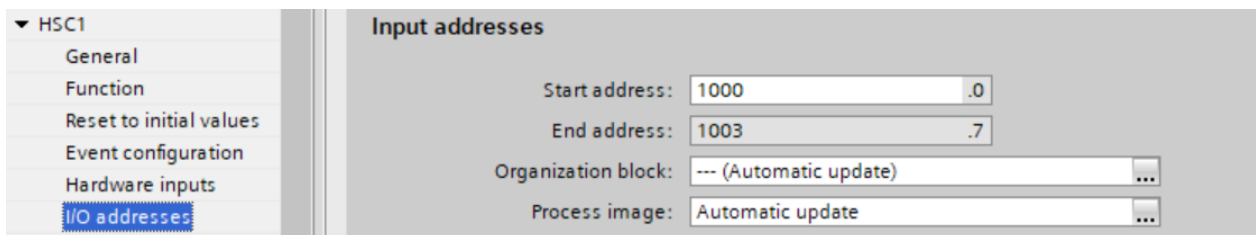
No need to choose an interrupt. The interrupt should be the cyclical interrupt executing the PID function:



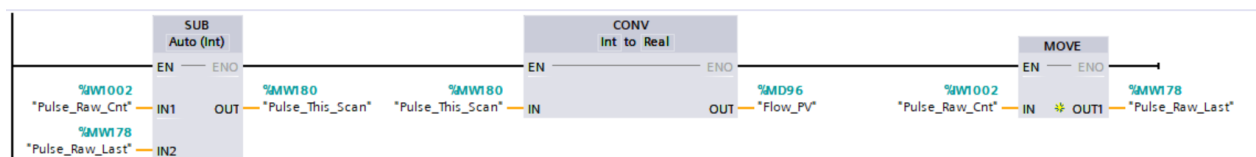
Next, identify the actual input addressed as the hsc (high speed counter) input:



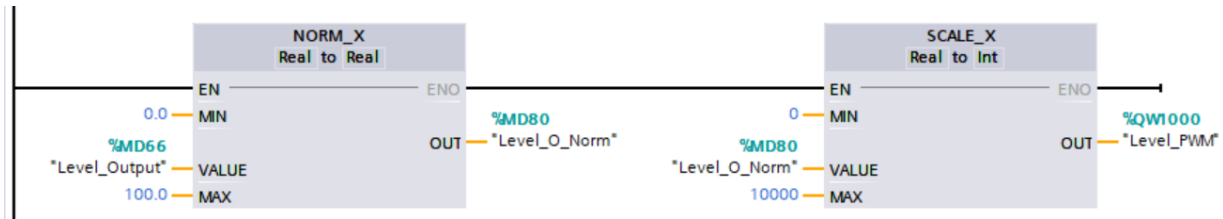
Then, identify the input address ID:1000-1003:



The address of the input used is IW1002. It is used in the following statement as the rolling value of the input count. This logic executes each time period and calculates the pulses in the last scan:



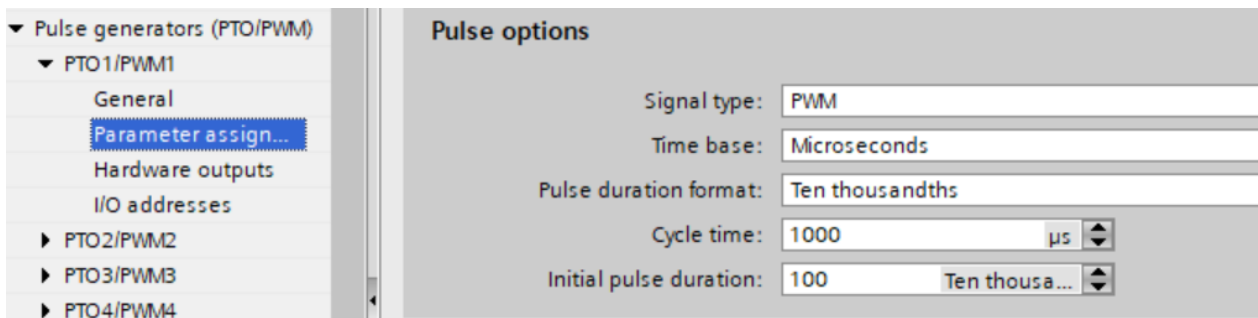
The address of the output is QW1000. It is used in the following statement as the value of the output count.



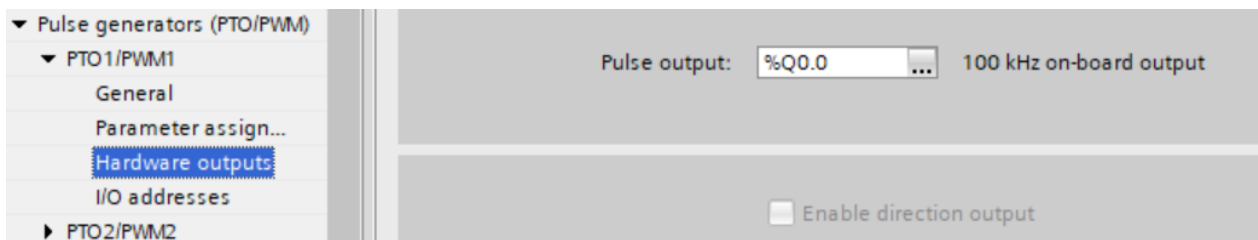
The configuration of the PWM output for control of the bilge pump for the Tank lab as well as the gear motor lab is a single PWM 24 V output that turns on a dc motor controller input:



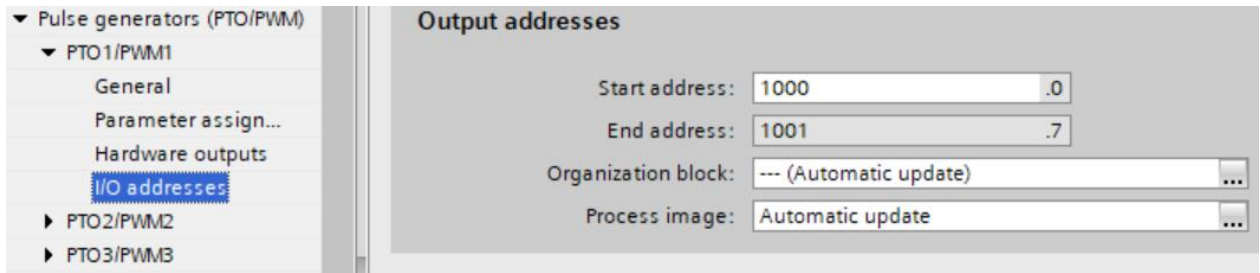
The pulse width modulated output is set up in microseconds. Other constants in the set-up include the overall pulse duration. The pulse length is 10 msec with a base of 10,000 counts:



The following statement identifies the output to be pulse modulated:



The following address gives the output address to load the pwm time into QW1000:



The following views of the output show various PWM settings. The first one is approximately 75% or a value in QW1000 of 7,500:

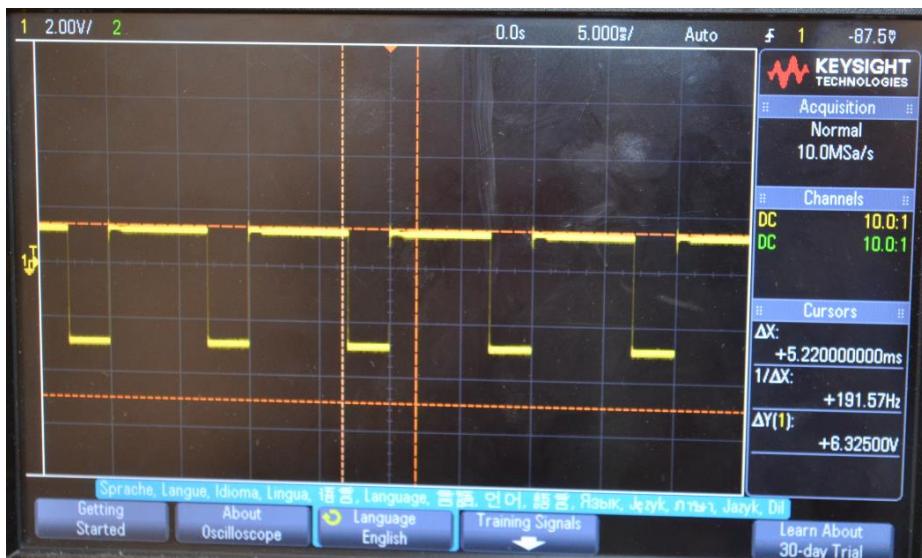


Fig. 19-39

This view shows approximately 90% on:

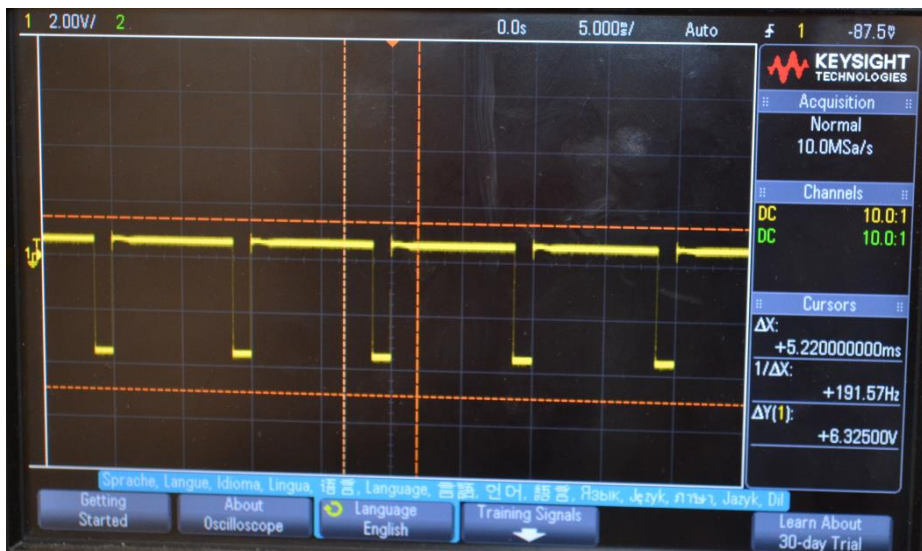


Fig. 19-40

A list of hardware identifiers for the various I/O points is found in the list of system constants under the system constant tab:

General IO tags System constants Texts					
Show hardware system constant					
Name	Type	Hardware identi.	Used by	Comment	
Local-MC	Hw_SubModule	51	PLC_1		
Local-Common	Hw_SubModule	50	PLC_1		
Local-Exec	Hw_SubModule	52	PLC_1		
Local	Hw_SubModule	49	PLC_1		
Local-PROFINET_interface_1	Hw_Interface	64	PLC_1		
Local-AI_2_AQ_2_1	Hw_SubModule	263	PLC_1		
Local-DI_14_DQ_10_1	Hw_SubModule	264	PLC_1		
Local-Pulse_1	Hw_Pwm	257	PLC_1		
Local-Pulse_2	Hw_Pwm	258	PLC_1		
Local-Pulse_3	Hw_Pwm	259	PLC_1		
Local-Pulse_4	Hw_Pwm	260	PLC_1		
Local-HSC_1	Hw_Hsc	261	PLC_1		
Local-HSC_2	Hw_Hsc	262	PLC_1		
Local-HSC_3	Hw_Hsc	265	PLC_1		
Local-HSC_4	Hw_Hsc	266	PLC_1		
Local-HSC_5	Hw_Hsc	267	PLC_1		
Local-HSC_6	Hw_Hsc	268	PLC_1		
Local-PROFINET_interface_1-Port_1	Hw_Interface	65	PLC_1		
Local-PROFINET_interface_1-Port_2	Hw_Interface	66	PLC_1		

The final set-up of the pwm and hsc devices includes a DB for each. This is found in the OB1 code. The hardware identifier is found in this instruction and ties the device to the action:

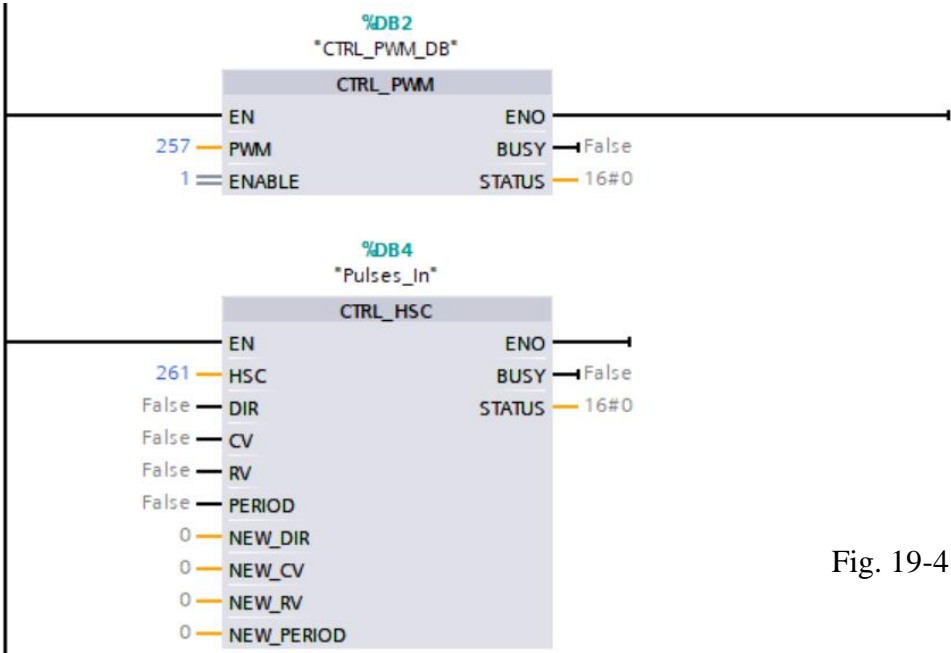


Fig. 19-41

PID control

STEP 7 provides the following PID instructions for the S7-1200 CPU:

The PID_Compact instruction is used to control technical processes with continuous input and output variables. The PID_3Step instruction is used to control motor-actuated devices, such as valves that require discrete signals for open- and close actuation.

Both PID instructions (PID_3Step and PID_Compact) can calculate the P-, I-, and D components during startup (if configured for "pretuning"). You can also configure the instruction for "fine tuning" to allow you to optimize the parameters. You do not need to manually determine the parameters.

Note: Execute the PID instruction at constant intervals of the sampling time (preferably in a cyclic OB). Because the PID loop needs a certain time to respond to changes of the control value, do not calculate the output value in every cycle. Do not execute the PID instruction in the main program cycle OB (such as OB 1).

The sampling time of the PID algorithm represents the time between two calculations of the output value (control value). The output value is calculated during self-tuning and rounded to a multiple of the cycle time. All other functions of PID instruction are executed at every call.

The PID (Proportional/Integral/Derivative) controller measures the time interval between two calls and then evaluates the results for monitoring the sampling time. A mean value of the sampling time is generated at each mode changeover and during initial startup. This value is used as reference for the monitoring function and is used for calculation. Monitoring includes the current measuring time between two calls and the mean value of the defined controller sampling time.

The output value for the PID controller consists of three components:

P (proportional): When calculated with the "P" component, the output value is proportional to the difference between the setpoint and the process value (input value).

I (integral): When calculated with the "I" component, the output value increases in proportion to the duration of the difference between the setpoint and the process value (input value) to finally correct the difference.

D (derivative): When calculated with the "D" component, the output value increases as a function of the increasing rate of change of the difference between the setpoint and the process value (input value). The output value is corrected to the setpoint as quickly as possible.

Link to S7-1200/1500 PID Manual:

<https://support.industry.siemens.com/cs/us/en/view/108210036>

The tuning rules are found on pgs. 265 - 266 under the descriptions of operating modes "Pretuning" and "Fine tuning" in the 1200.

The PID controller uses the following formula to calculate the output value for the PID_Compact instruction.

$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_i \cdot s} (w - x) + \frac{T_d \cdot s}{a \cdot T_d \cdot s + 1} (c \cdot w - x) \right]$$

y	Output value	x	Process value
w	Setpoint value	s	Laplace operator
K _p	Proportional gain (P component)	a	Derivative delay coefficient (D component)
T _i	Integral action time (I component)	b	Proportional action weighting (P component)
T _d	Derivative action time (D component)	c	Derivative action weighting (D component)

The PID controller uses the following formula to calculate the output value for the PID_3Step instruction.

$$\Delta y = K_p \cdot s \cdot \left[(b \cdot w - x) + \frac{1}{T_i \cdot s} (w - x) + \frac{T_d \cdot s}{a \cdot T_d \cdot s + 1} (c \cdot w - x) \right]$$

y	Output value	x	Process value
w	Setpoint value	s	Laplace operator
K _p	Proportional gain (P component)	a	Derivative delay coefficient (D component)
T _i	Integral action time (I component)	b	Proportional action weighting (P component)
T _d	Derivative action time (D component)	c	Derivative action weighting (D component)

To set up a PID block in your program, choose 'Technology' from Instructions and then 'PID Compact'. See below:

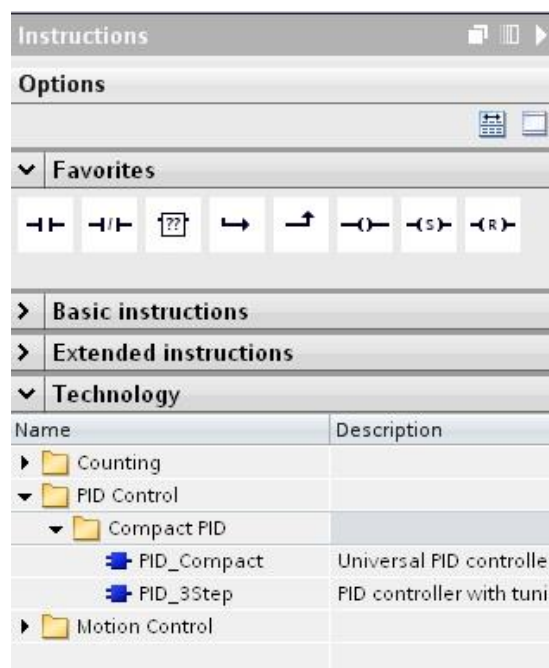
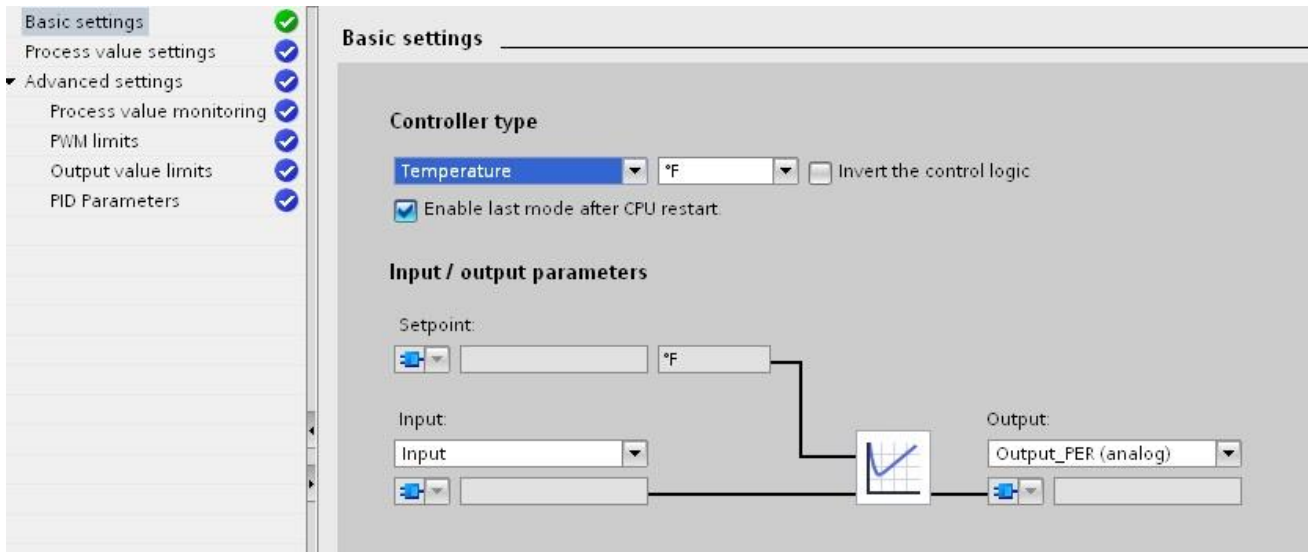


Fig. 19-42

The settings for the controller may be reached by clicking the icon in the upper right of the PID block. The block should also be placed in a Timed Interrupt OB:



An example from the Ball in Tube program is included in the following explanation. The second PID program developed is the Tank over Tank.

Inserting the PID instruction and technological object

STEP 7 provides two instructions for PID control. Use the PID_Compact instruction for the lab in this course, please!

The PID_Compact instruction and its associated technological object provide a universal PID controller with tuning. The technological object contains all of the settings for the control loop.

The PID_3Step instruction and its associated technological object provide a PID controller with specific settings for motor-activated valves. The technological object contains all of the settings for the control loop. The PID_3Step controller provides two additional Boolean outputs.

After creating the technological object, you must configure the parameters. You also adjust the autotuning parameters ("pretuning" during startup or manual "fine tuning") to commission the operation of the PID controller.

LAD / FBD	SCL	Description
	<pre>"PID_Compact_1" (Setpoint:= real in_, Input:= real in_, Input_PER:= word in_, ManualEnable:= bool in_, ManualValue:= real in_, Reset:= bool in_, ScaledInput=> real out_, Output=> real out_, Output_PER=> word out_, Output_PWM=> bool out_, SetpointLimit H=> bool out_, SetpointLimit L=> bool out_, InputWarning H=> bool out_, InputWarning L=> bool out_, State=> int out_, Error=> dword out_);</pre>	<p>PID_Compact provides a PID controller with self-tuning for automatic and manual mode. PID_Compact is a PIDT1 controller with anti-windup and weighting of the P- and D-component.</p>

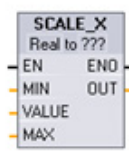
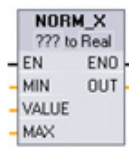
Fig. 19-43

¹ STEP 7 automatically creates the technological object and instance DB when you insert the instruction. The instance DB contains the parameters of the technological object.

² In the SCL example, "PID_Compact_1" is the name of the instance DB.

When programming the inputs and outputs, the following two instructions are used to scale and normalize the analog value. Use the NORM_X function first to convert the number to a real in the range 0-1 and then use SCALE_X to scale the normalized value to a range for the real value.

Table 6-6 SCALE_X and NORM_X instructions

LAD / FBD	SCL	Description
	<pre> out := SCALE_X(min := _undef_in_, value := _real_in_, max := _undef_in_); </pre>	<p>Scales the normalized real parameter VALUE where (0.0 <= VALUE <= 1.0) in the data type and value range specified by the MIN and MAX parameters:</p> $OUT = VALUE (MAX - MIN) + MIN$
	<pre> out := NORM_X(min := _undef_in_, value := _undef_in_, max := _undef_in_); </pre>	<p>Normalizes the parameter VALUE inside the value range specified by the MIN and MAX parameters:</p> $OUT = (VALUE - MIN) / (MAX - MIN),$ <p>where (0.0 <= OUT <= 1.0)</p>

¹ Equivalent SCL: $out := value (max-min) + min;$ ²Equivalent SCL: $out := (value-min) / (max-min);$

Descriptions of various parameters in the PID block are found below:

Parameter and type		Data type	Description
Setpoint	IN	Real	Setpoint of the PID controller in automatic mode. Default value: 0.0
Input	IN	Real	Process value. Default value: 0.0 You must also set Config.InputPEROn = FALSE.
Input_PER	IN	Word	Analog process value (optional). Default value: W#16#0 You must also set Config.InputPEROn = TRUE.
ManualEnable	IN	Bool	Enables or disables the manual operation mode. Default value: FALSE <ul style="list-style-type: none"> On the edge of the change from FALSE to TRUE, the PID controller switches to manual mode, State = 4, and Retain.Mode remains unchanged. On the edge of the change from TRUE to FALSE, the PID controller switches to the last active operating mode and State = Retain.Mode.
ManualUP	IN	Bool	In manual mode, every rising edge opens the valve by 5% of the total actuating range, or for the duration of the minimum motor actuation time. ManualUP is evaluated only if you are using OutputPer and if position feedback is available. Default value: FALSE <ul style="list-style-type: none"> If Output_PER is FALSE, the manual input turns Output_UP on for the time that corresponds to a movement of 5% of the device. If Config.ActuatorEndStopOn is TRUE, then Output_UP does not come on if Actuator_H is TRUE.
ManualDN	IN	Bool	In manual mode, every rising edge closes the valve by 5% of the total actuating range, or for the duration of the minimum motor actuation time. ManualDN is evaluated only if you are using OutputPer and if position feedback is available. Default value: FALSE <ul style="list-style-type: none"> If Output_PER is FALSE, the manual input turns Output_DN on for the time that corresponds to a movement of 5% of the device. If Config.ActuatorEndStopOn is TRUE, then Output_DN does not turn on if Actuator_L is TRUE.
ManualValue	IN	Real	Process value for manual operation. Default value: 0.0 In manual mode, you specify the absolute position of the valve. ManualValue is evaluated only if you are using OutputPer, or if position feedback is available. Default value: 0.0
Feedback	IN	Real	Position feedback of the valve. Default value: 0.0 To use Feedback, then set Config.FeedbackPerOn = FALSE.

The values in the table above are necessary to make the PID block work correctly. Some may be set once and not included in the program as variables. Others must be included as programmed variables. For example, if Input_PER is used, this input must be represented as a percent from 0 to 100.0. This value is the value fed to the PID block from the analog process variable, in this case the laser. The variable must be represented in Input_PER as a ratio from 0 to 100.

Other variables in the table above are useful when coordinating with the faceplate. For example, if the PID algorithm is set to manual, the ManualValue variable must be set to the desired state of the output of the PID. The variable is moved to this location and the output is set to this value.

Parameter and type		Data type	Description
Actuator_L	IN	Bool	If Actuator_L = TRUE, the valve is at the lower end stop and is no longer moved in this direction. Default value: FALSE
Reset	IN	Bool	Restarts the PID controller. Default value: FALSE If Reset = TRUE: <ul style="list-style-type: none"> • "Inactive" operating mode • Input value = 0 • Interim values of the controller are reset. (PID parameters are retained.)
ScaledInput	OUT	Real	Scaled process value
ScaledFeedback	OUT	Real	Scaled valve position
Output_PER	OUT	Word	Analog output value. If Config.OutputPerOn = TRUE, then Output_PER is evaluated.
Output_UP	OUT	Bool	Digital output value for opening the valve. Default value: FALSE If Config.OutputPerOn = FALSE, then parameter Output_UP is evaluated.
Output_DN	OUT	Bool	Digital output value for closing the valve. Default value: FALSE If Config.OutputPerOn = FALSE, then parameter Output_DN is evaluated.
SetpointLimitH	OUT	Bool	Setpoint high limit. Default value: FALSE If SetpointLimitH = TRUE, the absolute upper limit of the setpoint is reached. In the CPU, the setpoint is limited to the configured absolute upper limit of the actual value.
SetpointLimitL	OUT	Bool	Setpoint low limit. Default value: FALSE If SetpointLimitL = TRUE, the absolute lower limit of the setpoint is reached. In the CPU the setpoint is limited to the configured absolute lower limit of the actual value.
InputWarningH	OUT	Bool	If InputWarningH = TRUE, the input value has reached or exceeded the upper warning limit. Default value: FALSE
InputWarningL	OUT	Bool	If InputWarningL = TRUE, the input value has reached or exceeded the lower warning limit. Default value: FALSE
State	OUT	Int	Current operating mode of the PID controller. Default value: 0 Use Retain.Mode to change the operating mode: <ul style="list-style-type: none"> • State = 0: Inactive • State = 1: Pretuning • State = 2: Manual fine tuning • State = 3: Automatic mode • State = 4: Manual mode • State = 5: Safety mode • State = 6: Output value measurement • State = 7: Safety mode monitoring with active trigger • State = 8: Inactive mode monitoring with active trigger
Error	OUT	Bool	If Error = TRUE, at least one error message is pending. Default value: FALSE
ErrorBits	OUT	DWord	Error message. Default value: DW#16#0000 (no error)

Likewise, these variables contain information to allow the PID algorithm to function properly. The state is a number from 0 to 8. We only use the values of 3 and 4 for the application given in the Ball-in-Tube program.

The I/O address of the analog input point is shown in the analog input addresses of the base processor unit. If additional analog points beyond two or if these points need a floating neutral, then an additional analog input card is needed. In our example for the ball-in-tube lab, the input addresses start at I:64. The first address is bytes I:64 and 65. The second input address is bytes I:66 and 67

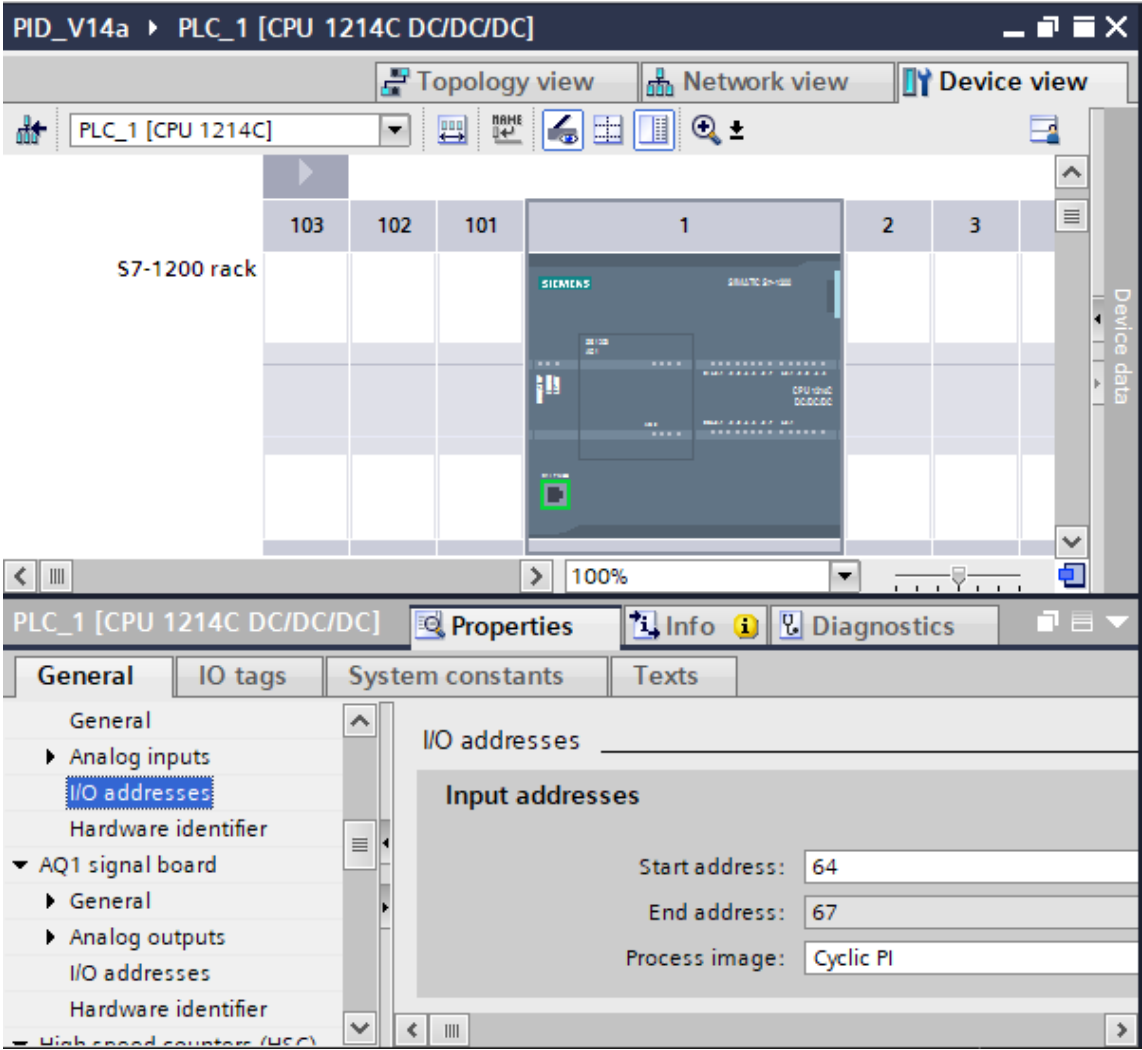


Fig. 19-44 Addresses of the Analog Inputs

Display of the analog points is done on a historical data plot shown below.

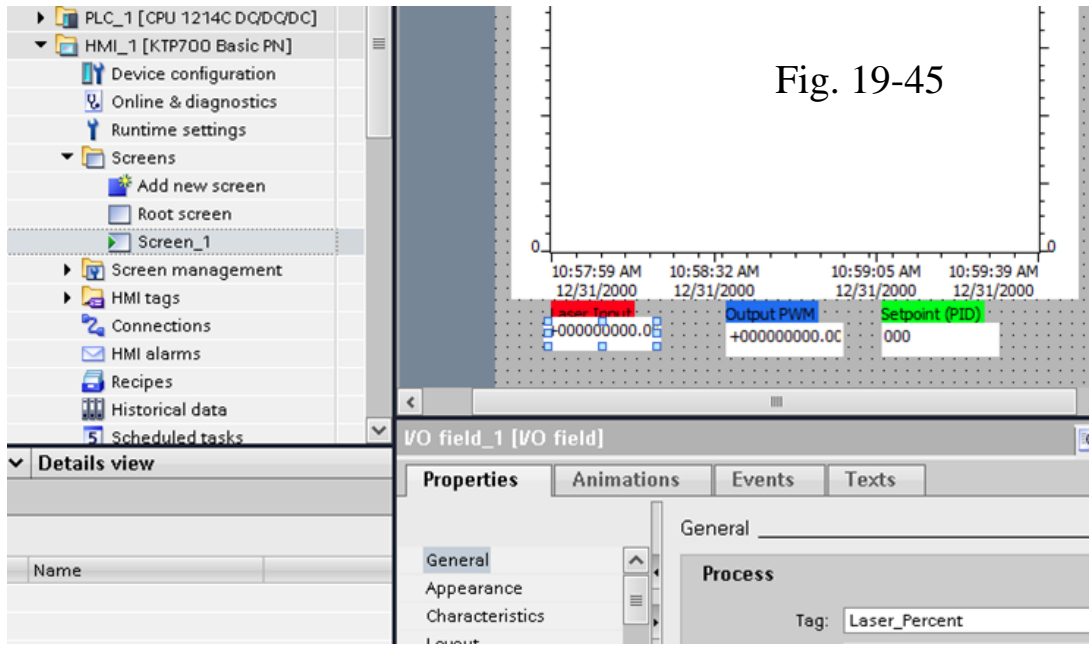
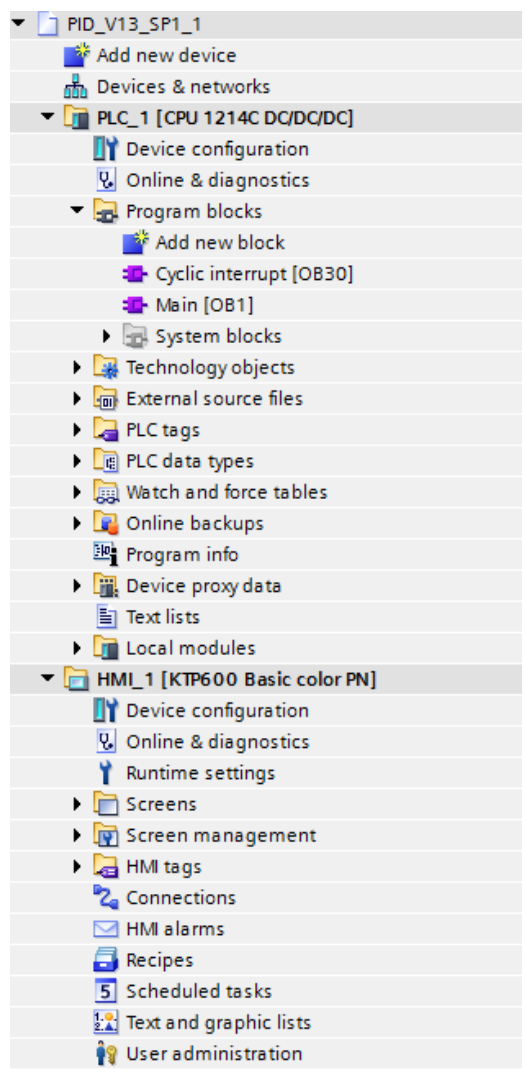
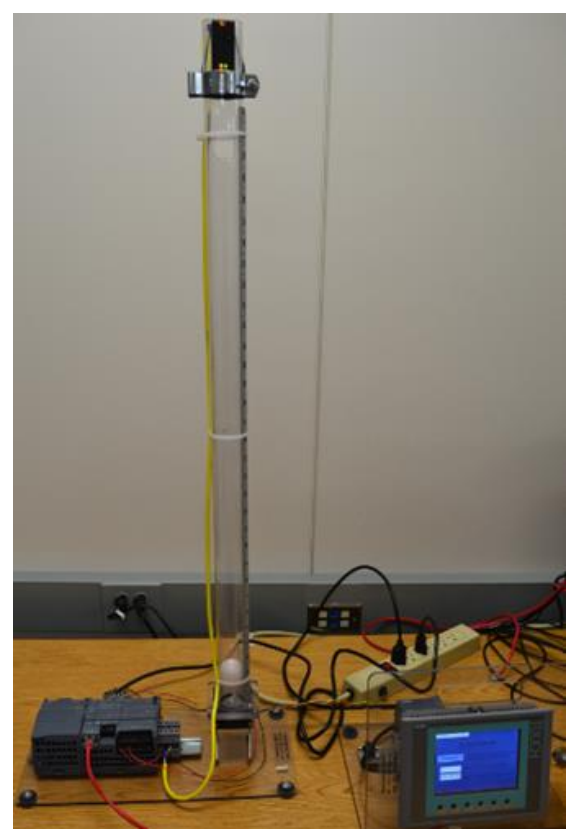


Fig. 19-45



Project Tree for Ball-in-Tube



Physical Layout of Ball-in-Tube

Fig. 19-46

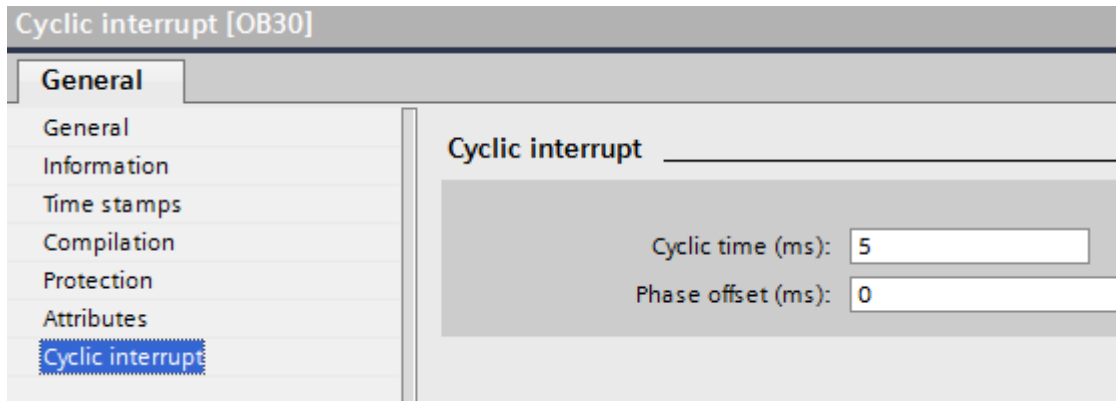


Fig. 19-47 Setting up the Cyclic Interrupt (OB30)

A separate Cyclic Interrupt Program must be built to provide execution of the PID Block. The PID program executes the PID algorithm after reading the Process Variable input. After execution of the algorithm, the PWM output determines the state of the output to the fan.

The PID algorithm for the Ball-in-Tube program is shown below. The instruction is configured and set up from programming statements as well as constants entered into the tables above.

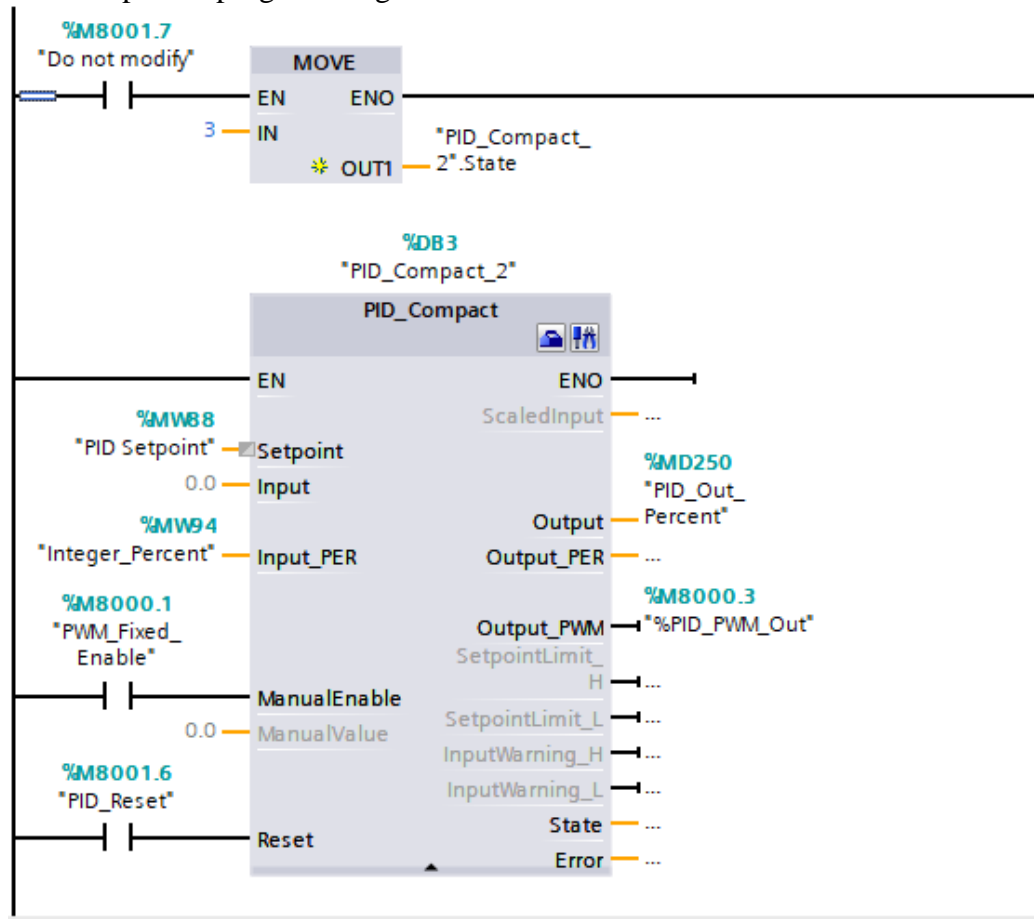


Fig. 19-48 The PID Block for the Ball-in-Tube Lab

The HMI panel below has a button to choose between auto and manual. In PID_PWM, the button is in automatic. When in auto, the setpoint is entered on a separate page. The manual value for the PID output may be entered below the button in manual mode.

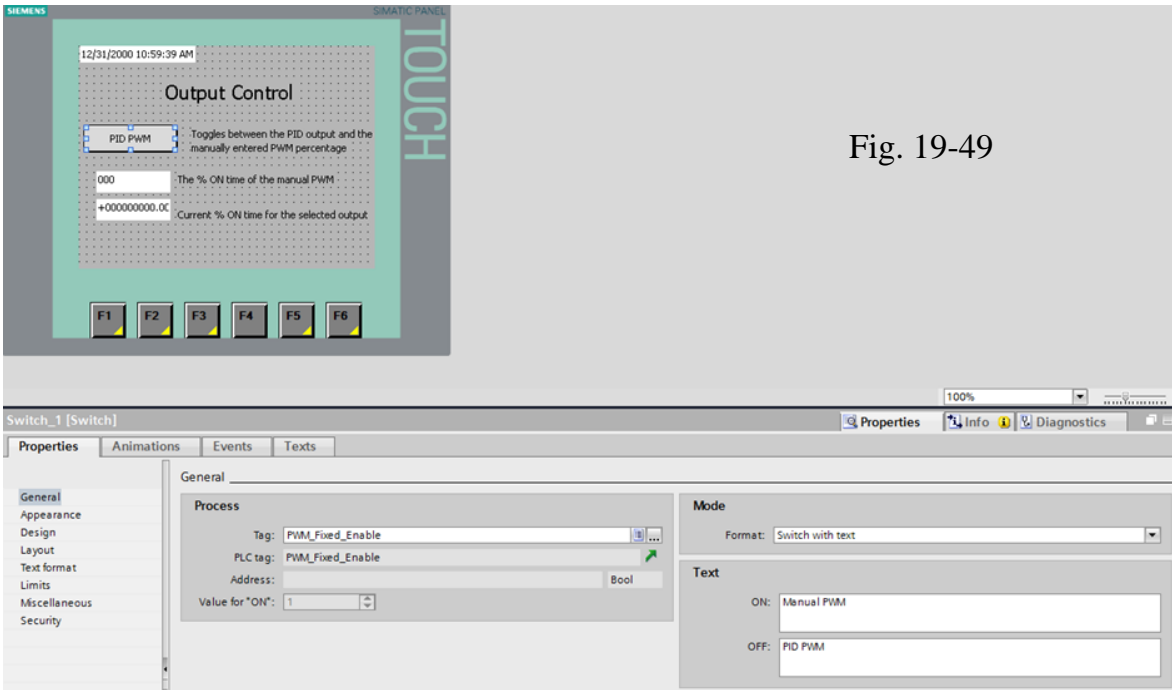


Fig. 19-49

Below the button is a data entry window for the value of percent on time for the fan. In this window is the percent on time for the output.

The Configuration editor for PID_Compact shows the following screen. Here, the user selects the units such as temperature or pressure. The user also determines whether variables such as the PV are Input or Input_Per. Most users would select 'general' for controller type.

Use the commissioning editor to configure the controller for auto-tuning at startup and for auto-tuning during operation. To open the commissioning editor, click the icon on either the instruction or the project navigator.

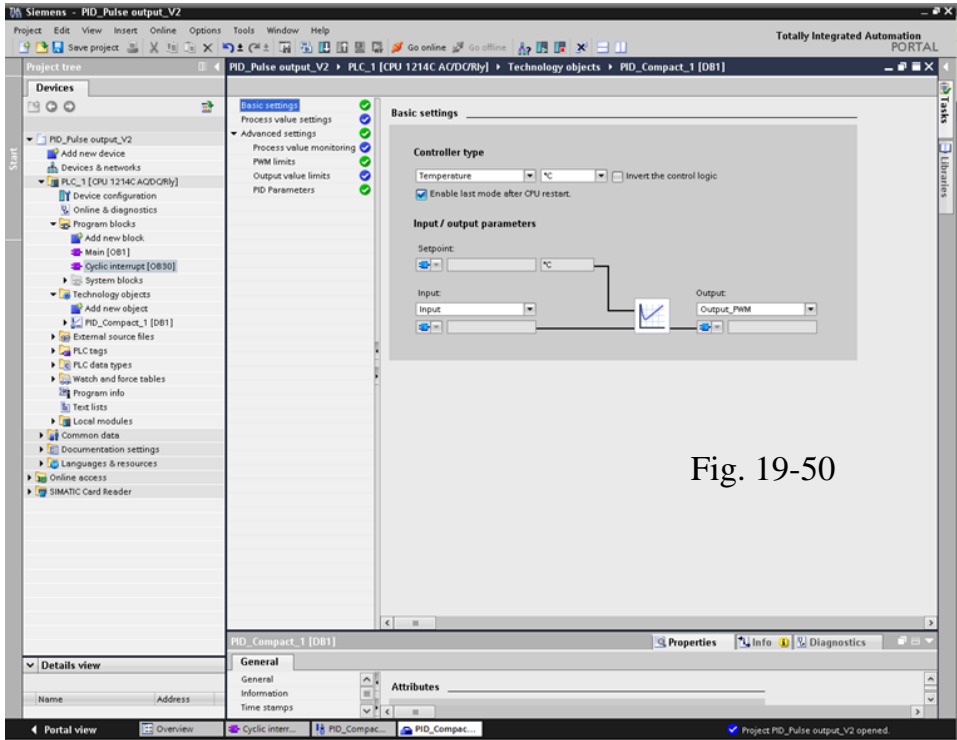


Fig. 19-50

The following table lists some common suggested actions for assisting the set-up of the PID controller:

Sample configuration settings for the PID_Compact instruction

Settings		Description
Basic	Controller type	Selects the engineering units.
	Invert the control logic	Allows selection of a reverse-acting PID loop. <ul style="list-style-type: none"> If not selected, the PID loop is in direct-acting mode and the output of PID loop increases if input value < setpoint. If selected, the output of the PID loop increases if the input value > setpoint.
	Enable last mode after CPU restart	Restarts the PID loop after it is reset or if an input limit has been exceeded and returned to the valid range.
	Input	Selects either the Input parameter or the Input_PER parameter (for analog) for the process value. Input_PER can come directly from an analog input module.
	Output	Selects either the Output parameter or the Output_PER parameter (for analog) for the output value. Output_PER can go directly to an analog output module.
Process value	Scales both the range and the limits for the process value. If the process value goes below the low limit or above the high limit, the PID loop goes to inactive mode and sets the output value to 0. To use Input_PER, you must scale the analog process value (input value).	

Allen-Bradley Analog Inputs and Outputs

Wiring diagrams for the card as well as the engineering range of the input and output channels are found on the next two pages.

1769-IF4XOF2/A
Terminal Door Label

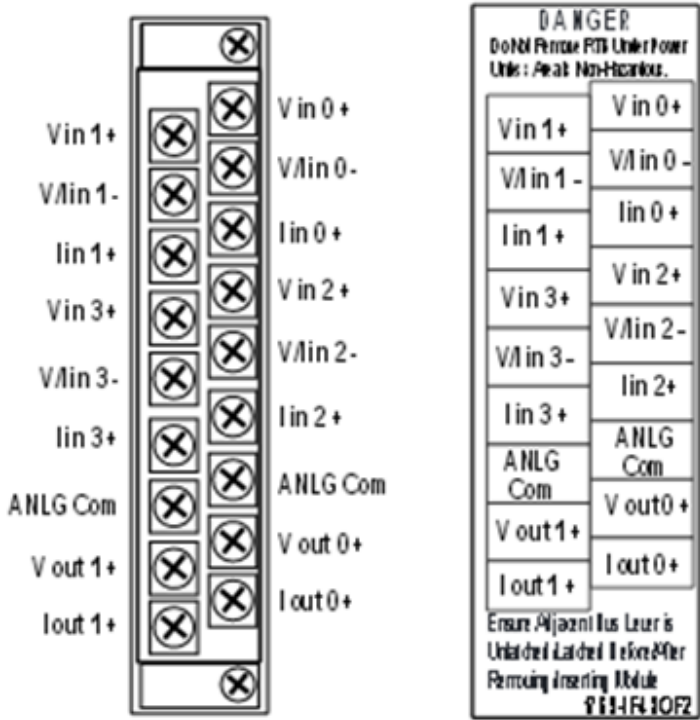


Fig. 19-51 1769-IF4XOF2/A and F2F/A Analog Card

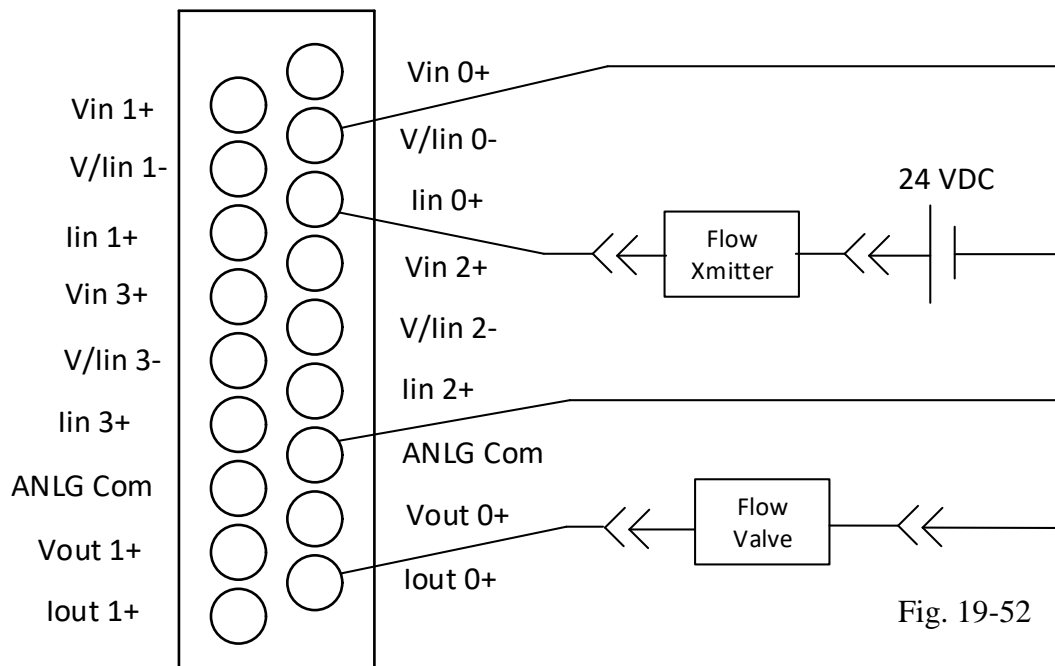


Fig. 19-52

The wiring diagram of the card is shown above. The input and output range of the 4-20 mA engineering units can be found by looking up the accuracy of the signals. Both have a range of 0 mA to 21 mA – 0 to 32640 decimal range. So, 4 mA would be $6217 (32640/21)*4$ and 20 mA would be 31085. Our range for the raw input and output then is 6217 – 31085.

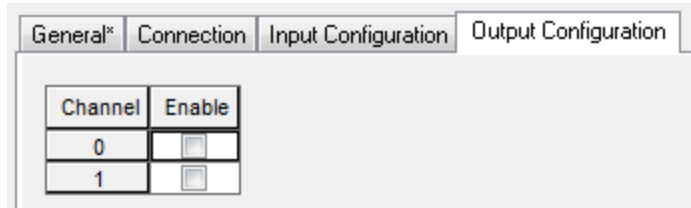
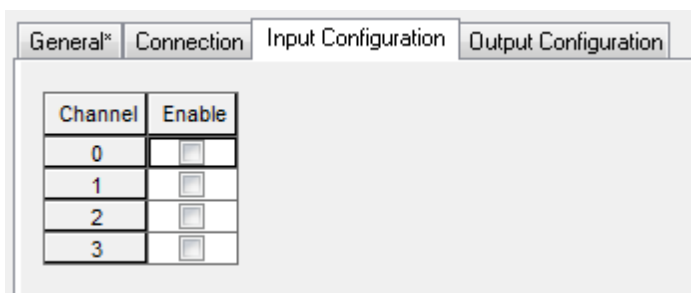
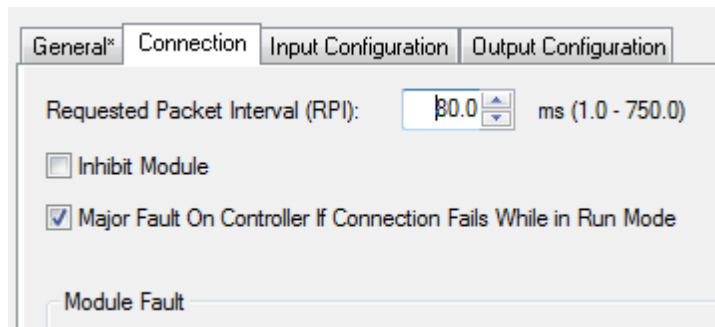


Fig. 19-53

Using the CompactLogix PID Block with RSView ME

The PID algorithm will be introduced in an application using the CompactLogix hardware and software to provide control of the same valve used in the SLC programming experiences. The graphical operator interface will be upgraded to the newer RSView ME operator interface.

Configure a PID Instruction

After you enter the PID instruction and specify the PID structure, you use the configuration tabs to specify how the PID instruction should function.

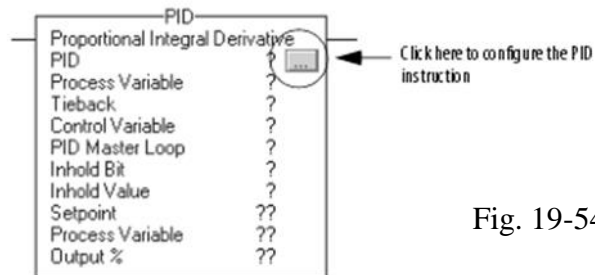


Fig. 19-54

Inclusion of the data tag to create the list shown above. The PID algorithm uses these data tags to calculate and control a PID block. For instance, the PV value for the block is mypid.PV. The SP or setpoint is mypid.SP. The example screens that follow show the newer IF4XOF2F/A card and are used to set up the scaling for the present system in the lab.

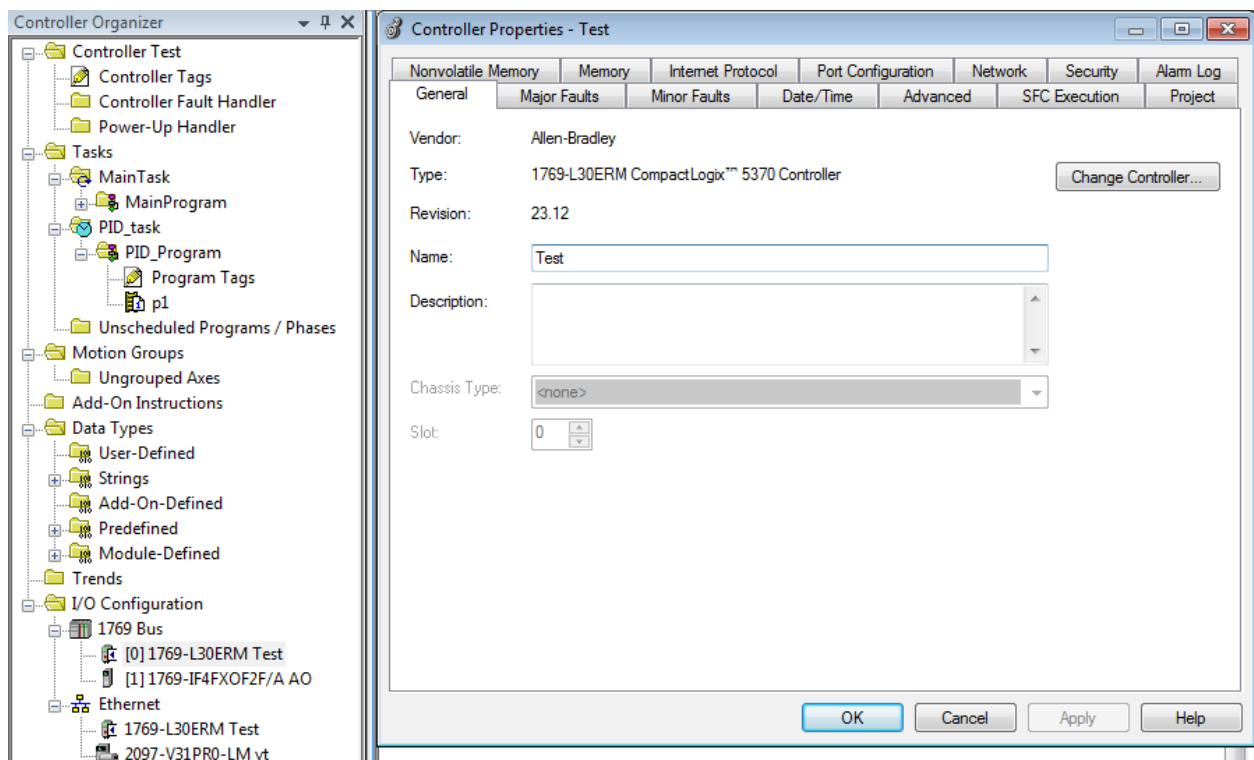


Fig. 19-55 Controller Configuration of the L30ERM

The task was set up to execute every 100 msec. This is shown in the figure below:

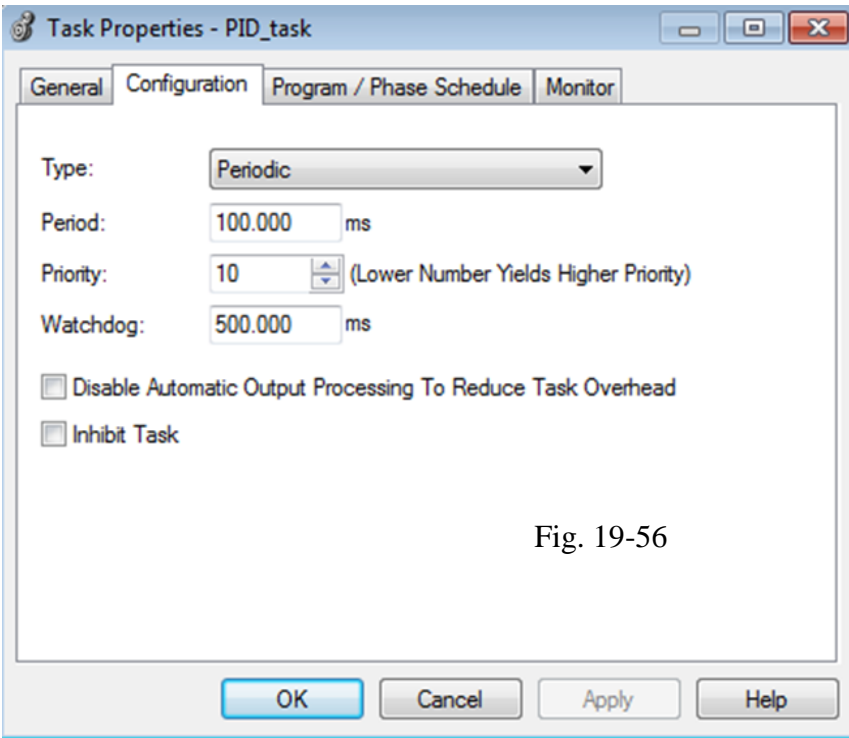


Fig. 19-56

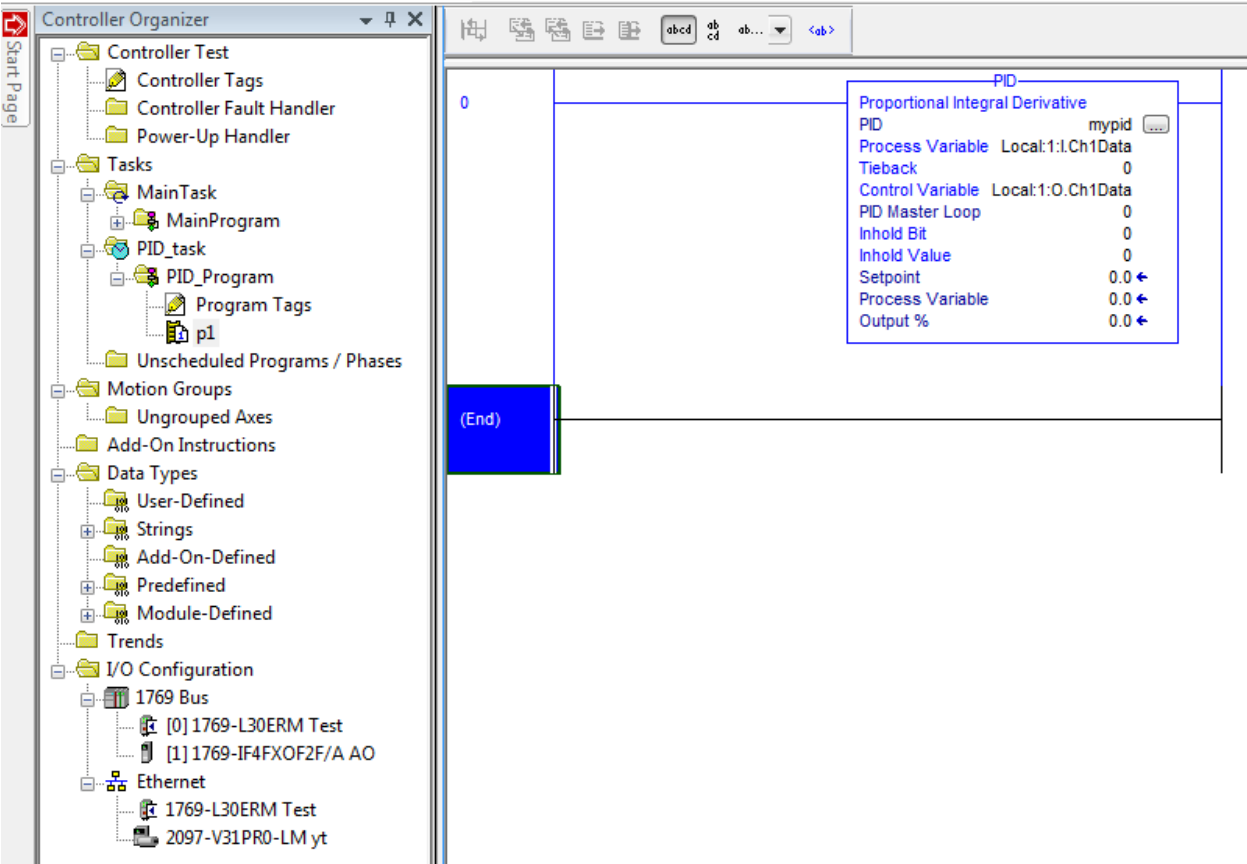


Fig. 19-57 PID Module Set in Periodic Task

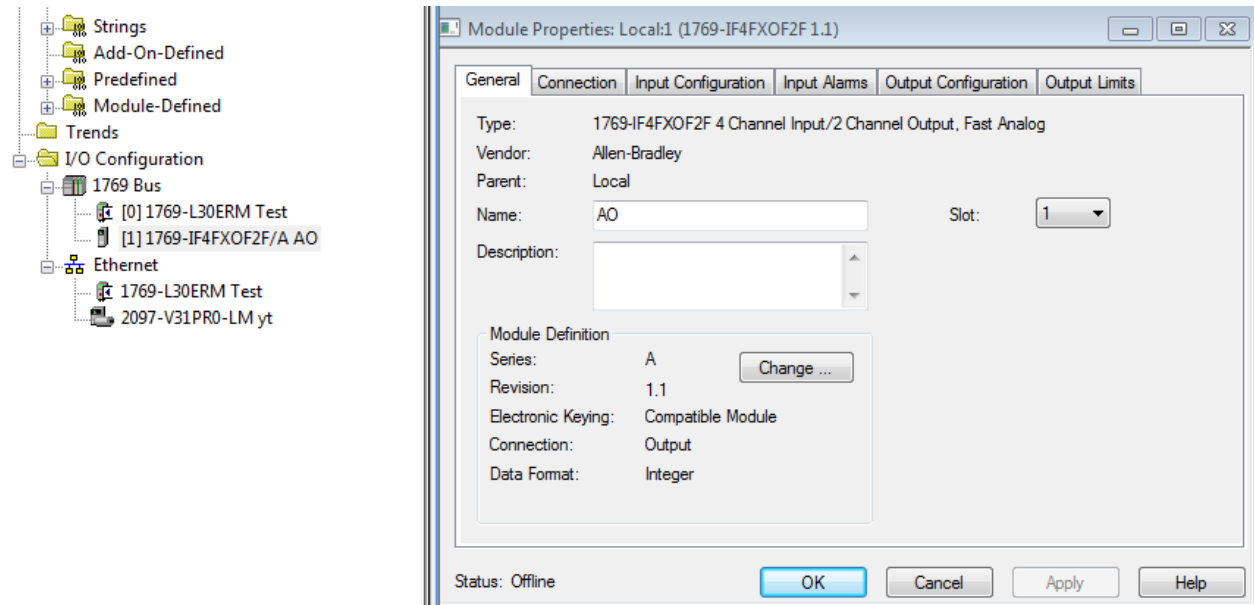


Fig. 19-58 Configuration of the PID Module Set in Periodic Task

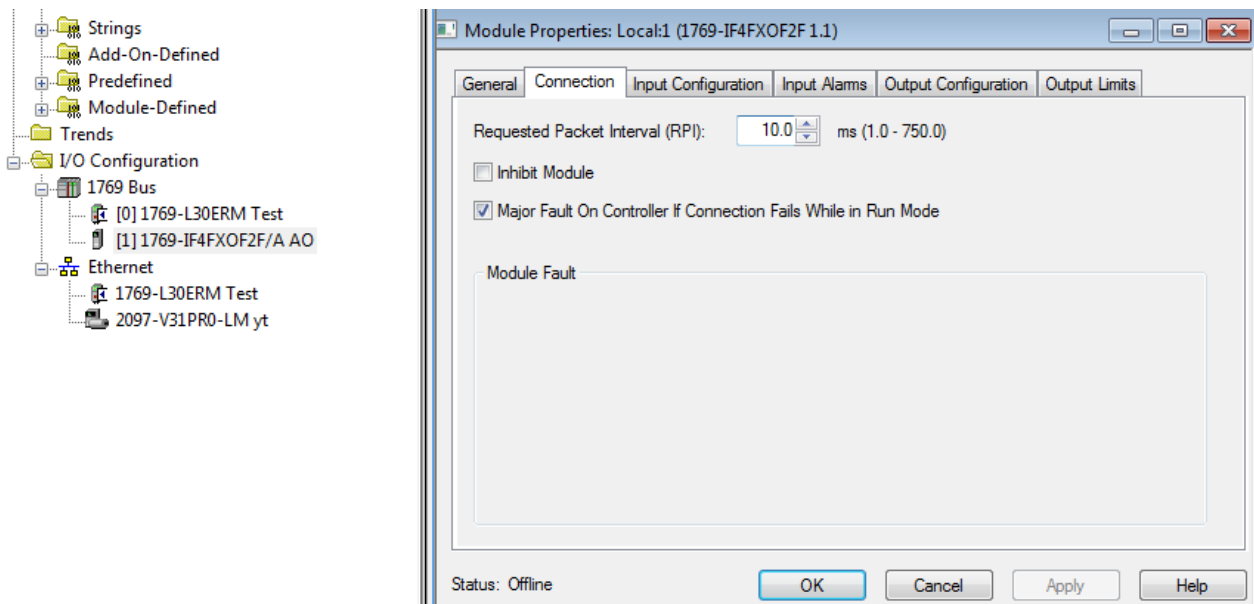


Fig. 19-59

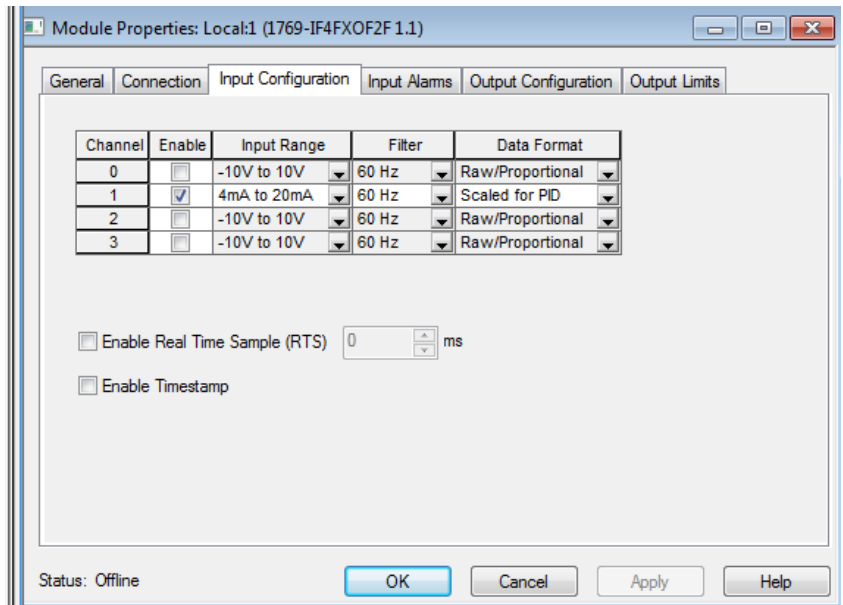
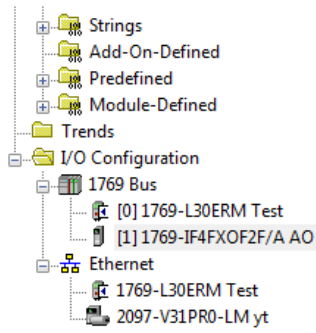


Fig. 19-60

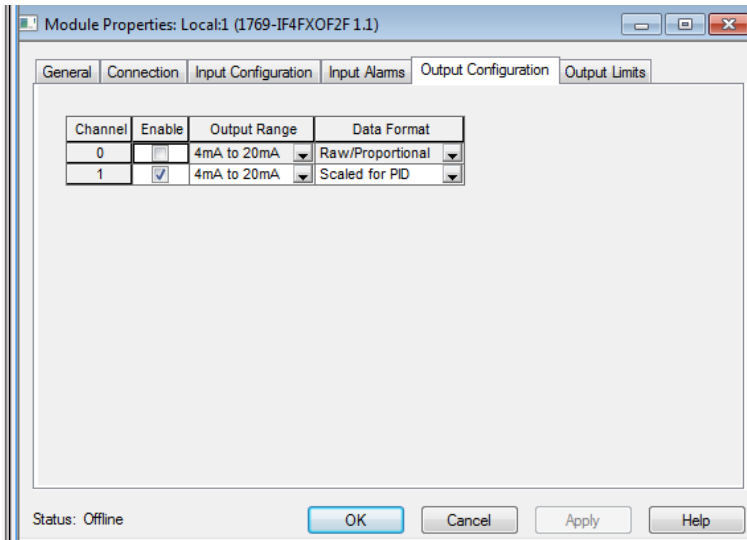
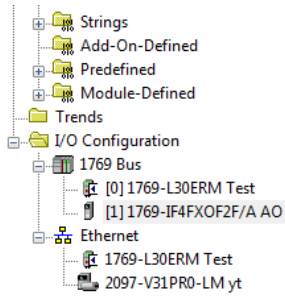


Fig. 19-61

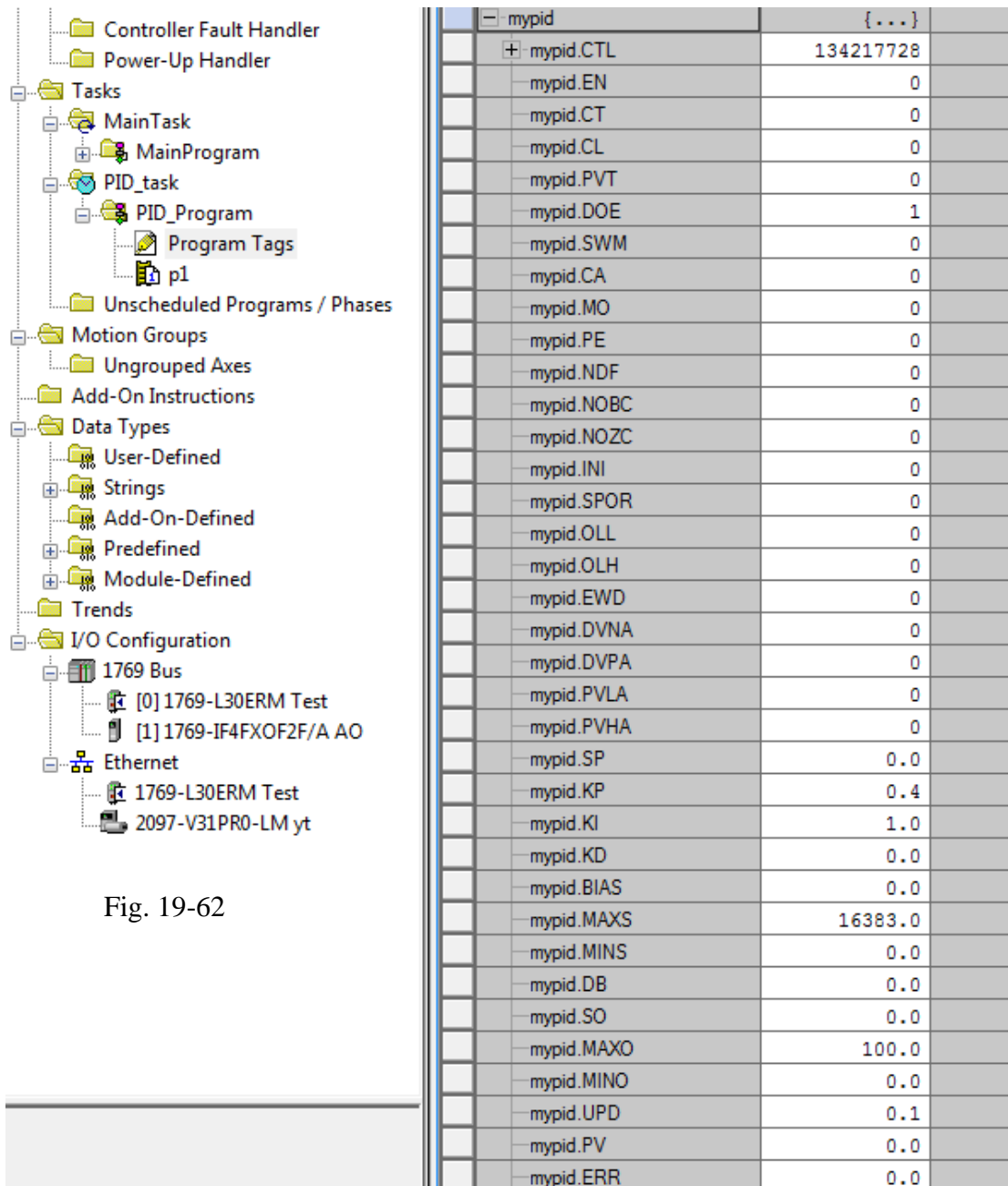


Fig. 19-62

The Program Tags for the PID mypid are shown with variable contents. These variables are useful as tag references used for communicating with the variables through program control.

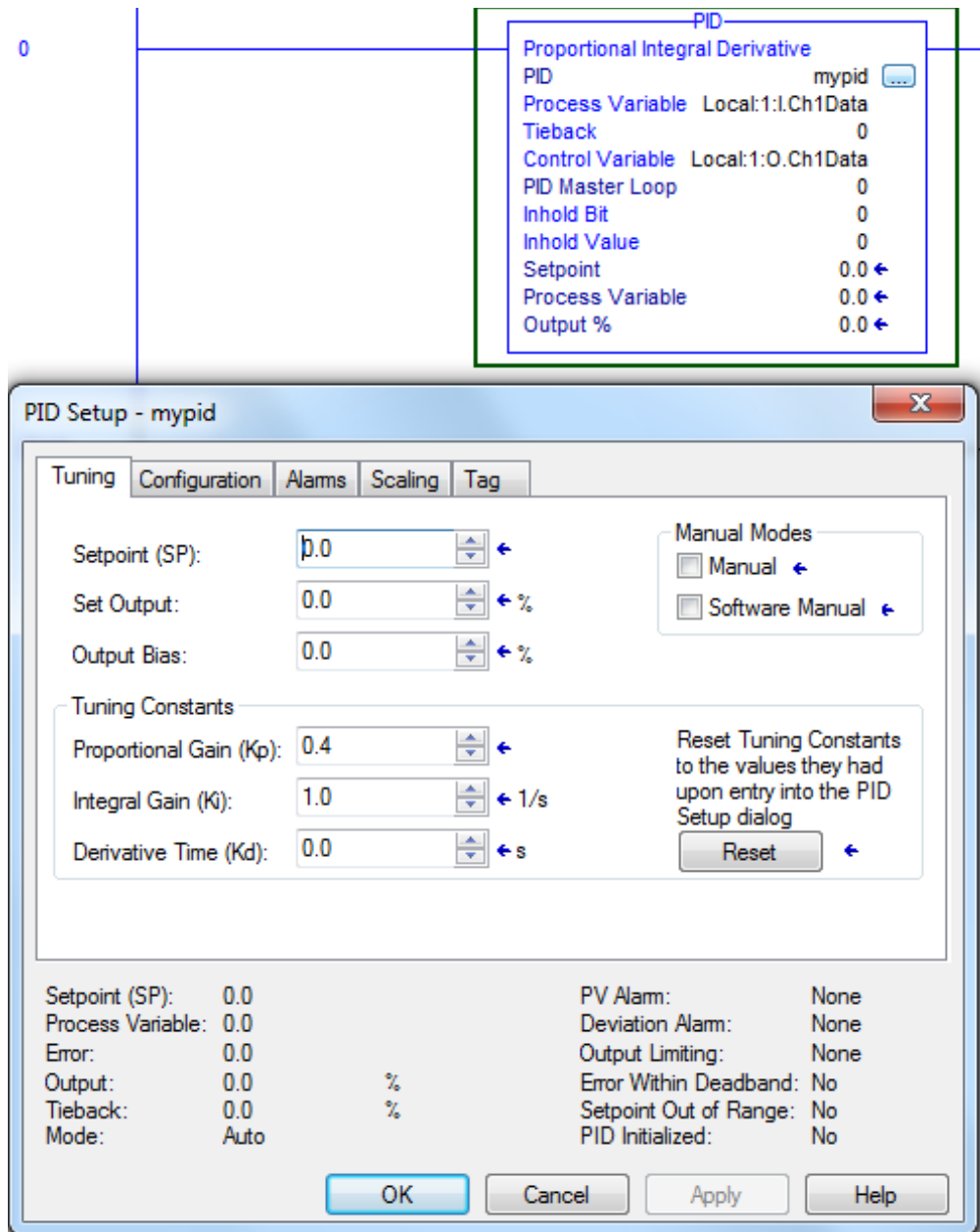


Fig. 19-63 PID Tag Setup-Tuning

The tuning tab shows the variables used to tune the PID block. The Kp, Ki and Kd tuning constants are probably the best variables for the water valve. These constants should not vary too much from the numbers shown or the PID block may become unstable.

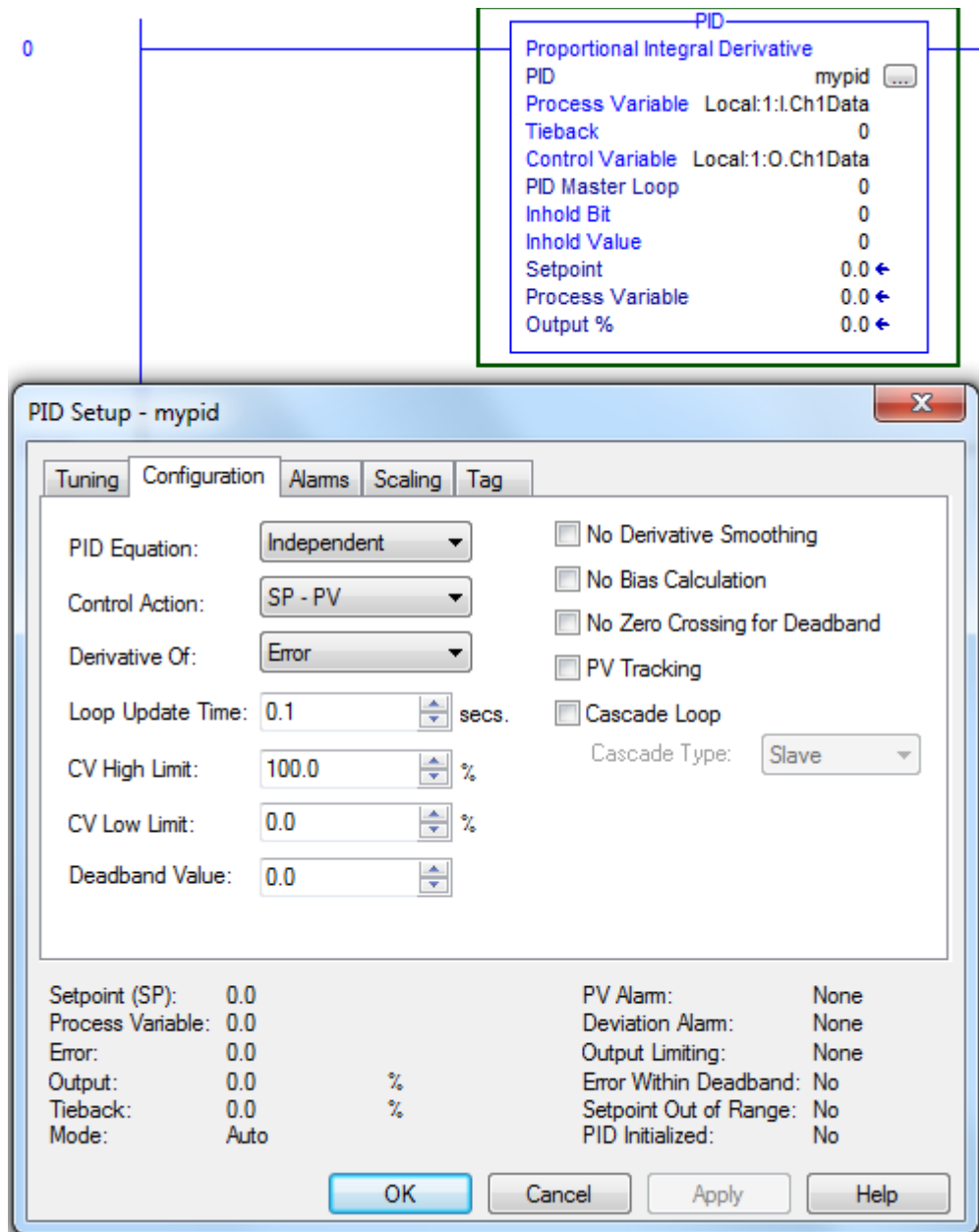


Fig. 19-64 PID Configuration

The configuration tab shows the variables used to set up the type of block used. The variables seen above are the ones used in the download example. There are a number of variables that are not used.

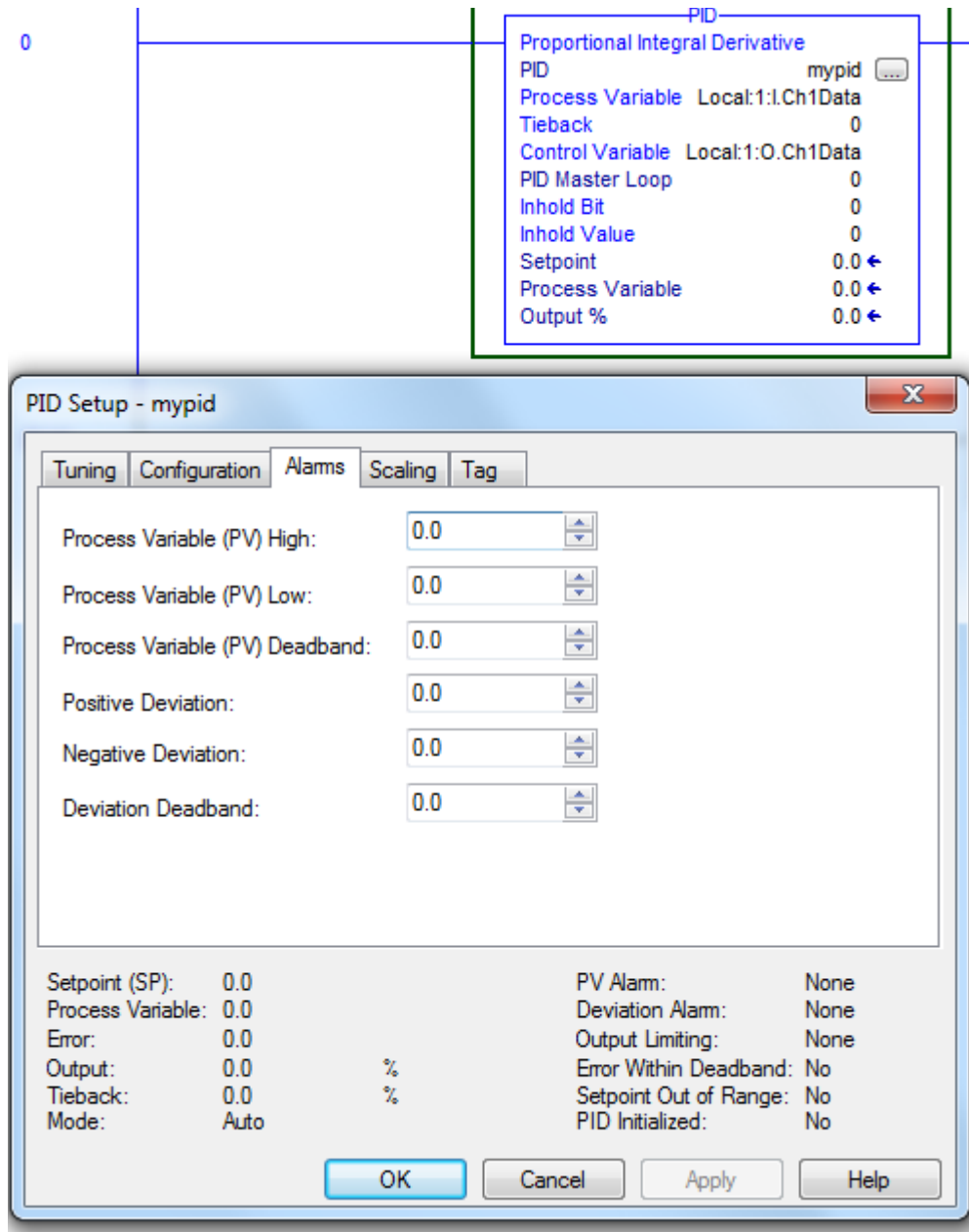


Fig. 19-65 PID Alarms

The alarms tab shows the alarm variables used to set up the block. The alarm limits are ignored for now but in a real application will be necessary when setting up a system of alarms.

The scaling tab shows the variables as set up in the block. We need to make a decision whether to scale the engineering units. The unscaled PV and CV are listed at 3200 low to 21000 high. The Engineering Units for the PV may be changed or left as is. For water, the engineered units should be 91 gpm max.

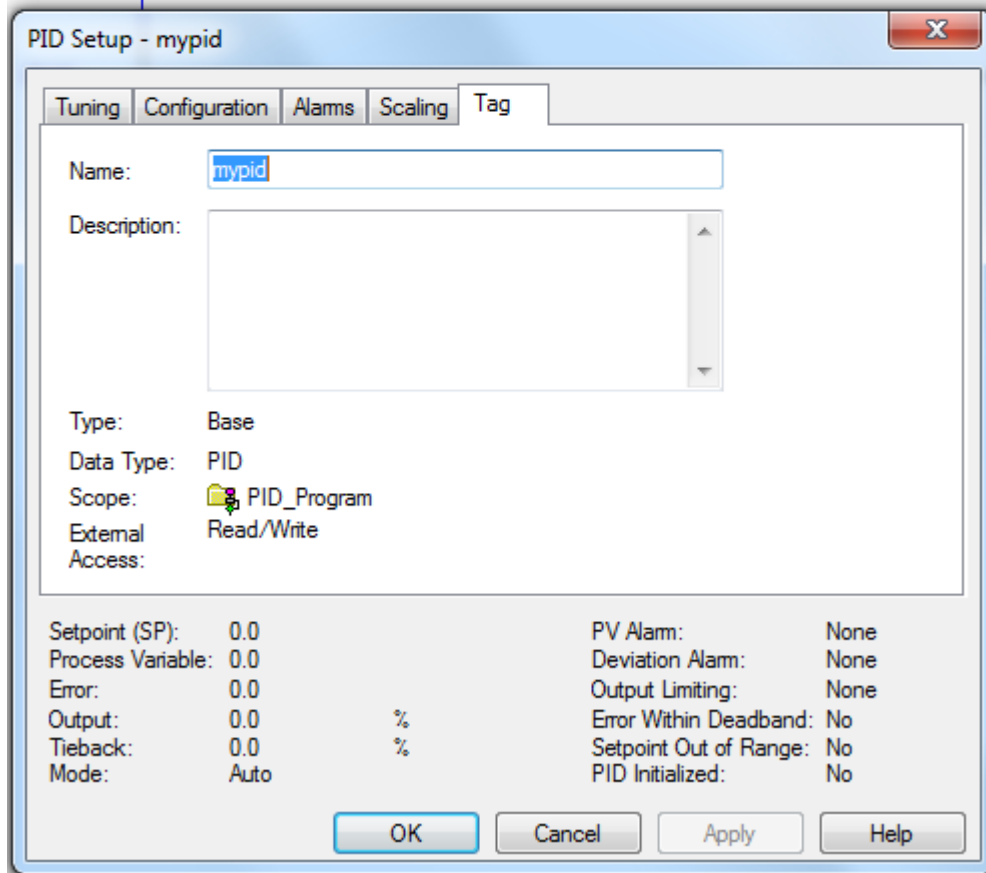
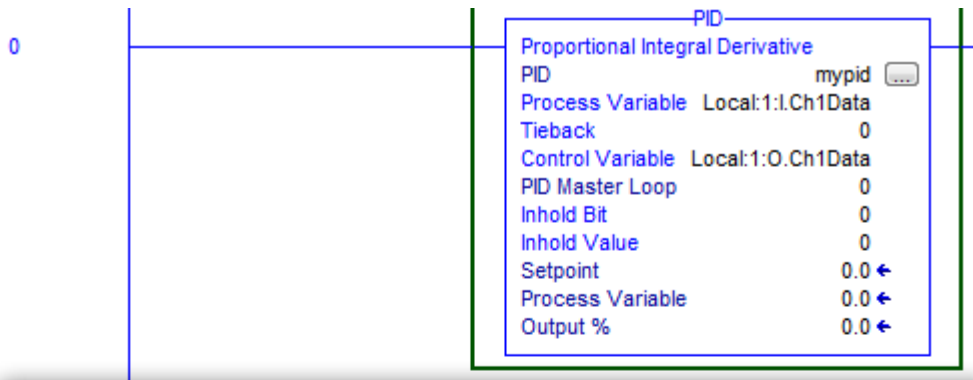
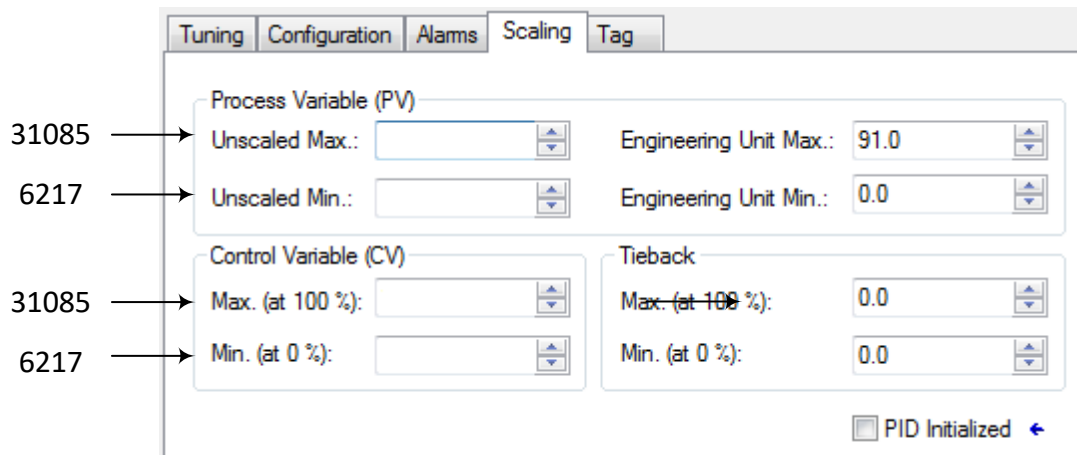


Fig. 19-66 PID Setup

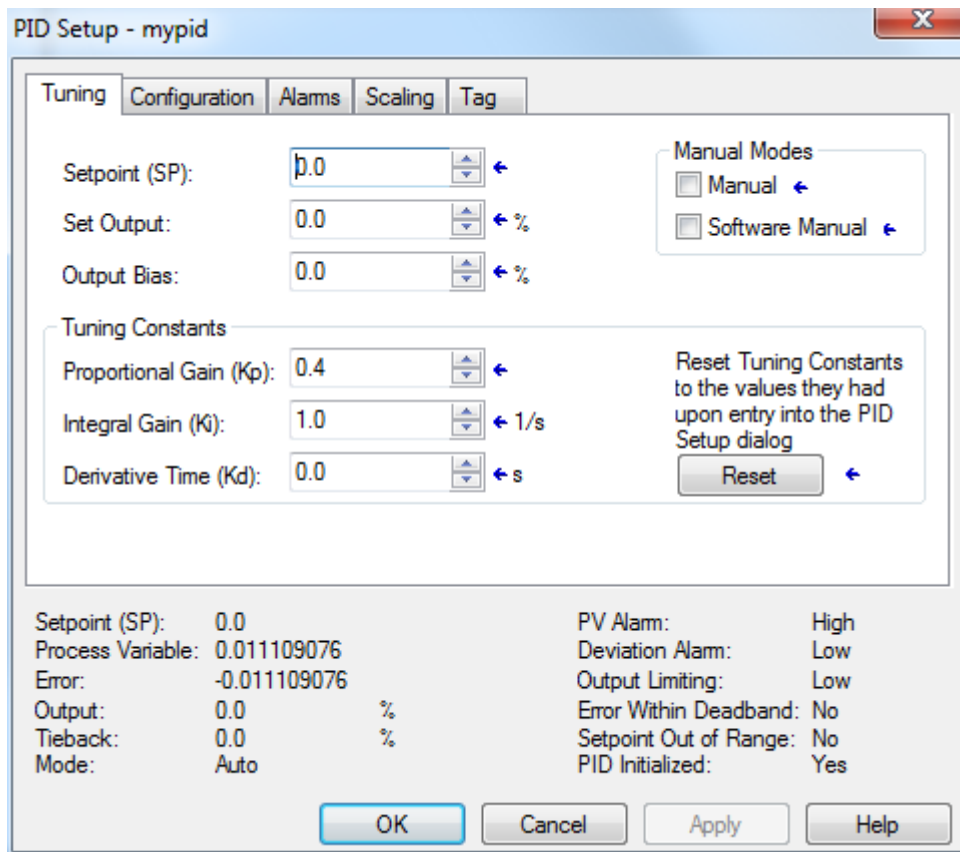


Fig. 19-67 Tuning Parameters

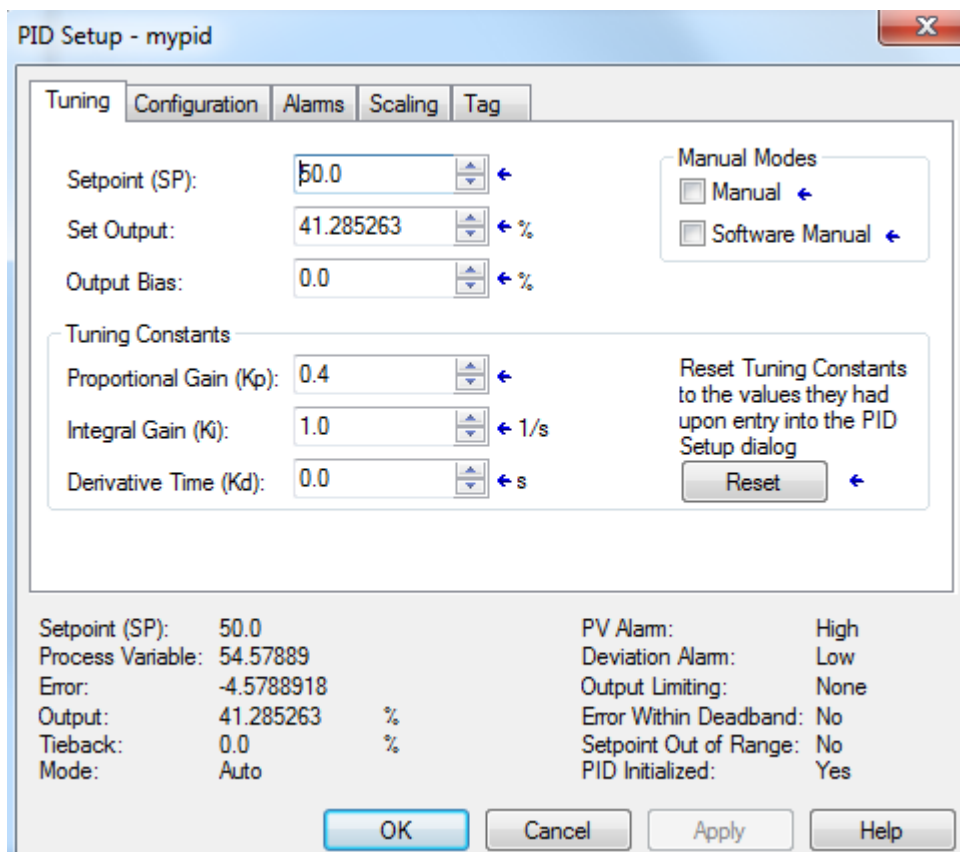


Fig. 19-68 Setpoint Trial

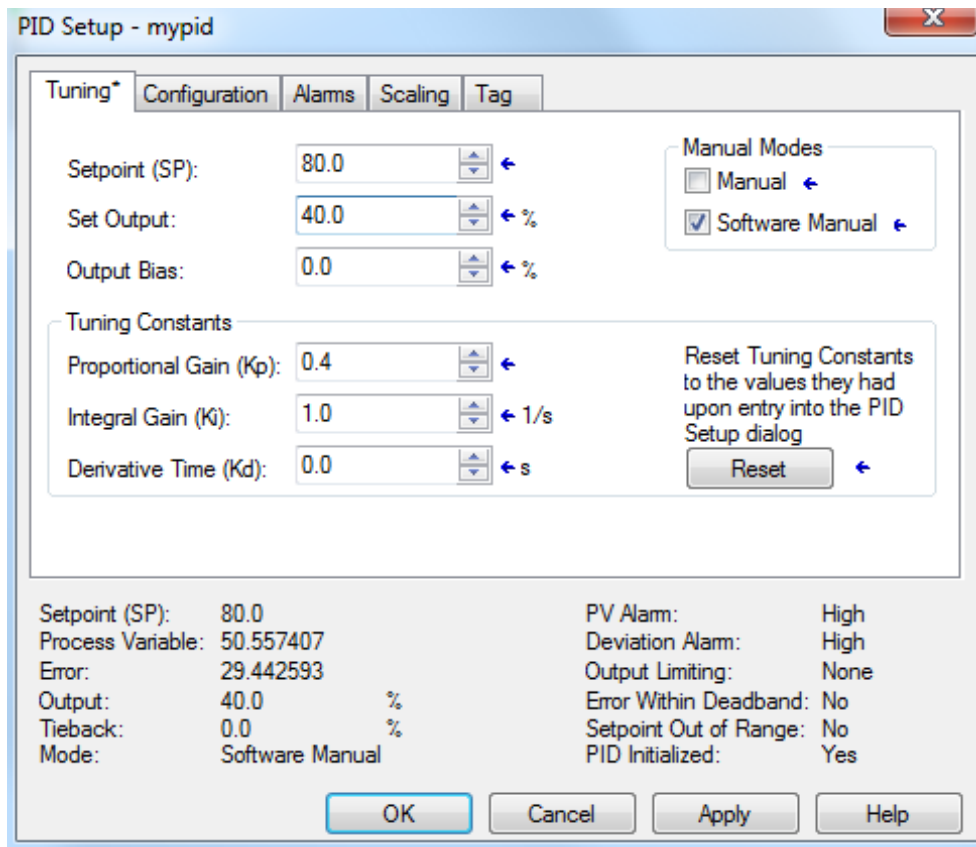


Fig. 19-69 Manual Trial

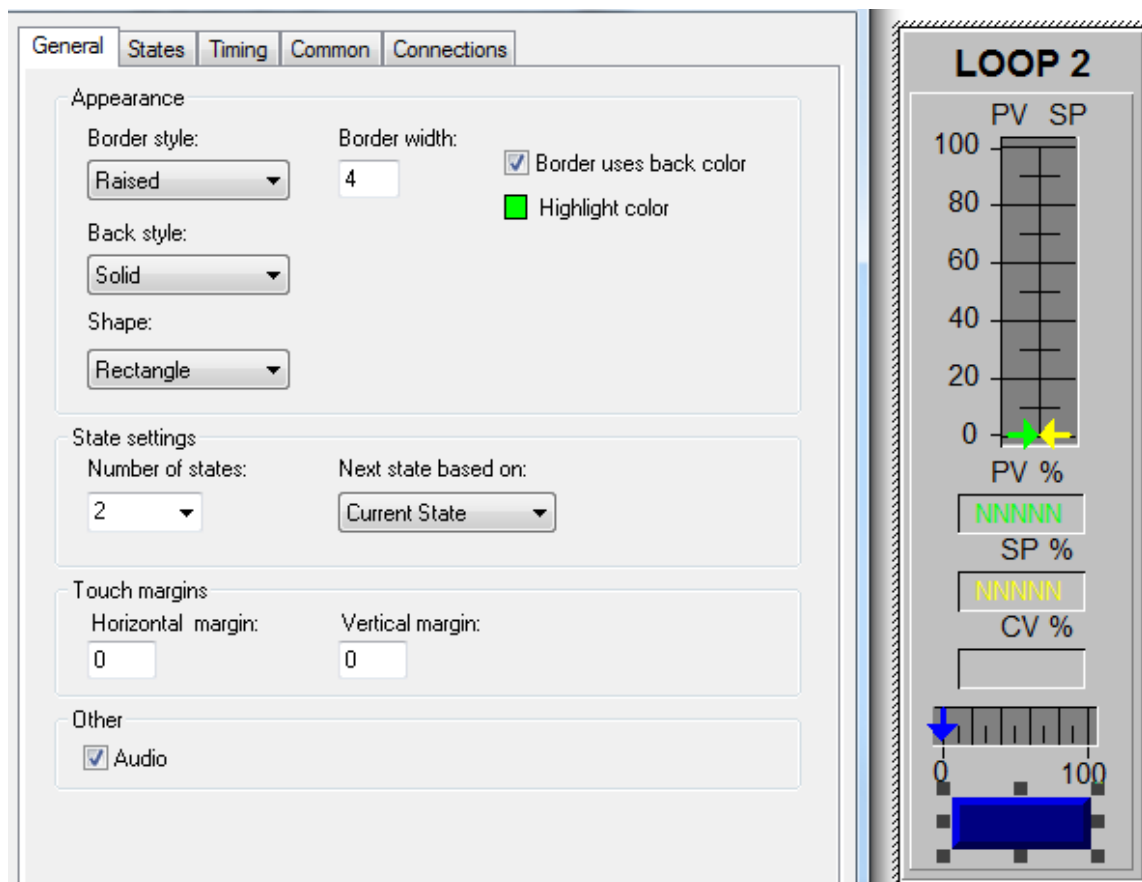


Fig. 19-70 Setup of the Faceplate

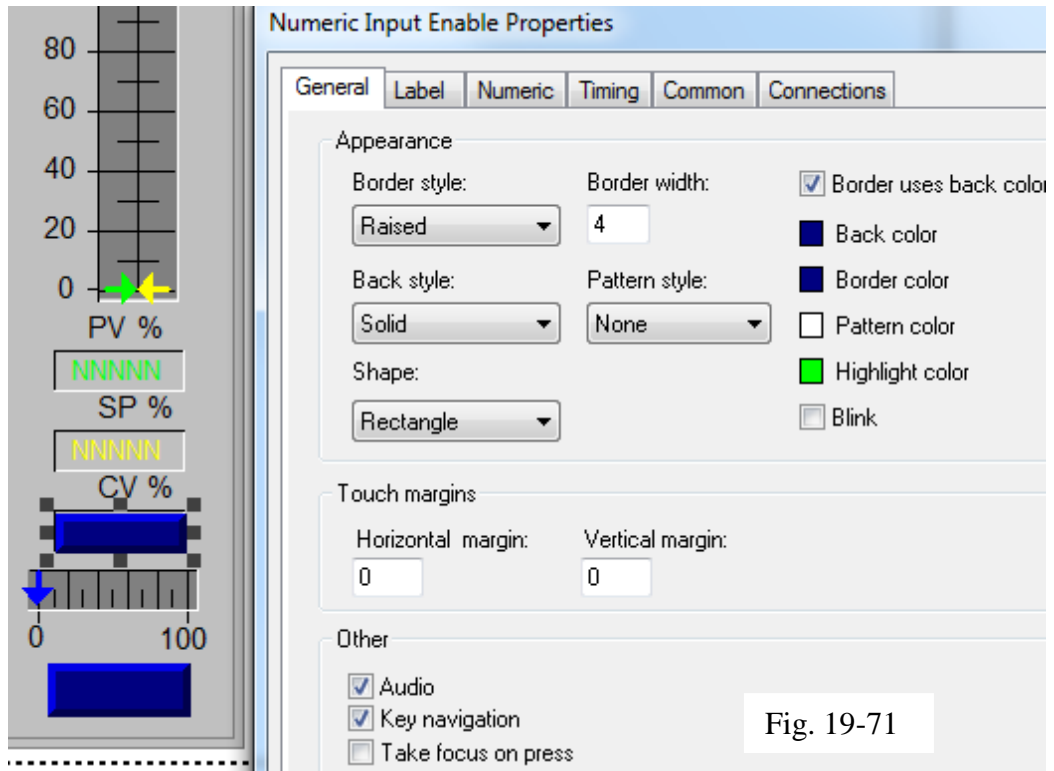


Fig. 19-71

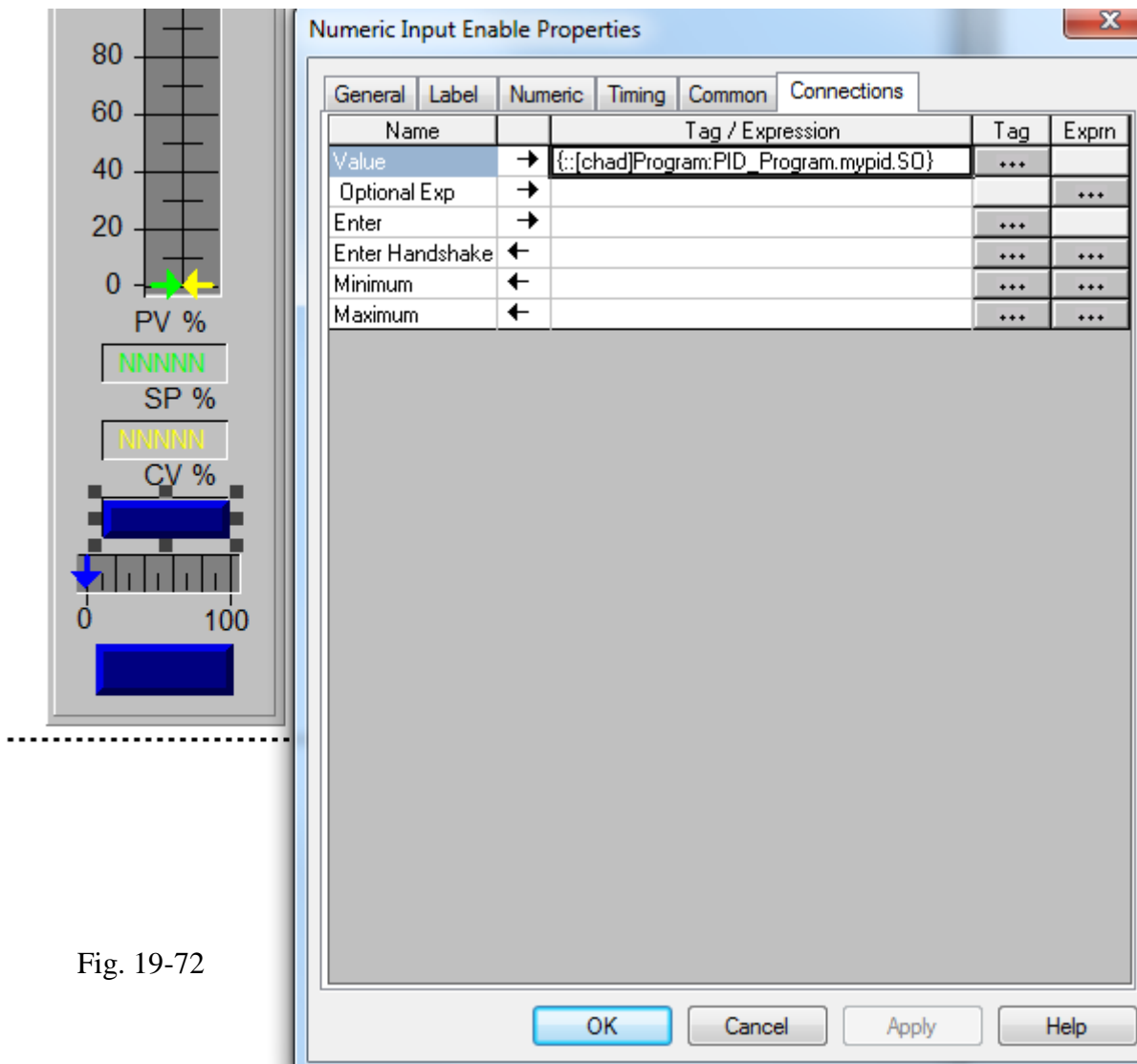


Fig. 19-72

Continuing the Allen-Bradley Configuration Pages

After you enter the PID instruction and specify the PID structure, you use the configuration tabs to specify how the PID instruction should function.

To specify tuning, select the Tuning tab. Changes take effect as soon as you click on another field.

To configure the PID:

Specify Setpoint (SP)	Enter a setpoint value (.SP).
Set output %	Enter a set output percentage (.SO) (In software manual mode, this value is used for the output. In auto mode, this value displays the output %.)
Output bias	Enter an output bias percentage (.BIAS).
Proportional gain (Kp)	Enter the proportional gain (.KP). For independent gains, it's the proportional gain (unitless). For dependent gains, it's the controller gain (unitless).
Integral gain (Ki)	Enter the integral gain (.KI). For independent gains, it's the integral gain (1/sec). For dependent gains, it's the reset time (minutes per repeat).
Derivative time (Kd)	Enter the derivative gain (.KD). For independent gains, it's the derivative gain (seconds). For dependent gains, it's the rate time (minutes).
Manual mode	Select either manual (.MO) or software manual (.SWM). Manual mode overrides software manual mode if both are selected.
PID equation	Select independent gains or dependent gains (.PE). Use independent when you want the three gains (P, I, and D) to operate independently. Use dependent when you want an overall controller gain that affects all three terms (P, I, and D).
Control action	Select either E=PV-SP or E=SP-PV for the control action (.CA).
Derivative of:	Select PV or error (.DOE). Use the derivative of PV to eliminate output spikes resulting from set-point changes. Use the derivative of error for fast responses to set-point changes when the algorithm can tolerate overshoots.
Loop update time	Enter the update time (.UPD) for the instruction.
CV high limit	Enter a high limit for the control variable (.MAXO).
CV low limit	Enter a low limit for the control variable (.MINO).
Deadband value	Enter a deadband value (.DB)
No derivative smoothing	Enable or disable this selection (.NDF)
No bias calculation	Enable or disable this selection (.NOBC).

No zero crossing in dbnd	Enable or disable this selection (.NOZC).
PV tracking	Enable or disable this selection (.PVT).
Cascade loop	Enable or disable this selection (.CL).
Cascade type	If cascade loop is enabled, select either slave or master (.CT).
<u>Specify Alarms</u>	
PV high:	Enter a PV high alarm value (.PVH).
PV low:	Enter a PV low alarm value (.PVL).
PV deadband:	Enter a PV alarm deadband value (.PVDB).
Positive deviation	Enter a positive deviation value (.DVP).
Negative deviation	Enter a negative deviation value (.DVN).
Deviation deadband	Enter a deviation alarm deadband value (.DVDB).
<u>Specify Scaling</u>	
PV unscaled maximum	Enter a maximum PV value (.MAXI) that equals the maximum unscaled value received from the analog input channel for the PV value.
PV unscaled minimum	Enter a minimum PV value (.MINI) that equals the minimum unscaled value received from the analog input channel for the PV value.
PV engineering units maximum	Enter the maximum engineering units corresponding to .MAXI (.MAXS)
PV engineering units minimum	Enter the minimum engineering units corresponding to .MINI (.MINS)
CV maximum	Enter a maximum CV value corresponding to 100% (.MAXCV).
CV minimum	Enter a minimum CV value corresponding to 0% (.MINCV).
Tieback maximum	Enter a maximum tieback value (.MAXTIE) that equals the maximum unscaled value received from the analog input channel for the tieback value.
Tieback minimum	Enter a minimum tieback value (.MINTIE) that equals the minimum unscaled value received from the analog input channel for the tieback value.
PID Initialized	If you change scaling constants during Run mode, turn this off to reinitialize internal descaling values (.INI)

Shifting to the HMI Program, RS Studio is entered and the Libraries choice and then Face Plates choice is entered.

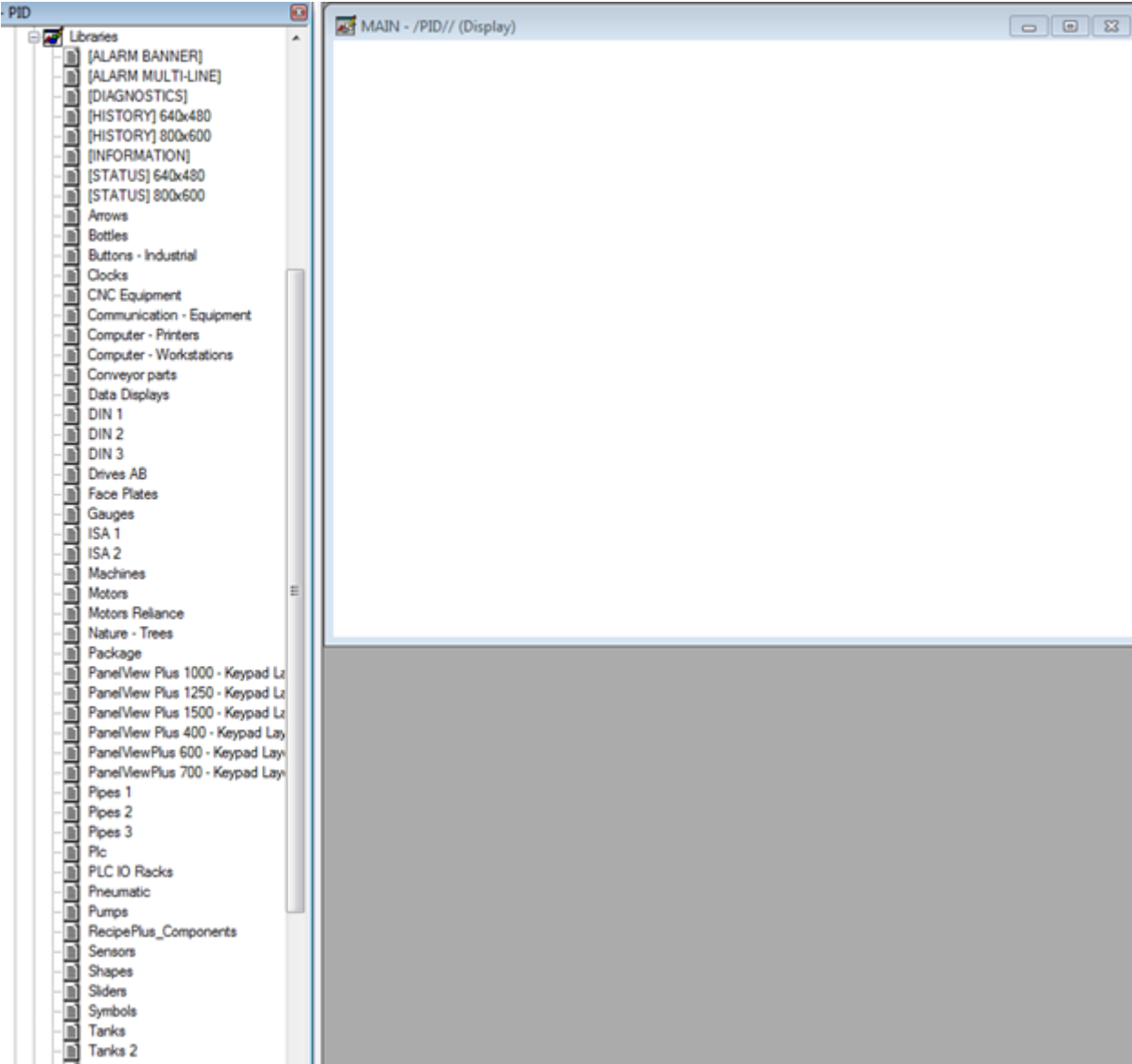


Fig. 19-74 Under Libraries – Face Plates

With RSStudio, build a screen from scratch using a face plate. There are a number of face plates in the template from which to choose.

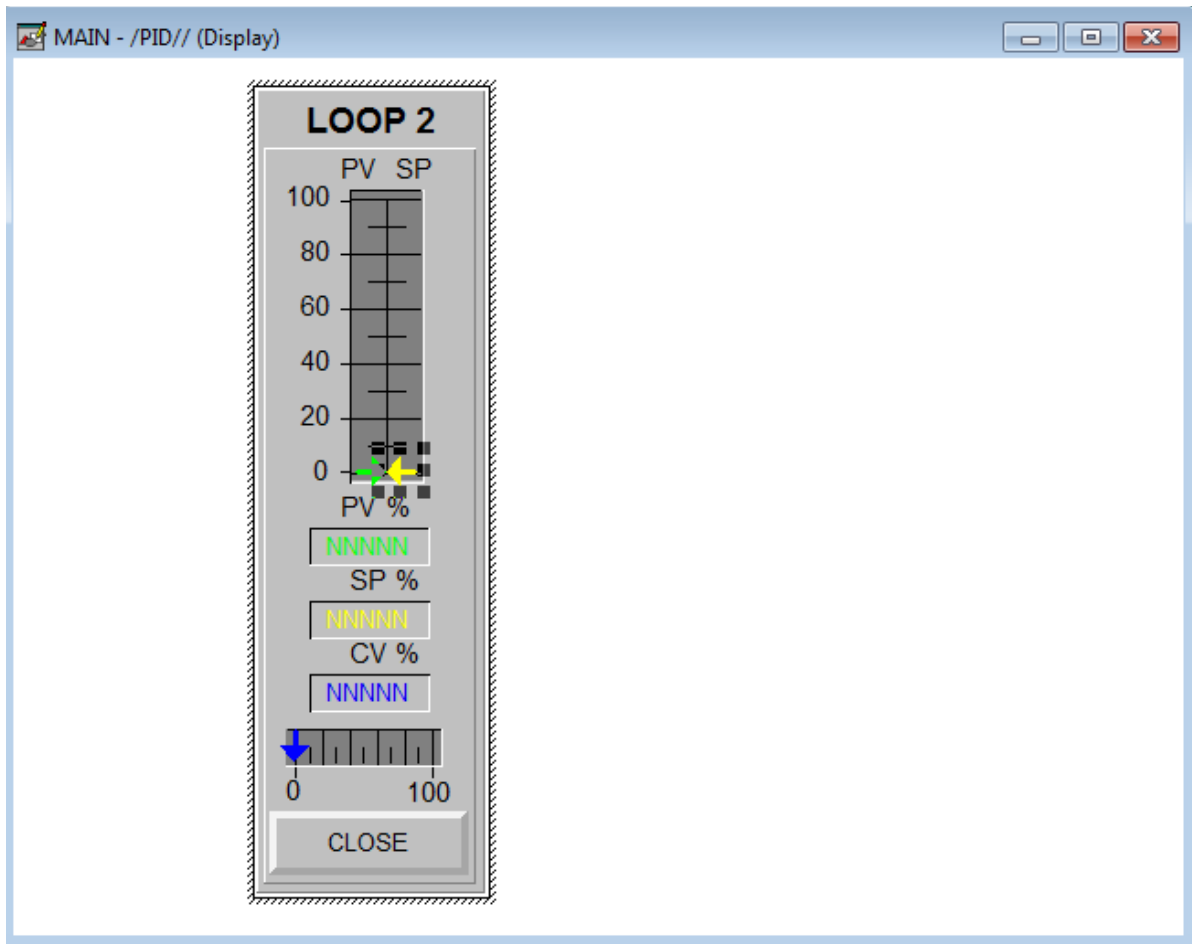


Fig. 19-75 HMI Loop Face Plate

The various parts of the face plate are animated. The next screen shows the details:

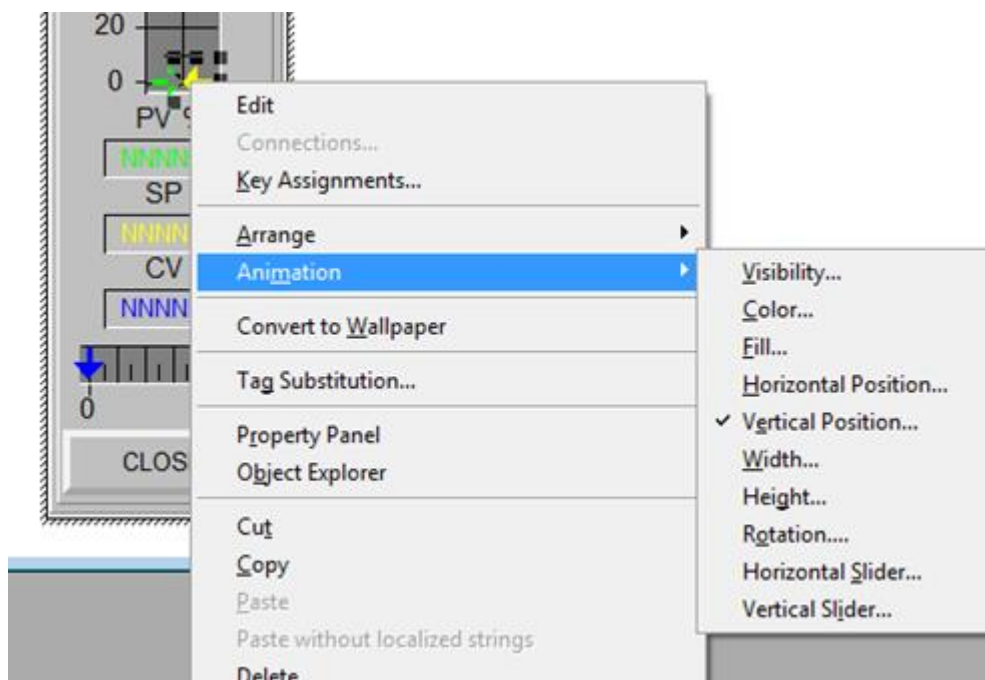


Fig. 19-76 Animation of the Arrow

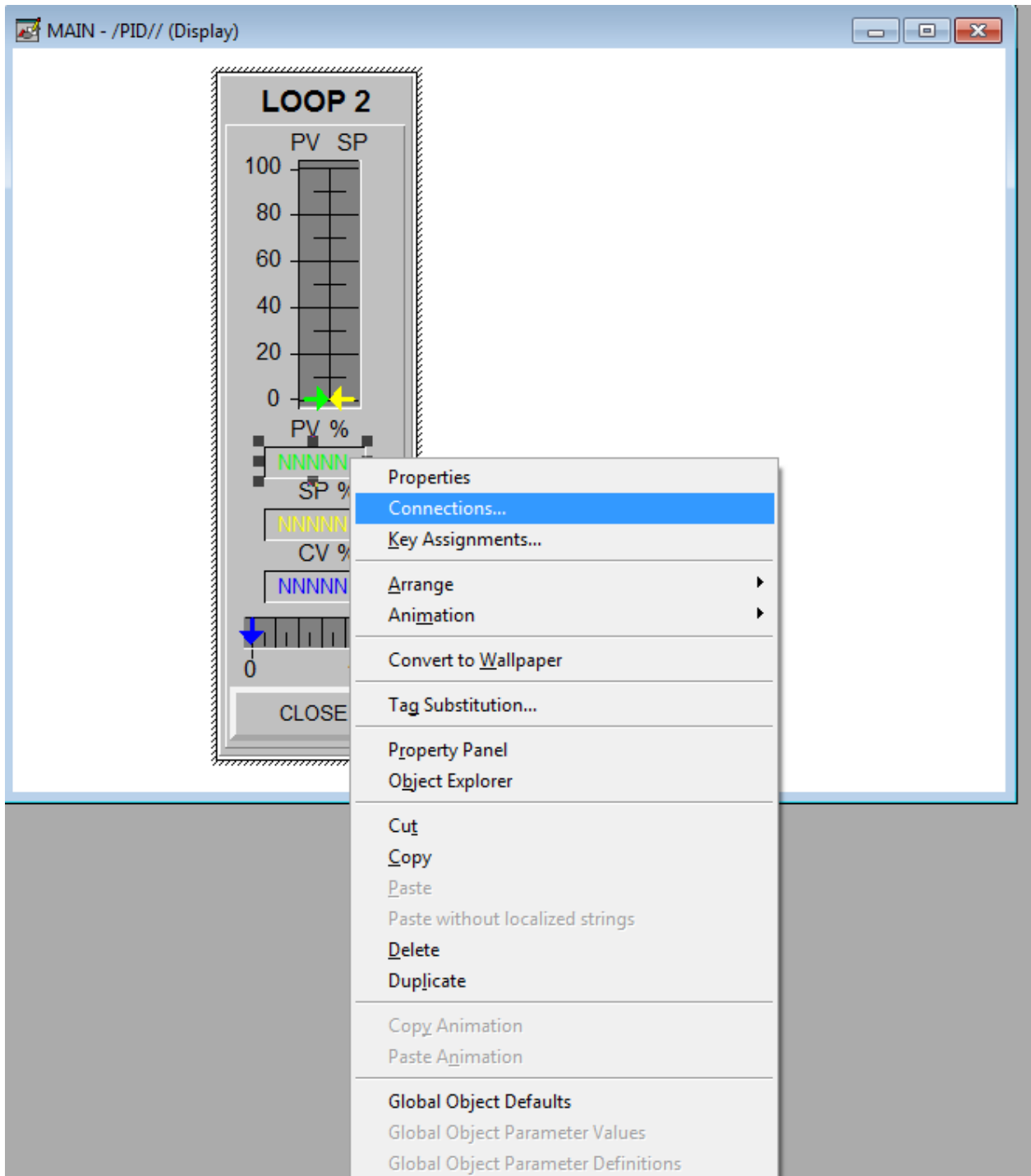


Fig. 19-77 Animation of the Numeric Entry

Building the Faceplate

The faceplates below are samples of single loop faceplates that are accepted by most industry. They resemble faceplates of actual PID controllers used prior to the computer. They may be more or less sophisticated than these and may include the 3-d look or not. These are samples of what is expected for proper HMI design of a faceplate.

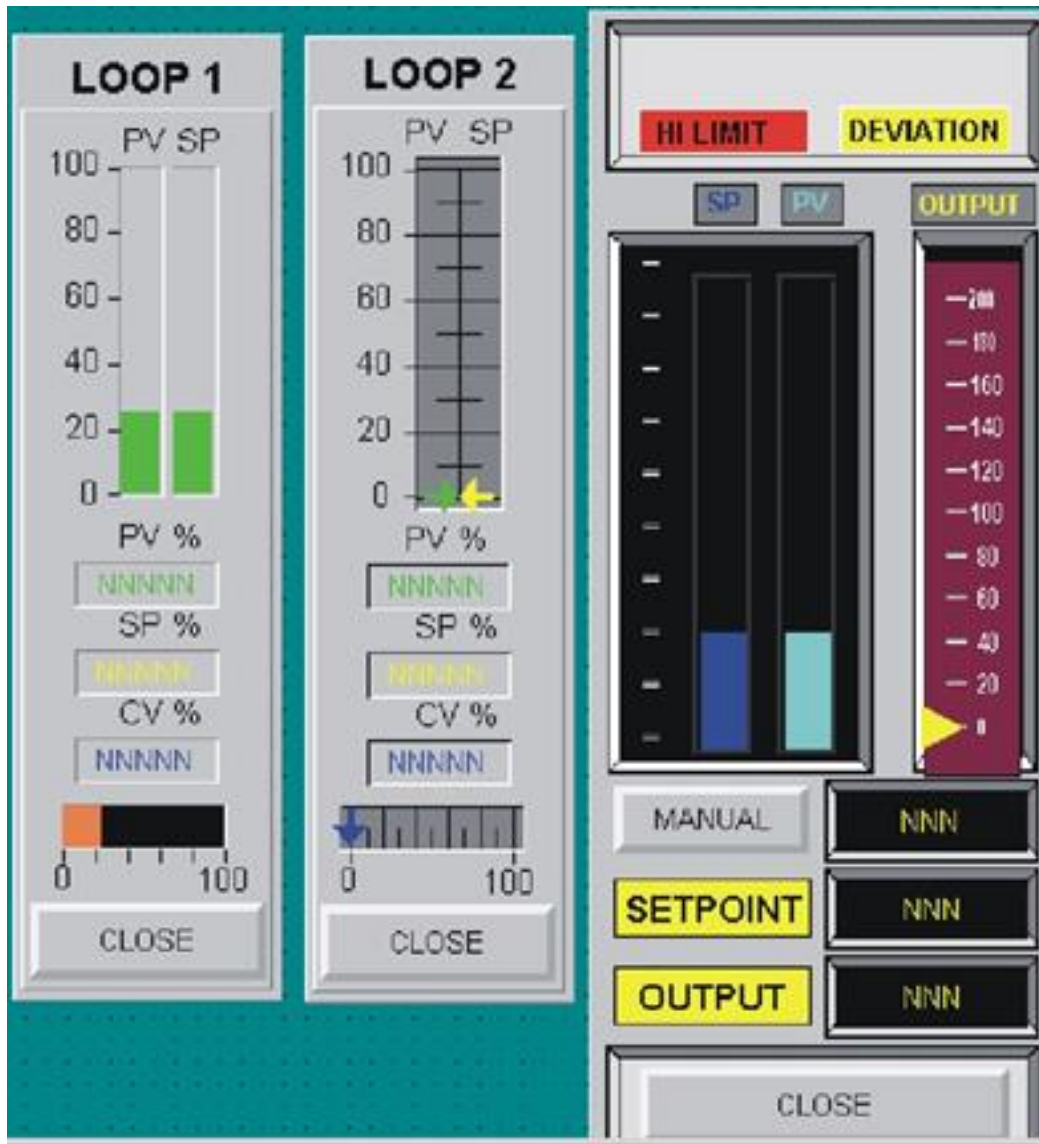


Fig. 19-78 One of Many

Choose a faceplate and begin modifying it for the application. Several tags are provided with each faceplate. These tags may set a number, allow entry of a number, move an animated arrow or fill a sliding window. Bits may be added for auto/manual and local/remote. Note that alarms may also be included such as the red and yellow tags above.

These faceplates may be modified with additional components. They may also be built from scratch using existing components. At one time, the faceplate could be unbundled. While no longer possible, the individual components may be animated by clicking them and then answering the questions.

The next two pages show the animation of the faceplates from Siemens and Allen-Bradley using the faceplate as the starting point for the animation. While the faceplate given is not available from Siemens, it can be built from parts using existing Siemens components. The up and down triangles shown in the earlier faceplate may also be added to these faceplates for a more complete system. The logic in the Siemens faceplate below show how to add the triangles.

The following logic can be used to add 1 % to the full scale value of the Setpoint. Similar logic can be used for 5% increase or for 1% or 5% decreases. The triangle buttons on the original faceplate showed these triangles. Similar buttons can be added to the CV or Output logic when the PID algorithm is in manual. Similar logic can be added to the Allen-Bradley program.

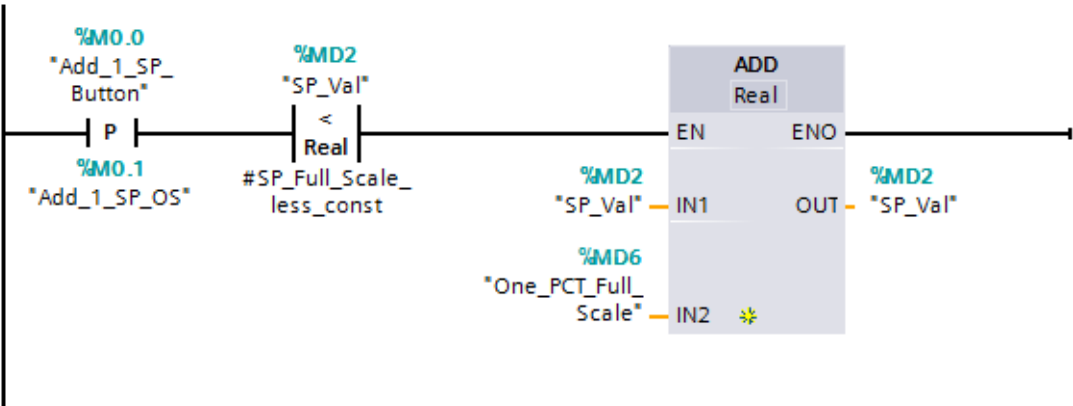


Fig. 19-79 Logic for Incremental Change of Setpoint

The following from HPHMI examples in Chapter 15 show a group of PID controllers. Each gives just the information necessary for the running of that controller. To change the mode of the controller or to run the controller in a mode other than remote requires a more complete faceplate. To add this feature, simply program an invisible button that calls a pop-up faceplate similar to the one below.

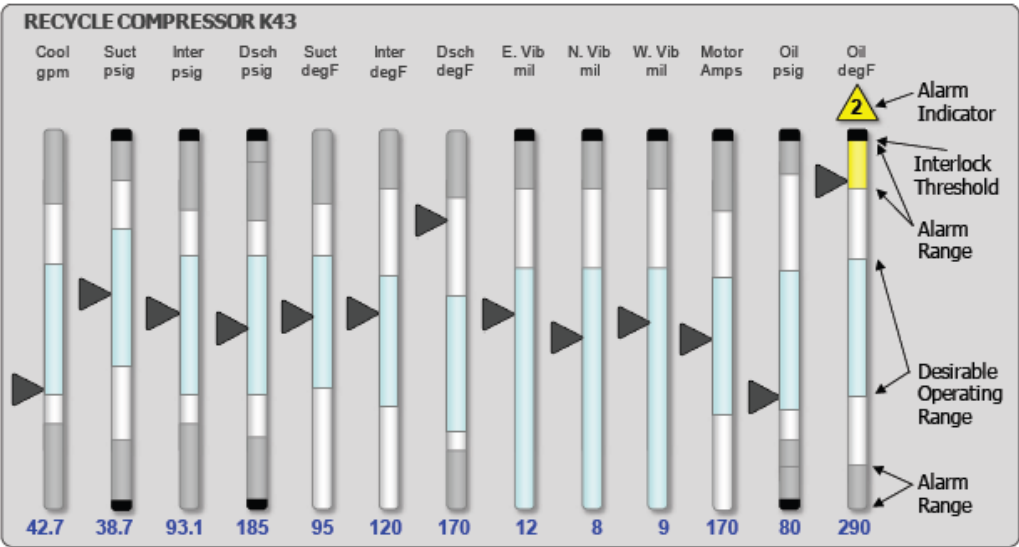


Fig. 19-80 Analog Depiction of Information

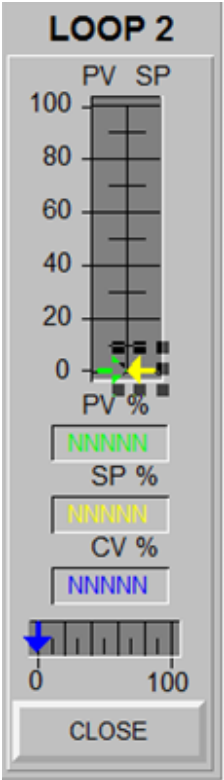
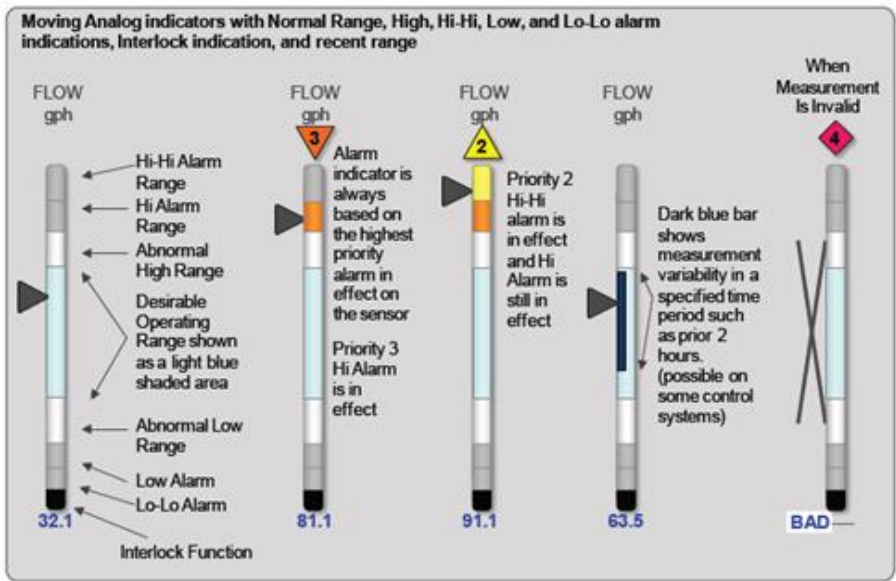


Fig. 19-81 Further Explanation of Moving Analog Indicators

Two new topics not explored in the earlier PanelView were alarm screens and trends. Alarm banners were available in the older PanelView but were not as flexible as the newer alarm screen. Also, trends are needed. Trend data is very important in that a trend of any variable can be used to diagnose a problem either in the start-up phase of a project or later during daily operation. Historical data trends will show long-term trends as well.

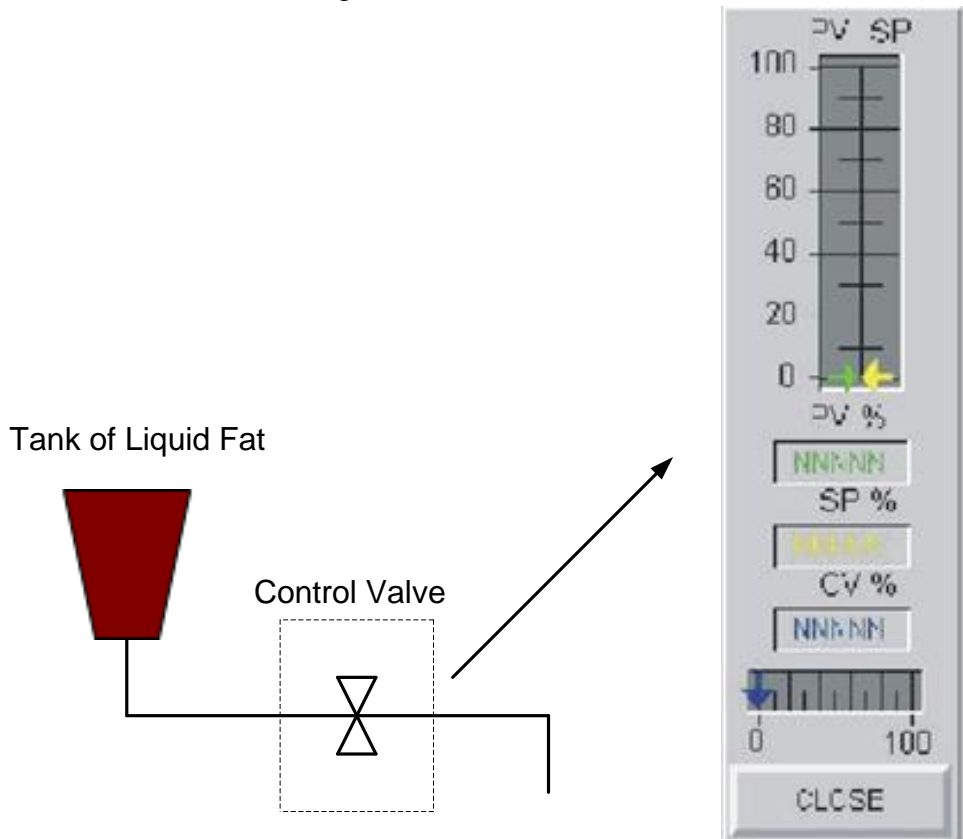


Fig. 19-82 Imaginary Button calling Faceplate

This figure shows a partially finished graphic of the ‘fat’ portion of the dog food extruder. When the invisible button around the valve is energized, the PID block faceplate appears allowing control of the valve in auto and manual mode. Local and remote control may also be added to the screen with the faceplate. The pipe may be enhanced as well to show flow when the valve is open and no flow when the valve is closed.

The graphical application may be run from the PC or downloaded to a target system. The tags for the graphical screen may be those in the PLC. Care must be taken when selecting where the process is to be displayed. If it is displayed from the computer screen, then Local is selected. If the display is downloaded to the Panelview32, then Target is selected. In order to display the process locally, a number of steps must be incorporated for the local application to correctly “see” the PLC.

Tuning the PID Block

It is interesting that a number of different PID algorithms exist. No one standard equation is used in all controllers. While the PID block has the same general function, nomenclature and the action of the block may differ.

Proportional Band	=	100/gain
Integral	=	1/reset
Derivative	=	rate – pre-act

Three classifications of PID algorithms are considered major classes of design equations. They are ideal, parallel and series or interacting. Equations for the three are listed below:

$$\text{Ideal:} \quad \text{Output} = Kc \left[e(t) + \frac{1}{I} \int e(t) d(t) + D \frac{de(t)}{dt} \right]$$

$$\text{Parallel:} \quad \text{Output} = Kp[e(t)] + \frac{1}{I} \int e(t) d(t) + D \frac{de(t)}{dt}$$

$$\text{Series (Interacting)} \quad \text{Output} = Kc \left[e(t) + \frac{1}{I} \int e(t) d(t) \right] \left[1 + D \frac{d}{dt} \right]$$

Different manufacturers use one of the above control algorithms as the basis for their PID block. The three do not respond identically to different situations. A control algorithm from one manufacturer cannot be guaranteed to work identically to the control algorithm of a second manufacturer. Differences in the derivative action are especially critical to the operation. For this reason, many do not use derivative action in the tuning of a loop. To not use derivative action, set the derivative or D value to zero.

Manufacturers such as Honeywell, Bailey, Allen-Bradley, Modicon, Foxboro, Fisher, and Texas Instruments pick one of the above types of equation to implement on their controllers. Some manufacturers allow a choice between which algorithm is used. It is the engineer’s or technician’s responsibility to understand the application, the PID equation, and choose the best

overall solution for the application.

Using the PID Algorithm to Control a Process

To configure a system, a flow diagram must be drawn to identify the parts of the system. The example below is of a dog-food manufacturing facility. The basic process for making the dog food is the extruder whose function is to make dog food from dry ingredients along with some steam, fat, and other wet ingredients. As the motor speeds up, more ingredients are to be added and as the motor slows down, the added ingredients are to slow down as well. The PID block will be used to add one wet ingredient, fat.

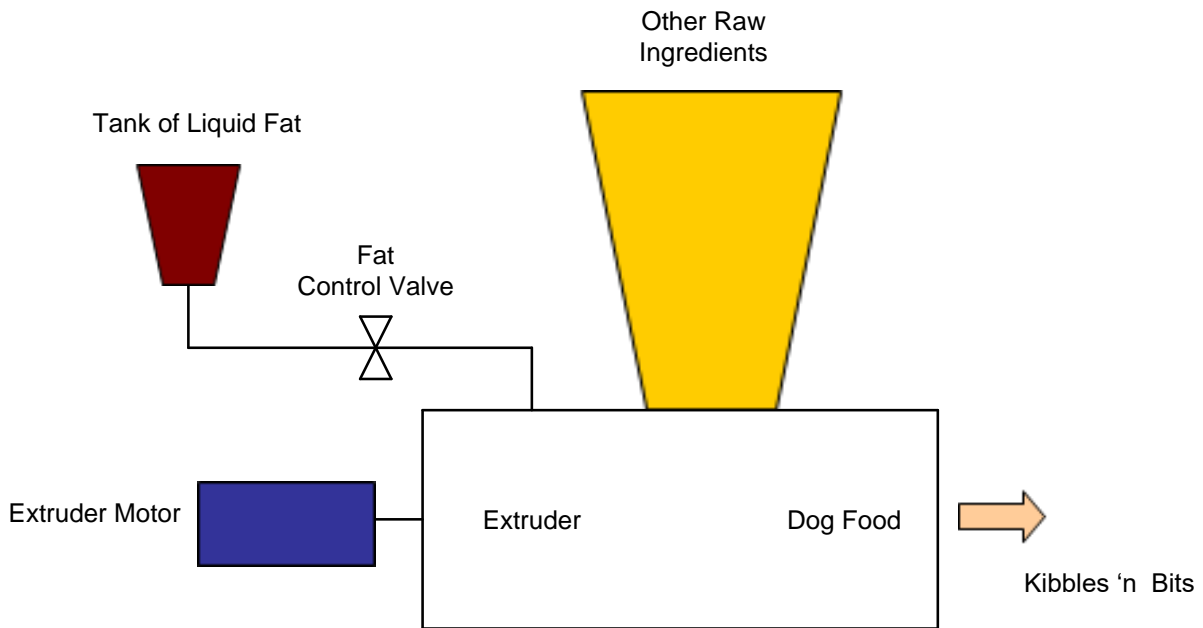


Fig. 19-83 Extruder/Mixing System making Dog Food

Since the extruder motor speed runs the feed speeds for the other ingredients in the process, its speed sets the master speed for the process. All other feed speeds will be a percent of the motor speed.

Control signals for the Dog Food Control include:

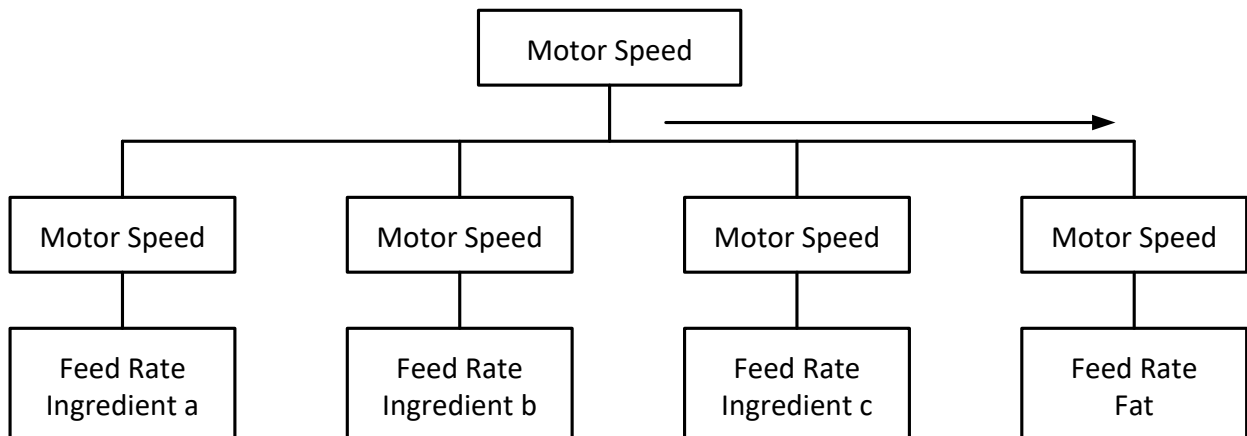


Fig. 19-84 Cascade or Remote Signal

When the PID algorithm is in remote, the motor speed furnishes the value. Variables are usually multiplied by a constant with motor speed * multiplier giving the value of the setpoint when the local-remote switch is in remote.

The example will be used as a lab exercise at the end of the chapter. Design of the faceplate will show selector switch positions for local versus remote and manual versus auto. Usually a graphic of the system is provided with a button activated that shows the faceplate. The screen with the faceplate is not the primary screen but is accessed as needed. The process screen displays the entire process with various pop-up buttons available to show the PID algorithm for that portion of the process as the operator needs to access a specific PID block. Many times the buttons to activate the PID block are configured as invisible. If the operator pushes the area around the valve – ‘fat’ valve in this case – the PID block for ‘fat’ will be displayed. The diagram below follows the signal path through the PID block and is useful as a programming aid. Looking only at the Fat Feed, the following process flow will be implemented:

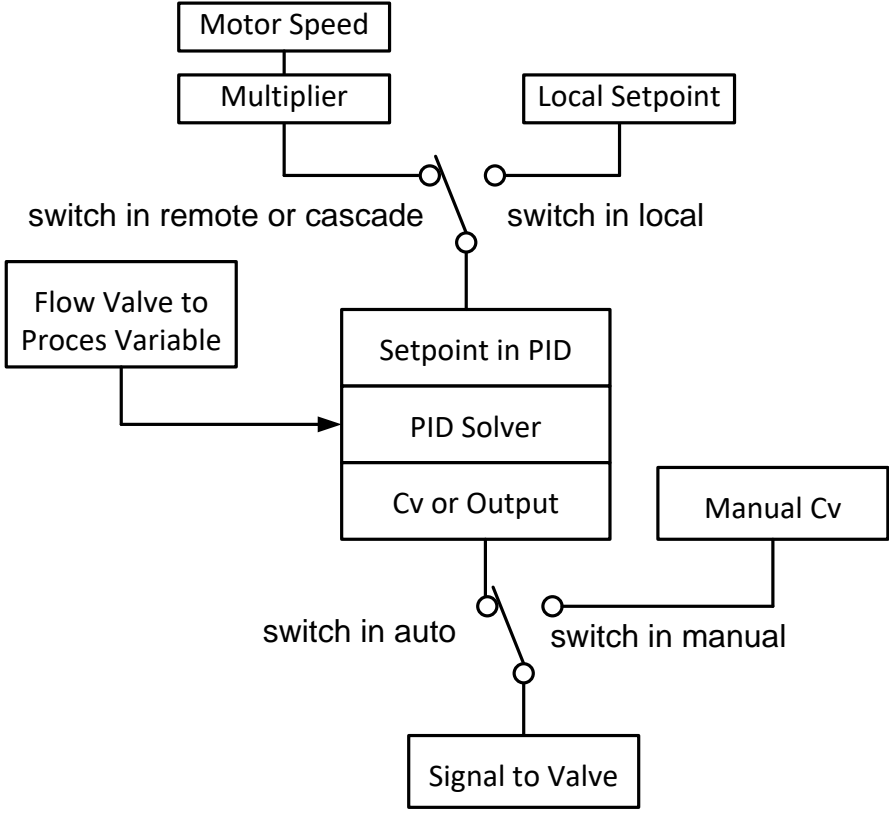


Fig. 19-85 Motor Speed Settings for Ingredient Adds

Bumpless Transfer

When the PID block is switched from manual to auto, the function responds to the SP presently available to the block. If the process is sensitive to sudden changes in PID output, then the program should include logic to give the output a signal matching the present flow when the block was in manual. This is referred to as bumpless transfer.

With the more advanced PID blocks of the PLC/5 and Control Logix platform, the output value that is described as the value to write to so that the output will be bumpless is the .SO value. The

.SO value of the PID block should be given the value that the operation would like the output to have when the PID block is first put in Auto. This value is usually the value of the output when the PID block is in Manual. The MOV operation should guarantee bumpless transfer when the block moves from Manual to Auto.

For example, if the block was in manual and flow was 25.5 gallons per minute, when the PID block is transferred to auto, flow should continue to maintain 25.5 gallons per minute. With PID blocks, the addition of logic requires writing the present flow rate to the setpoint when the block transfers from manual to auto.

Non-Standard Controller Modes

A number of additional modes may be created for the PID block. Bits must be programmed externally to the PID block for many of these other control modes.

An example is Control Output Tracking (COT). In COT, the loop is forced to manual and the output moves to a programmed position until conditions in the program are stable enough for the system to proceed to auto. In COT, the mode shown to the operator is AUTO with COT. The system is perceived to be in Auto but the output or CV is actually in Manual. This mode is ideally suited for burner start-up with a large number of burners. When the burners are first turned on, the gas and combustion air are not able to be controlled under automatic control. The burners need to operate in the extreme low range of the CV but the control valve cannot be allowed to completely shut off. In the low range of most valves, proper flow rates are not accurate and control becomes very unstable. COT allows the PID loops to operate for a set period of time in manual at a preset position until the burners are all started and flows are at their mid-range positions more capable of accurately being controlled. Then the PID algorithms take effect in Auto and the PID loops begin the process of controlling the temperature in the furnace. To the operator, the system appears to be in auto but in the program, the PID algorithm is being controlled in manual until the auto mode is capable of accurately controlling the PID block. COT is to be used only in start-up situations or in recovery operations in which it is necessary to operate at a low-end setting to keep the burner system from shutting down.

When operating in a mode such as COT or Maintenance and when the mode is removed, the loop should resume its former status.

Use a toggle input from the HMI and the following logic to program bits for A/M, L/R, COT, and Maintenance.

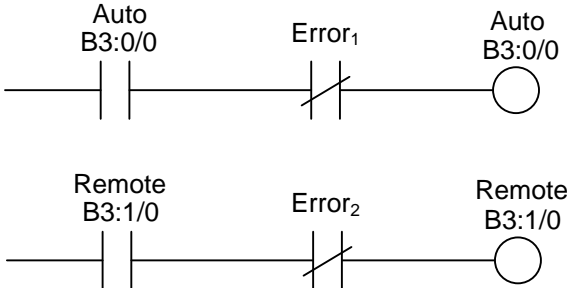


Fig. 19-86

Use of toggle bits to turn on a mode may not at first resemble a seal or latch circuit but in fact they act in a manner similar to both. The toggle bit (B3:0/0 or /1) may be turned on by an operator through the HMI and will remain on until the operator removes the toggle or until the NC contact logic interrupt the flow. When this happens, the circuit reverts to the safer off state. In the example of auto/manual (bit B3:0/0), the bit will turn off to the manual state. Note that the actual state of the SLC Auto/Manual bit is reversed from this logic.

Floating Point PID

The subject of what type of PID block to choose is an easy decision. Always use the Floating Point PID block if floating point is available. The number representing the flow or pressure or temperature is an actual number with units and no need to be transposed to another number elsewhere. With the integer PID block, it is very important to keep a record of the various transpositions so the PID block can be used at maximum efficiency with numeric values sent to the operator that relate to the process.

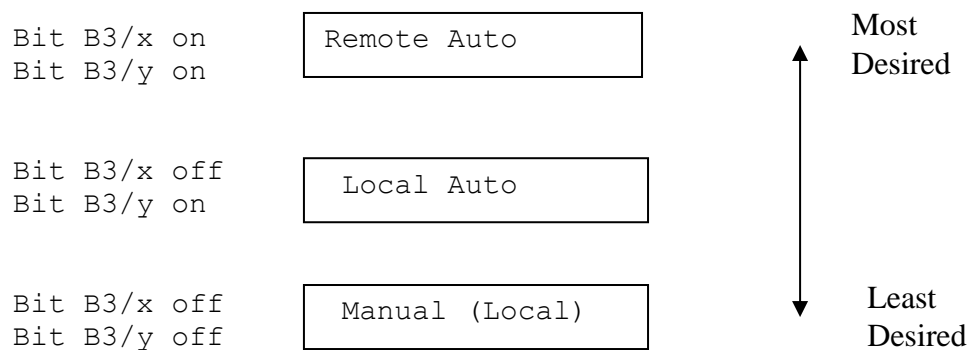
PID function blocks using floating point numbers are preferred. For instance, if flow varies from 0 to 45 gpm, then the numbers entered for minimum SP and maximum SP's would be 0.00 and 45.00. However, to gain accuracy, any integer setpoint should use the entire range from 0 to 16383. The min. value 0.00 equals 0 and the max value 45.00 equals 16383. With the integer PID block, there is a translation in the values between internal units and values displayed to the operator. For examples in the text, this translation is ignored. In an actual application, however, each translation must be implemented with an appropriate SCP instruction. Effort to keep all translations in order is not seen as necessary and most complex applications tend to use floating-point PID.

Calibration may be used to determine units of flow. In order to determine flow, a test is run with a watch and a calibration system. For instance, running a 5 gallon bucket full of water in a certain time is an acceptable method of calibrating flow through a valve. Repeating the calibration a number of times over a range of settings gives a better overall measurement.

Fault Circuits

Faults occur at different levels in the program and require a variety of responses. Some types of faults should shut the process down. Shutting down may require that valves turn off. Many times, to shut down automatic operation is desired and the valves are to stop moving, staying in the same position. If the desire is to move from Auto to Manual, the bit in the PID algorithm labeled AM must be changed from 0 to 1. The bit is set to 0 in Auto and 1 in Manual. The fault contact represents various faults that can harm the process if the PID algorithm is allowed to continue in auto.

Two levels are present in most processes. As with the dog food application, the process is capable of being run in remote or local for both automatic modes or in manual. In a hierarchical picture, remote mode is favored over local mode and the manual mode is the least desirable mode to run the process. This may be pictured as:



Bit B3/x is the Remote Control Bit
Bit B3/y is the Auto/Manual Control Bit

Fig. 19-87

Note that when the PID block is in auto, the control bit is on. A second bit must be programmed to reverse the status of this bit to turn off the AM bit in the PID block to correctly run the PID block.

One of the control button types in PanelView is ideal to program the Remote/Local and Auto/Manual layout for the PID block. It is the Multistate Button. Define two multistate buttons for the process above. Reference the first multistate button to B3/x to represent Remote or Local. Reference the second multistate button to B3/y to represent Auto or Manual.

Let B3:0/0 represent the remote/local mode and let B3:0/1 represent auto/manual.

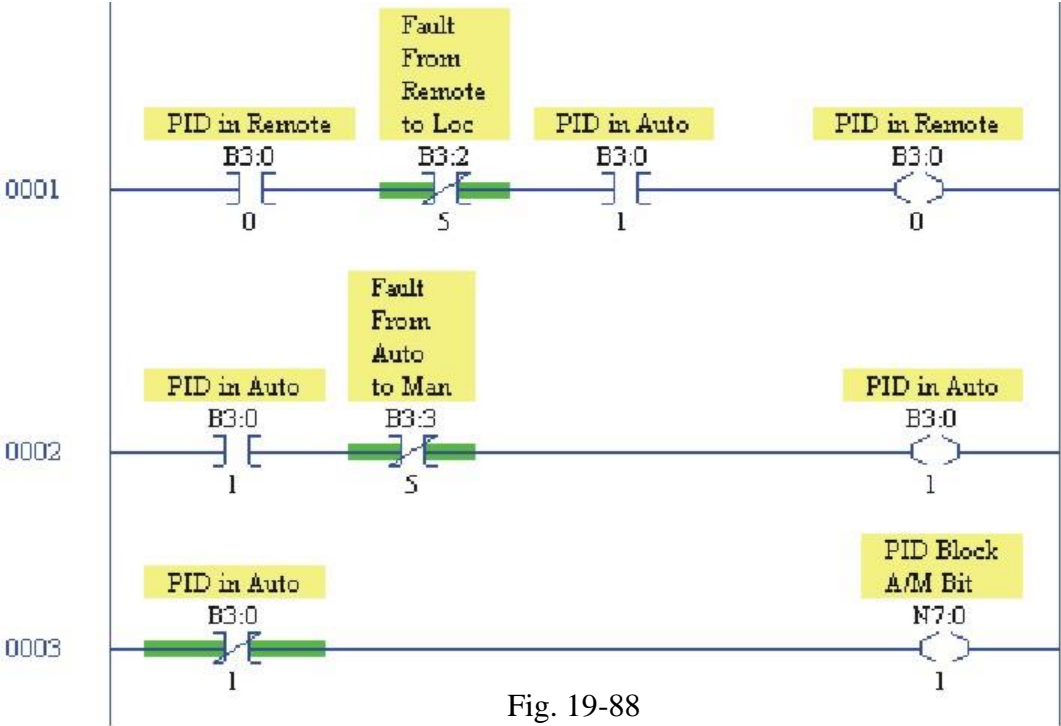


Fig. 19-88

The logic for control bits for remote/local and auto/manual is provided. Multistate pushbuttons are programmed in the HMI for B3:0/0 and B3:0/1. B3:0/0 is labeled Remote when the bit is on and Local when the bit is off. B3:0/1 is in Auto when the bit is on and Manual when the bit is off. The state is set to 'on' when the operator places the buttons in the remote or auto mode. The operator can also place the buttons in local or manual mode. Operation of the process can also place the process in the local or manual mode as well when faults occur. Faults as represented by B3:2/5 will energize the NC contact and take the PID block from remote to local. Faults represented by B3:3/5 will energize the NC contact and take the PID block from auto to manual.

Multistate buttons are used for remote/local and auto/manual so one button can be used instead of two buttons. Most graphical applications encourage the use of a single button as opposed to two separate buttons. Using the multistate button provides a single button with toggle functionality. Multistate buttons also respond to program logic in the PLC and will turn on or off with logic internal to the program.

To complete the mode program for the PID block, be able to add logic to the rungs above to turn on or off B3/0 and B3/1 from the program as well as from the HMI. From the HMI software, configure two multistate buttons. These buttons are programmed as follows:

Button 1	B3:0/1	Tag
	Off	Local
	On	Remote

Fig. 19-89

Button 2	B3:0/0	Tag
	Off	Manual
	On	Auto

Faults that move the operation from remote to local are different than faults that move the operation from automatic to local. Always, the option most highly sought is for the operation to run in remote. However, if a fault occurs in the process but not necessarily in the individual PID block, the fault should cause the process to revert to local from remote and sound an alarm.

If a fault occurs in the PID block, the best practice is to change the block from automatic to manual. One of these faults is referred to as anti-reset windup. In manual, the algorithm is not active and the error term is reset to zero eliminating the integral term from growing with a growing error.

Example of Fault Causing Switch from Remote to Local

When looking at PV, a temperature profile may be found to form a composite PV. The values of a number of different temperature inputs are summed together. The sum is weighted with the weighted values having to add to 100%. If the weights do not add to 100%, the individual PID blocks used to control their CV outputs are switched to local mode. The local setpoint is used until the weights have been adjusted to add to 100% and the operator switches control back to remote.

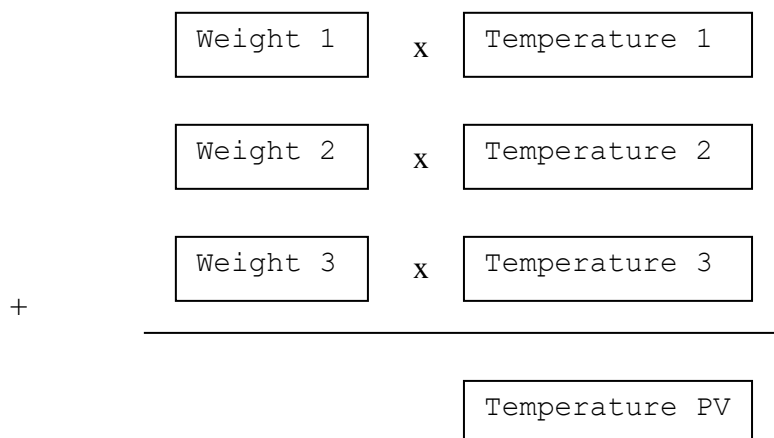


Fig. 19-90

In the example, Weights 1-3 must add to 100 % for the Temperature PV to run the temperature PID block in remote.

Example of Fault Causing Switch from Auto to Manual

When operating between Auto and Manual, the PID block should be monitored so that a failure to achieve the desired result is not defeated by faulty equipment. If the equipment fails, the PID block should be faulted to the Manual Mode and an alarm sounded. For instance, if a valve is attached to the CV and the valve does not turn when the CV changes, this should be considered a fault condition. To find if this is the case, the CV or output is compared to a position on an analog scale. The sensor is usually nothing more than a potentiometer. If the CV does not keep within 10% (or other constant) over a time period such as 10 seconds, the PID block for the valve should fail.

Another type of failure is the restriction of flow that can cause the CV to travel to full 'on'. A restriction in flow may be simulated by simply pinching off a hand valve in the line of flow. Any restriction over time can cause the CV to not be able to control the process. If the CV is allowed to go to 100% for a period of time, the PID block should fault and the output be placed in Manual. Ranges other than 100% may be used as well with a time delay appropriate to shut down the process in abnormal conditions. The programmer must be able to decide acceptable ranges for these cutoffs, usually through experience with the PID block and with the process.

Eliminating Anti-Reset Windup

In order to avoid anti-reset windup of the PID controller, the controller must be switched from auto to manual when conditions exist that would wind up the controller integral term. The integral term is reset to zero in manual mode. To detect integral error, monitor the PV. If the PV does not follow the CV after a preset time, something is perceived to be wrong with the system and action should be taken.

For example, a check valve may be turned off starving the system. When this happens, the PID controller must be placed in manual to eliminate windup and an alarm sounded.

An experienced operator will find the problem and reset the loop to auto control. And the system will continue to function with only a small upset to the system. If the PID block is allowed to wind up over several minutes or hours, the output valve may stay open 100% (or closed 100%) for long periods of time after the system comes back into operation before control is re-established. In this time period, excessive gas may flow through a gas valve causing an explosion or too much liquid may flow through a control valve flooding a process vessel downstream. In any case, the result usually upsets the entire system causing scrapped product or worse.

When switched from Auto to Manual, the error integral term is reset to zero:

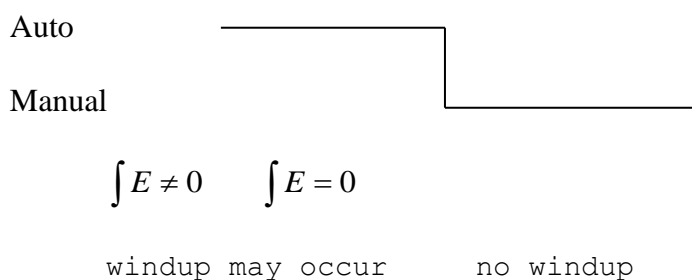


Fig. 19-91

When switched from Manual to Auto, the error integral term starts at zero and adjusts:

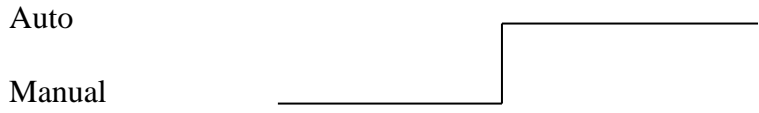


Fig. 19-92

$$\int E = 0 \quad \int E \neq 0$$

no windup error term initially 0

Changes from Manual to Auto are usually made by the operator and imply that the operator is aware that a problem occurred, has found the problem and is ready to put the process back into Auto.

Building a Ramp Block

A ramp block is a function block that is added in front of a PID block to change the SP over a period of time instead of immediately. It is constructed in the PLC diagram to increment from the old SP to the new SP in increments of 1. More sophisticated ramp blocks allow the ramp rate to be set by an operator or engineer. Some PLC instruction sets include a ramp block. The SLC instruction set does not include a separate Ramp block so one must be programmed from available instructions.

In this example, the old setpoint was 50 and the new setpoint was 62. In order to move from the old setpoint to the new one, the SP value must be incremented to climb. The rate at which the SP is incremented may be changed which varies the rate at which the new SP achieves its value. For example, if the time interval is lengthened, the new setpoint is reached much later.

A setpoint may vary as high as 5000 or more integer units and the incremental ramping may need to be very rapid (in msec). Quickly moving ramp blocks are possible with the higher speed timer blocks. Ramp blocks may also require very slow operation and this can be accomplished using slower preset timer blocks. Examples of slow-acting ramp blocks include cure operations that require hours to advance the setpoint to the final point or a ramp-soak operation for operations such as steel in which the annealing requires a slow temperature rise over an extended period of time.

It is preferred for the ramp block to move in small increments. If the increment speed in units of 1 is less than the PID update speed, increments should definitely be handled in increments of 1. The goal of the ramp block should be a smooth continuous ramping. An example of a Ramp of a Setpoint built from Ladder follows:

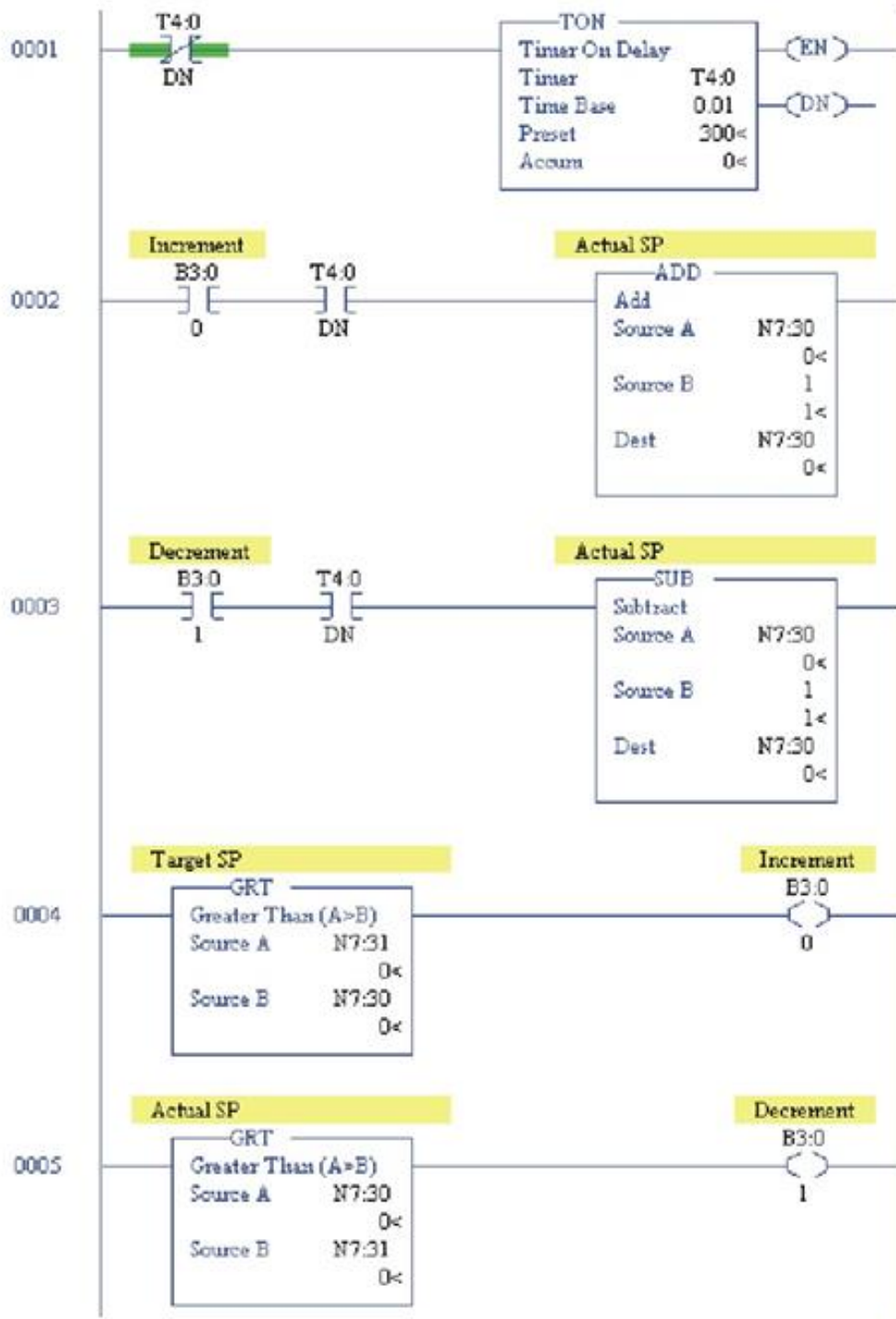


Fig. 19-93 Ramp Block using Ladder

Ramp blocks are used to cause the PID block to be tuned to a different set of tuning constants than would be required if the ramp were not present. A PID having ramping would have a set of tuning parameters that would be tuned to respond to only much smaller step changes seen with small upsets in the process. In block diagram format, if a ramp function is needed, it may be shown as a block before the PID SP as follows:

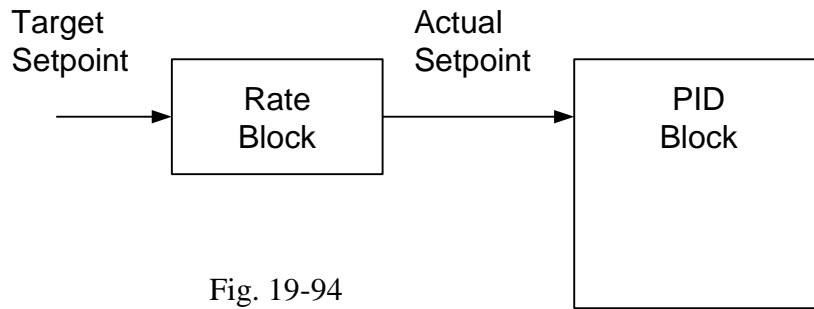


Fig. 19-94

Loops within Loops

The discussion now describes multiple PID blocks used to control a process. The following example shows how a PID loop can be imbedded within another PID loop:

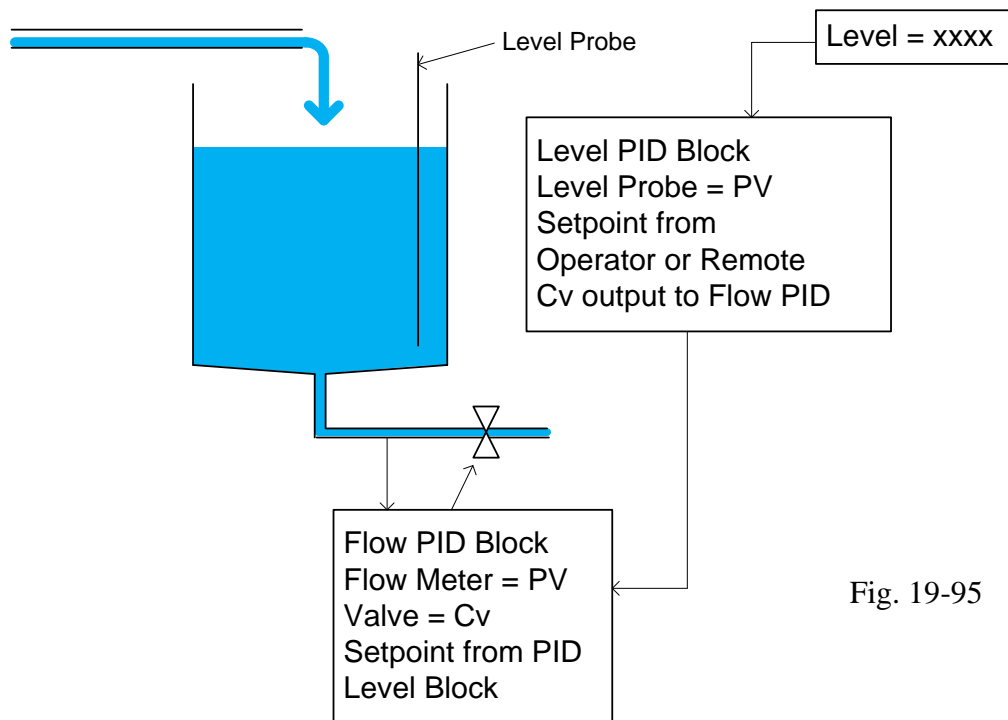


Fig. 19-95

In the example above, the inner loop is the flow valve with its setpoint the CV from the Level PID block. The outer loop is the Level PID block controlling level in the tank.

To successfully tune loops such as these, it is important to establish the order for tuning the loops. It is also important to establish parameters for tuning them.

1. Tune the inner loop first. In this case, tune the Flow PID loop first.
2. Establish comfortable tuning parameters for it and then proceed to tune the outer loop. The outer loop should be tuned to respond more slowly than the inner loop. The outer loop in the example is the Level PID loop. Try to tune it to respond about 2 to 10 times slower than the inner loop.
3. Stability problems occur in general if the two loops are tuned too closely together or the outer loop is tuned to respond more quickly than the inner loop. So, keep the

inner loop fast, outer loop slow and observe any instability. Ramp blocks should not be used on PID blocks such as these unless they are very quick acting. The inner loop should not have a Ramp block.

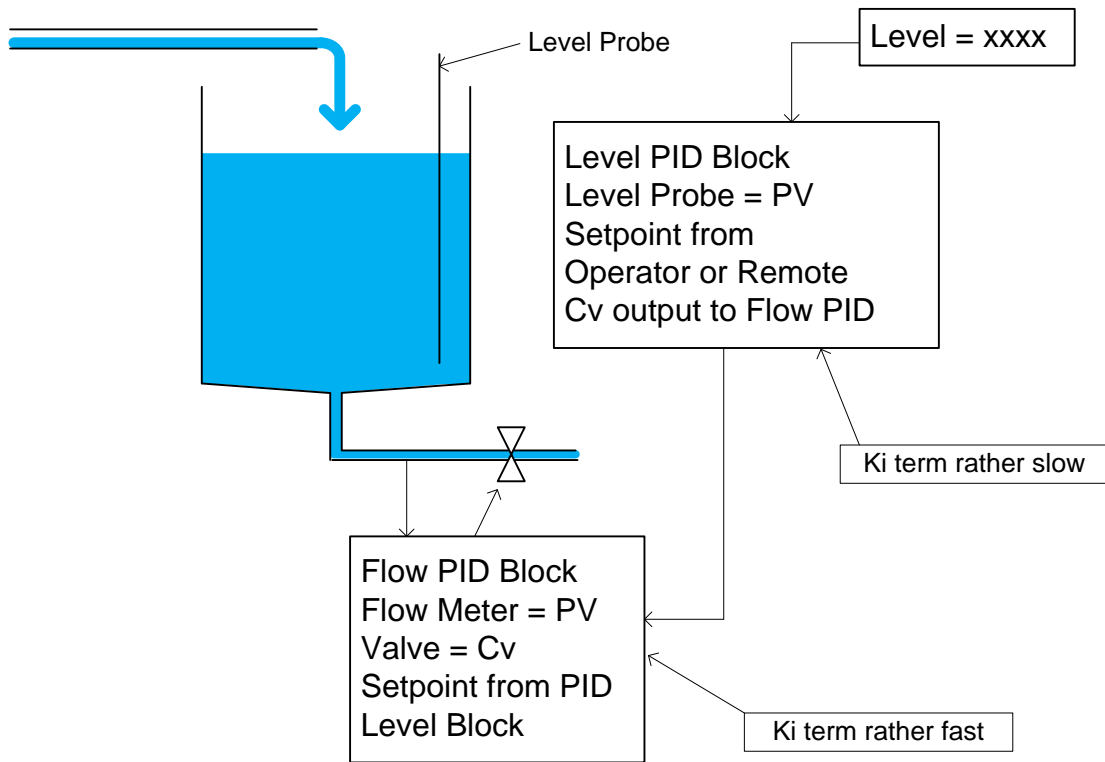


Fig. 19-96 Loop in a Loop

Additional PID Labs are included – all from Siemens to further develop the PID concept to other control projects.

PID Beyond the Lab

Using Multiple Controllers for Temperature Control

Most systems used in process control require a number of PID loops working together. In the example of the dog food extruder, the system would have included a PID controller for each ingredient. In general, each control element requires a PID block.

In the case of temperature control with gas and oxygen combustion, temperature is a PID block as well as gas and oxygen flow. The interaction of temp, gas and air are shown next:

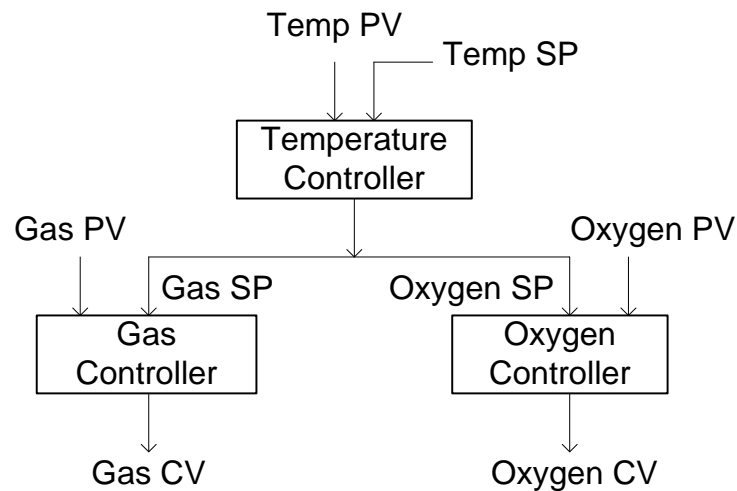


Fig. 19-97

This algorithm controls the combustion for a furnace or section of a furnace. Temperature Setpoint may come from a number of sources. The local SP may come from an entry from an operator. Setpoints may also be calculated using a formula for best performance. Setpoints from a formula would be considered as remote setpoints in the temperature PID loop.

In some applications involving gas and oxygen, the oxygen must be guaranteed to be in excess relative to fuel. Otherwise, excess gas may build up in the chamber and explode. Above certain temperatures, gas will burn without exploding. This is an especially prevalent condition in some steel reheat furnaces.

In the case of gas and oxygen below the critical temperature for gas to burn, a cross-limiting control scheme is introduced to allow only enough gas to be present to burn with at least enough oxygen or combustion air to burn all the gas all the time. This implies that the gas valve always must be more closed than the oxygen valve (times the air-fuel ratio). Control of the cross-limiting requires the same temperature control as the master control but introduces lag control, high select, low select and other control blocks in addition to the PID control. The oxygen control for the cross-limiting control algorithm would be:

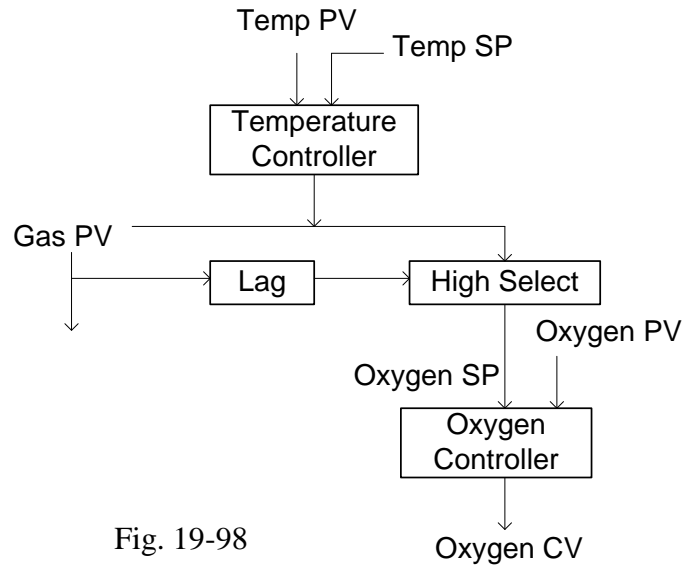


Fig. 19-98

The gas control for the cross-limiting control algorithm would be:

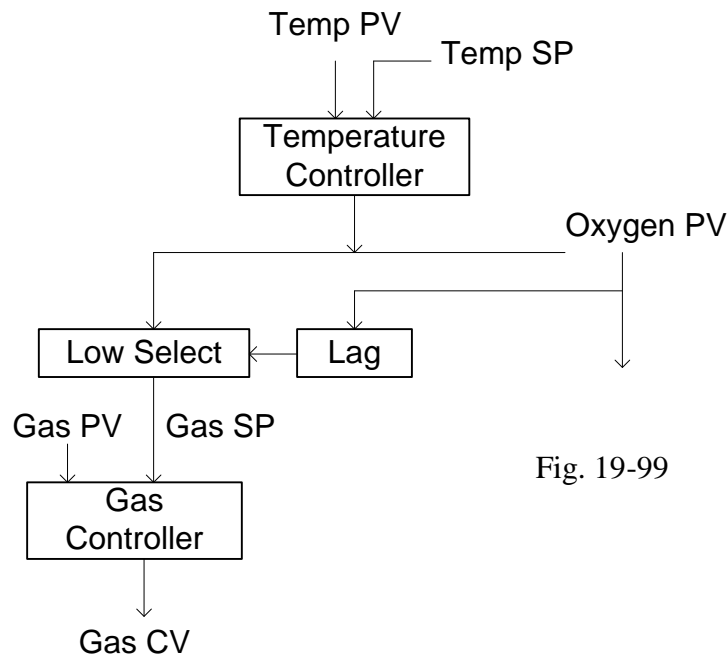


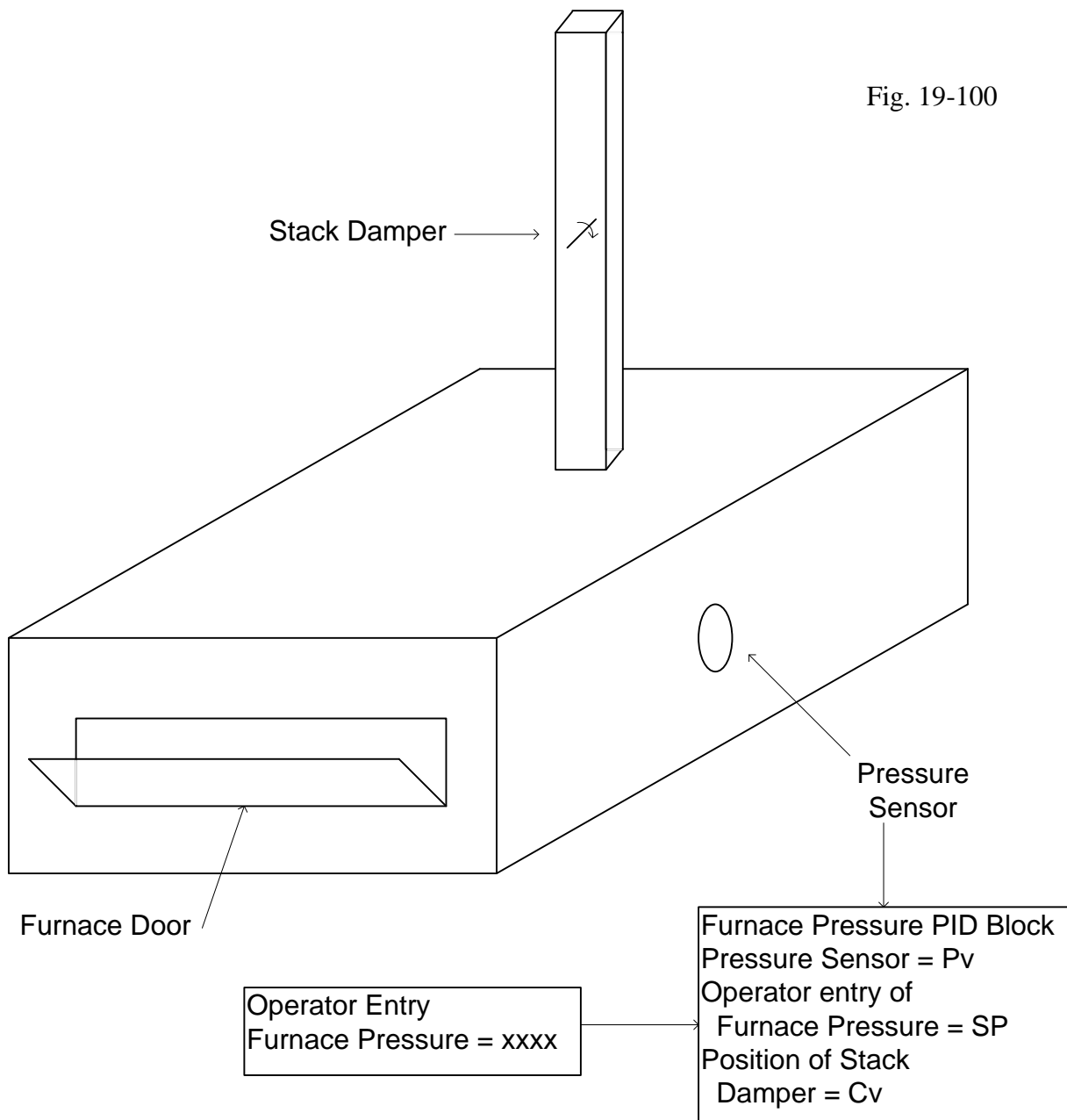
Fig. 19-99

As can be seen, the Gas PID block selects the lower of the values of the Temperature Setpoint or the Oxygen value after a lag has occurred. The effect of the cross-limiting control is to assure a Gas-Oxygen ratio that will never allow more gas into the combustion chamber than can be burned in the combustion process. This is an example of a much more complex algorithm than was first discussed earlier with a simple PID block. The same PID blocks are still used.

Example of PID Block for Feedforward Control

The PID block is a device used for feedback control. Many times, however, a small amount of feed-forward control is required. Feed-forward control may include control that anticipates an action and is ready to apply control as a situation arises more quickly than the pure feedback solution is able to provide. Since there is only one set of tuning parameters for the PID block, it is not practical to switch to a second set of parameters for a special case. The following example shows how a little tweaking of the PID block can be useful for some anticipatory or feed-forward control. The example below is of a furnace with a door on the front. This example shows just one of many additions to the PID block to give it characteristics not normally associated with PID control.

The gas burners use air for combustion and the air must be exhausted through an exhaust stack. Pressure in the furnace is adjusted by adjusting the damper in the stack. Pressure should be adjusted to be slightly negative so flames do not jump out of the door when the door is opened.



The concern of the pressure PID loop is:

What happens when the door opens?

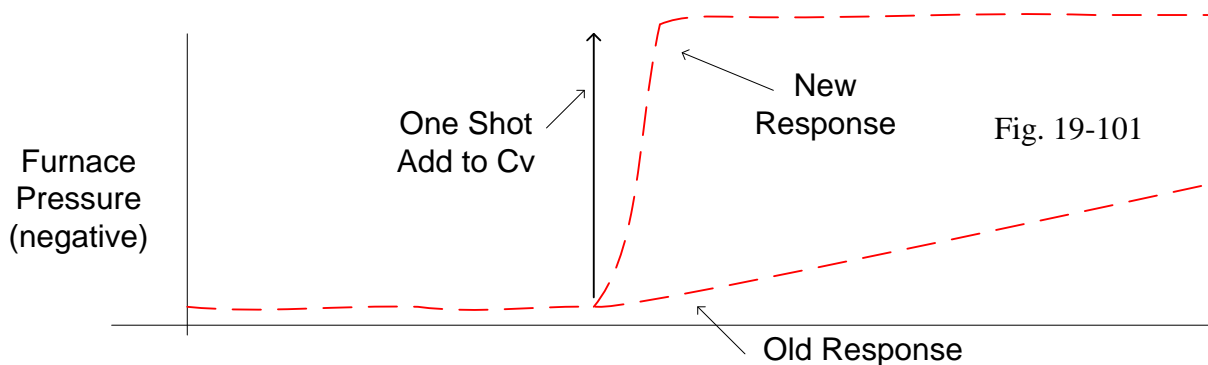
This is a major concern because the PID loop must respond in a much different manner in this circumstance than under normal operating conditions with the door closed. The fact that an event such as the door opening occurs helps to accomplish the control of this task. While not true feed forward, augmentation of the PID block will help offset the pressure upset and keep the flames pretty much inside the furnace. (Flames coming out the furnace tend to ignite grease from bearings causing grease fires around the furnace.)

To accomplish better pressure control, place a limit switch on the door and adjust the output of the PID block so the output will open the damper rapidly and then recover. The constant of the jump is a number that should be adjustable by an operator in the maintenance mode only.

When the door swings open, perform the following operation using a one-shot rung:

$$CV = CV + \text{constant}$$

This statement should be written only once to the CV. Use a one-shot circuit to add the constant to CV. The CV then is allowed to recover to its new value but from a new higher starting point as opposed to the original value. The value of the constant is the amount shown by the arrow below. This is a constant that is adjusted to fit the application. Once set, it should not be changed.



The response is a simulated response but makes the point that the response to a pressure change requires fast action to adjust to the conditions of the door opening. A change in the CV provides this type of change. The change in CV will start the adjustment procedure and trick the PID tuning parameters into responding to the new situation quickly instead of a slow acting controller as would be the case for the regular control of oven pressure.

While the addition of a small incremental value to CV may be considered a trick on the PID block, it is important to note that such an action may be accomplished in the PLC very easily. Ladder logic accommodates this type of programming through the use of one-shot ladder logic and math functions. This type of change to the PID block provides quick response to an upset outside the normal range of the PID block's algorithm. The actual move may only be able in the manual mode. To move to manual, change the CV and then move back to auto is recommended for this action to occur successfully.

Introducing Flow Diagrams (P&ID's)

Processes are described using flow diagrams. Symbols for diagrams are defined by the organization – International Society of Automation, aka Instrumentation, Systems, and Automation Society, aka Instrument Society of America, aka ISA. Letter codes are written in circles representing various devices that control a process. For instance, FIC represents Flow Rate, Indicator, Controller. **Any three-letter code with C as the final letter represents a PID controller.** First a review of the letter codes used to configure an instrument:

Letter	First Position	Succeeding Positions
A	Analysis	Alarm
B	Burner Flame	
C	Conductivity	Control
D	Density / Differential	
E	Voltage	
F	Flow Rate / Ratio	
G	Gaging	Glass
H	Hand	High
I	Current	Indicate
J	Power / Scan	
K	Time	
L	Level	Light / Low
M	Moisture	Middle/ Manual
N	Choice	
O	Choice	
P	Pressure	
R	Radioactivity	Record
S	Speed	Switch
T	Temperature	Transmit
V	Viscosity	Valve
W	Weight	Well
X	Interlock	
Y	Choice	Relay
Z	Position	Drive

Process Measurement	Element Type	Element	Transmitter	Indicator	Indicator controller	Controller	Ratio Controller	Recorder
Analysis	Code	E	T	I	IC	C	FC	R
	A	AE	AT	AI	AIC	AC	AFC	AR
Conductivity	C	CE	CT	CI	CIC	CC	CFC	CR
Density	D	DE	DT	DI	DIC	DC	DFC	DR
Voltage	E	EE	ET	EI	EIC	EC	EFC	ER
Flow	F	FE	FT	FI	FIC	FC	FFC	FR
Dimension	G	GE	GT	GI	GIC	GC	GFC	GR
Hand	H	HE	HT	HI	HIC	HC	HFC	HR
Current	I	IE	IT	II	IIC	IC	IFC	IR
Time	K	KE	KT	KI	KIC	KC	KFC	KR
Level	L	LE	LT	LI	LIC	LC	LFC	LR
Humidity	M	ME	MT	MI	MIC	MC	MFC	MR
Power	N	NE	NT	NI	NIC	NC	NFC	NR
Pressure	P	PE	PT	PI	PIC	PC	PFC	PR
Delta Pressure	dP	dPE	dPT	dPI	dPIC	dPC	dPFC	dPR
Quantity	Q	QE	QT	OI	OIC	QC	QFC	QR
Radioactivity	R	RE	RT	RI	RIC	RC	RFC	RR
Speed	S	SE	ST	SI	SIC	SC	SFC	SR
Temperature	T	TE	TT	TI	TIC	TC	TFC	TR
Delta Temperature	dT	dTE	dTT	dTI	dTIC	dTC	dTFC	dTR
Viscosity	V	VE	VT	VI	VIC	VC	VFC	VR
Weight	W	WE	WT	WI	WIC	WC	WFC	WR
Vibration	Y	YE	YT	YI	YIC	YC	YFC	YR
Position	Z	ZE	ZT	ZI	ZIC	ZC	ZFC	ZR

The table above contains descriptions of various types of transmitters, indicators, controllers and recorders. Most PID blocks are used to program controller items. There is a one-to-one programming transfer for most xIC (various, Indicating Controller) or xC controllers.

Process and Instrumentation Drawings (P&ID) are formalized drawings of a process explaining flow and movement of material. It is important to know the symbols for this type of drawing. It is also important to be able to understand the functionality of the devices on the drawing so the engineer or technologist can program the process on the PLC or other computer.

It is also hoped that down the road, the engineer or technologist is allowed to design the P&ID for others. The programmer usually understands the process as well as anyone and has insight into the complexities of the process and should be allowed to take responsibility for design of the P&ID.

A note about PID vs P&ID: Of course, the similarities are glaring. PID refers to the control block Proportional Integral Derivative, a control algorithm. P&ID refers to Process and Instrumentation Drawings. Some refer to them as Piping and Instrumentation Drawings.

Process Measurement	Element Type	Hand Switch	Hand Valve	Totalizer	Indicating Totalizer	Solenoid Valve	Control Valve	Calculation
	Code	HS	HV	Q	IQ	XV	V	Y
Analysis	A	AHS	AHV	AQ	AIQ	AXV	AV	AY
Conductivity	C	CHS	CHV	CQ	CIQ	CXV	CV	CY
Density	D	DHS	DHV	DQ	DIQ	DXV	DV	DY
Voltage	E	EHS	EHV	EQ	EIQ	EXV	EV	EY
Flow	F	FHS	FHV	FQ	FIQ	FXV	FV	FY
Dimension	G	GHS	GHV	GQ	GIQ	GXV	GV	GY
Hand	H	HHS	HHV	HQ	HIQ	HXV	HV	HY
Current	I	IHS	IHV	IQ	IIQ	IXV	IV	IY
Time	K	KHS	KHV	KQ	KIQ	KXV	KV	KY
Level	L	LHS	LHV	LQ	LIQ	LXV	LV	LY
Humidity	M	MHS	MHV	MQ	MIQ	MXV	MV	MY
Power	N	NHS	NHV	NQ	NIQ	NXV	NV	NY
Pressure	P	PHS	PHV	PQ	PIQ	PXV	PV	PY
Delta Pressure	dP	dPHS	dPHV	dPQ	dPIQ	dPXV	dPV	dPY
Quantity	Q	QHS	QHV	QQ	QIQ	QXV	QV	QY
Radioactivity	R	RHS	RHV	RQ	RIQ	RXV	RV	RY
Speed	S	SHS	SHV	SQ	SIQ	SXV	SV	SY
Temperature	T	THS	THV	TQ	TIQ	TXV	TV	TY
Delta Temperature	dT	dTHS	dTHV	dTQ	dTIQ	dTXV	dTV	dTY
Viscosity	V	VHS	VHV	VQ	VIQ	VXV	VV	VY
Weight	W	WHS	WHV	WQ	WIQ	WXV	WV	WY
Vibration	Y	YHS	YHV	YQ	YIQ	YXV	YV	YY
Position	Z	ZHS	ZHV	ZQ	ZIQ	ZXV	ZV	ZY

Devices such as hand switches, valves and some electronic devices such as totalizers and calculation elements are described here. Most calculation elements are executed inside the computer and algorithms become much too difficult to describe on the P&ID. The designer of the P&ID is free to decide how much of the calculation information is to be included on the drawing.

Devices such as those of the table above are primarily used for checking position of switches and for various types of alarm. It is not uncommon to assign switches for end-of-travel on analog devices. With most analog systems, there is an alarm reserved for both low and low-low. Low-low is the signal that is just past low and should be attached to an alarm as well as shut-off logic. The same logic is used for high and high-high. The inner alarm is the low or high alarm bit and the low-low and high-high are the outer or fail-safe alarm.

	Element Type	Ratio Calculation	Switch Low	Switch High	Alarm Low	Alarm Low Low	Alarm High	Alarm High
Process Measurement	Code	FY	SL	SH	AL	ALL	AH	AHH
Analysis	A	AFY	ASL	ASH	AAL	AALL	AAH	AAHH
Conductivity	C	CFY	CSL	CSH	CAL	CALL	CAH	CAHH
Density	D	DFY	DSL	DSH	DAL	DALL	DAH	DAHH
Voltage	E	EFY	ESL	ESH	EAL	EALL	EAH	EAHH
Flow	F	FFY	FSL	FSH	FAL	FALL	FAH	FAHH
Dimension	G	GFY	GSL	GSH	GAL	GALL	GAH	GAHH
Hand	H	HFY	HSL	HSB	HAL	HALL	HAH	HAHH
Current	I	IFY	ISL	ISH	IAL	IALL	IAH	IAHH
Time	K	KFY	KSL	KSH	KAL	KALL	KAH	KAHH
Level	L	LFY	LSL	LSH	LAL	LALL	LAH	LAHH
Humidity	M	MFY	MSL	MSH	MAL	MALL	MAH	MAHH
Power	N	NFY	NSL	NSH	NAL	NALL	NAH	NAHH
Pressure Delta Pressure	P dP	PFY dPFY	PSL dPSL	PSH dPSH	PAL dPAL	PALL dPALL	PAH dPAH	PAHH dPAHH
Quantity	Q	QFY	QSL	QSH	QAL	QALL	QAH	QAAH
Radioactivity	R	RFY	RSL	RSH	RAL	RALL	RAH	RAHH
Speed	S	SFY	SSL	SSH	SAL	SALL	SAH	SAHH
Temperature Delta Temperature	T dT	TFY dTfY	TSL dTSL	TSH dTSH	TAL dTAL	TALL dTALL	TAH dTAH	TAHH dTAAH
Viscosity	V	VFY	VSL	VSH	VAL	VALL	VAH	VAHH
Weight	W	WFY	WSL	WSH	WAL	WALL	WAH	WAHH
Vibration	Y	YFY	YSL	YSH	YAL	YALL	YAH	YAAH
Position	Z	ZFY	ZSL	ZSH	ZAL	ZALL	ZAH	ZAAH

These tables demonstrate the breadth of labeling that can be included on a device. The devices are also numbered and contain a 3 or 4 digit number in addition to the device type name. These numbers are usually assigned sequentially and are placed on a metal tag that is attached to the device itself. In the plant, one should be able to find a device, then find its metal tag, and find the reference to the device on the P&ID. Names of devices are used on electrical drawings as well as on the P&ID. If a device is referenced as a flow transmitter and numbered 087, then FT-087 is referenced on all drawings using the same name.

The design of a P&ID may start with a senior engineer familiar with the process. Other sources for P&ID's are reference books such as the Liptak reference handbook Process Control. Texts and company reference drawings are good sources for a starting point for a new P&ID. Of course, names such as those listed above are to be used in defining the devices used in the process.

For example, the flow drawing of level control using flow would be drawn as follows:

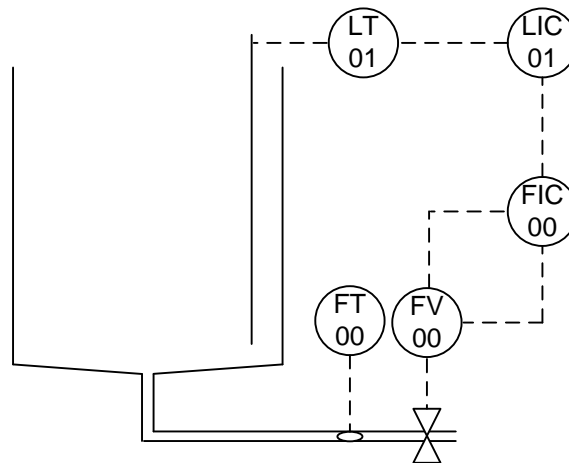


Fig. 19-102

While in many P&ID's the symbols are kept as simple as possible, there is delineation in the ISA S5.1 standard for location as well as type of device. These symbol types are shown below:

A discrepancy between the symbols and the usage of the devices is that the PLC has traditionally been viewed as only useful for some safety circuits and for discrete control. The PLC has taken over much of the analog control and more logically fits the computer function as well as the traditional PLC role. The device providing control has changed dramatically over the years from discrete hardwired controllers to DCS systems and finally to PLC analog systems. The primary rationale for using the PLC in analog situations is cost.

For instance, the door-mounted limit switch on the oven above would be drawn as:

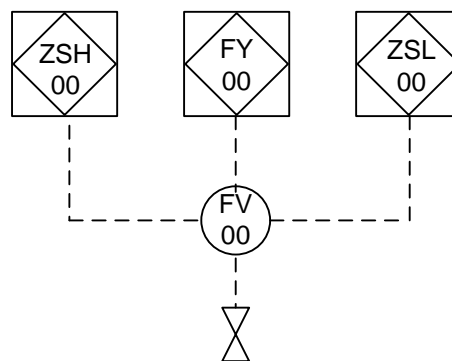
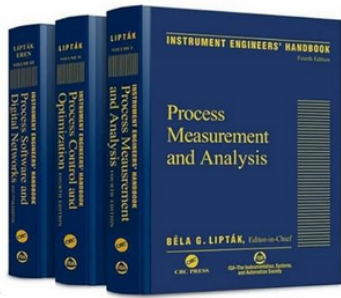


Fig. 19-103

The following cut from Amazon shows one of the most definitive set of texts on use of the P&ID symbols in industry. Bela Liptak gives an exhausting supply of P&ID drawings with explanation for a huge variety of processes. These texts, especially the middle one, Process Control and Optimization, has been used by this author to program a process. These texts are invaluable for someone involved in process design and implementation, either with a PLC or other control method. Various examples from his Process Control and Optimization Text are given below as practice problems for building the PLC program to control the various processes.



Instrument Engineers Handbook, Fourth Edition, Three Volume Set 4th Edition



by Bela G. Liptak (Author)

4.1 ★★★★★ (3)

[See all formats and editions](#)

This set consists of:

- *Instrument Engineers' Handbook, Fourth Edition, Volume One: Process Measurement and Analysis* (Published June 2003, ISBN 9780849310836)
- *Instrument Engineers' Handbook, Fourth Edition, Volume Two: Process Control and Optimization* (Published September 2005, ISBN 9780849310812)
- *Instrument Engineers' Handbook, Fourth Edition, Volume Three: Process Software and Digital Networks* (Published August 2011, ISBN 9781439817766)

Unsurpassed in its coverage, usability, and authority, the latest edition to Béla G. Lipták's three-volume **Instrument Engineers' Handbook** continues to serve as the premier reference for instrument engineers around the world. The acclaimed "bible" of instrument engineering helps users select and implement hundreds of measurement and control instruments and analytical devices. It also aids in the design of cost-effective process control systems that optimize production and maximize safety.

Follow the author



Béla G. Lipták

Follow

Example Programming for P&ID:

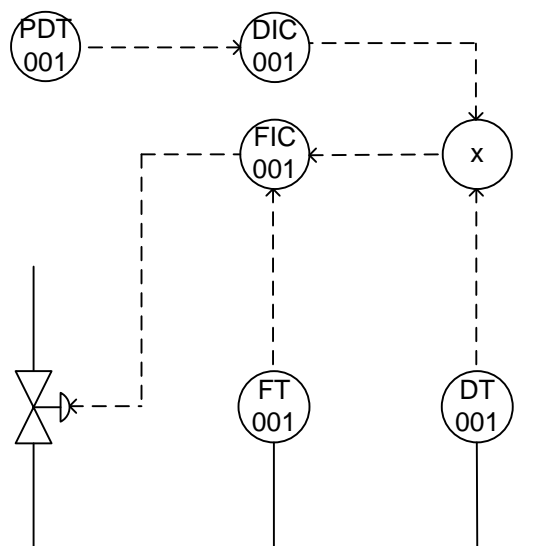


Fig. 19-104a

The P&ID above is used to generate a PLC ladder diagram as follows:

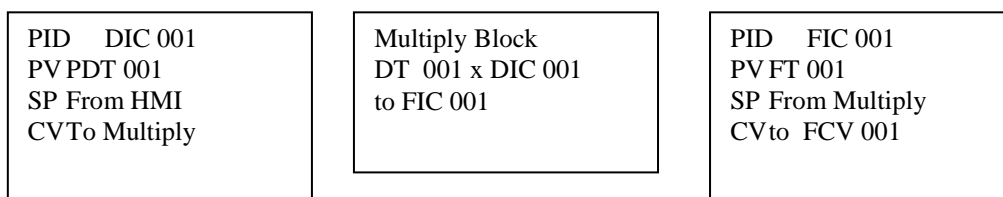


Fig. 19-105

Test Question:

Using either the PID blocks from A-B or Siemens, provide a program that will work in auto mode for the following P&ID. Define and use variables as inputs, outputs and internal variables as necessary. Describe these variables in a table. ORP refers to Oxidation-Reduction potential.

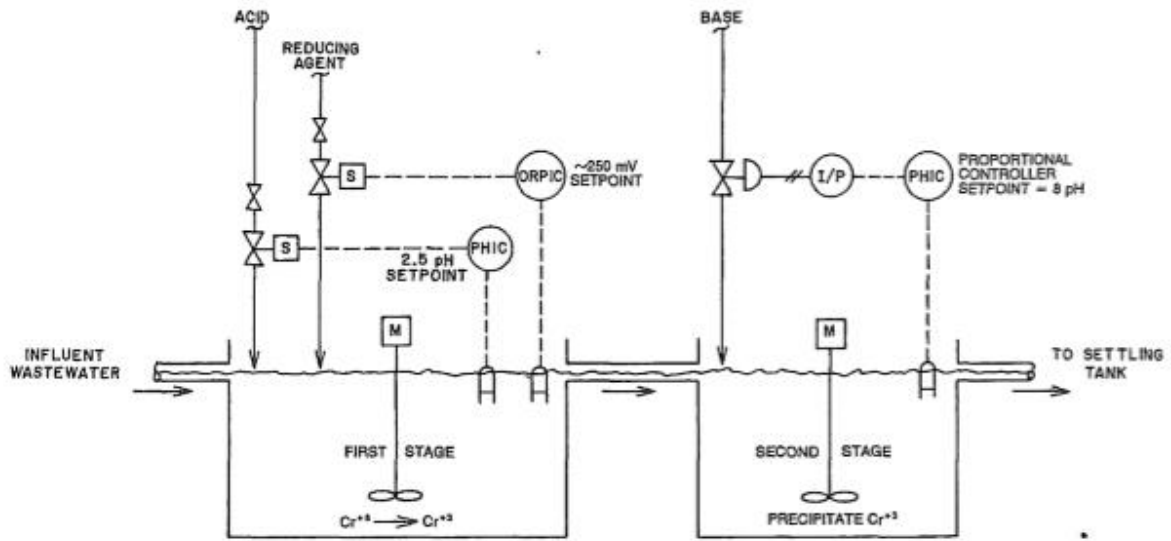


FIG. 8.21b
Continuous chrome treatment.

Fig. 19-104b

Find PID controllers, assign numbers to each and identify any other math relationships. This is the auto control portion of the program controlling this P&ID.

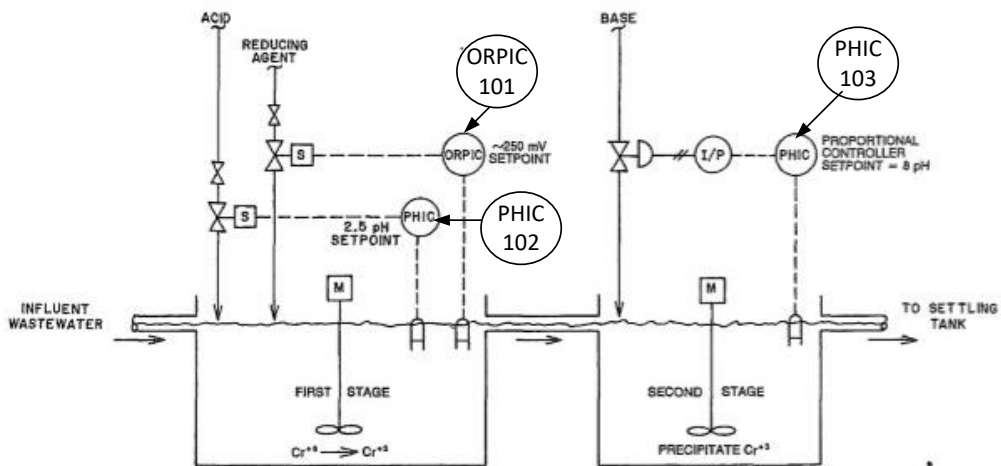
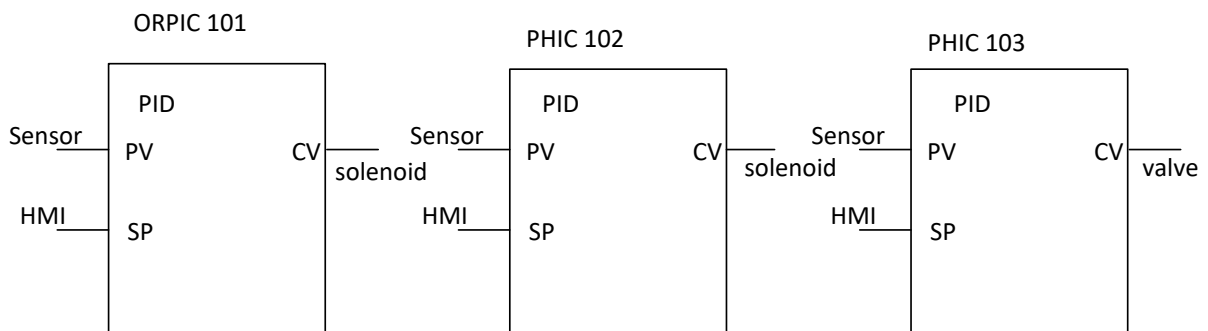


FIG. 8.21b
Continuous chrome treatment.



Test Question:

Using either the PID blocks from A-B or Siemens, provide a program that will work in auto mode for the following P&ID. Use variables as inputs, outputs and internal variables as necessary. Describe these variables in a table. PV's and CV's are marked in the diagram. Unmarked SP's are setpoints from the HMI screen. Ignore sqrt operation. Assume that FT's give a flow rate and the sqrt function is performed at the transmitter, not the controller. (AE – Analysis Element, AR – Analysis Recorder, ASL – Analysis Switch Low, ASH – Analysis Switch High, AAL – Analysis Alarm Low, AAH – Analysis Alarm High)

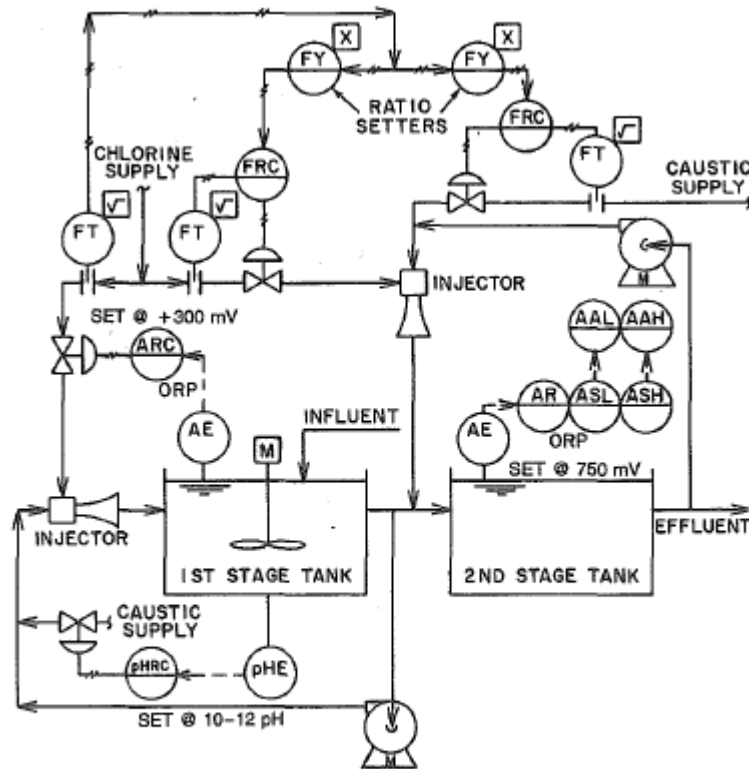


Fig. 19-104c

FIG. 8.29b

Continuous oxidation of cyanide waste with chlorine. Influent here has continuous constant flow rate and variable quality.

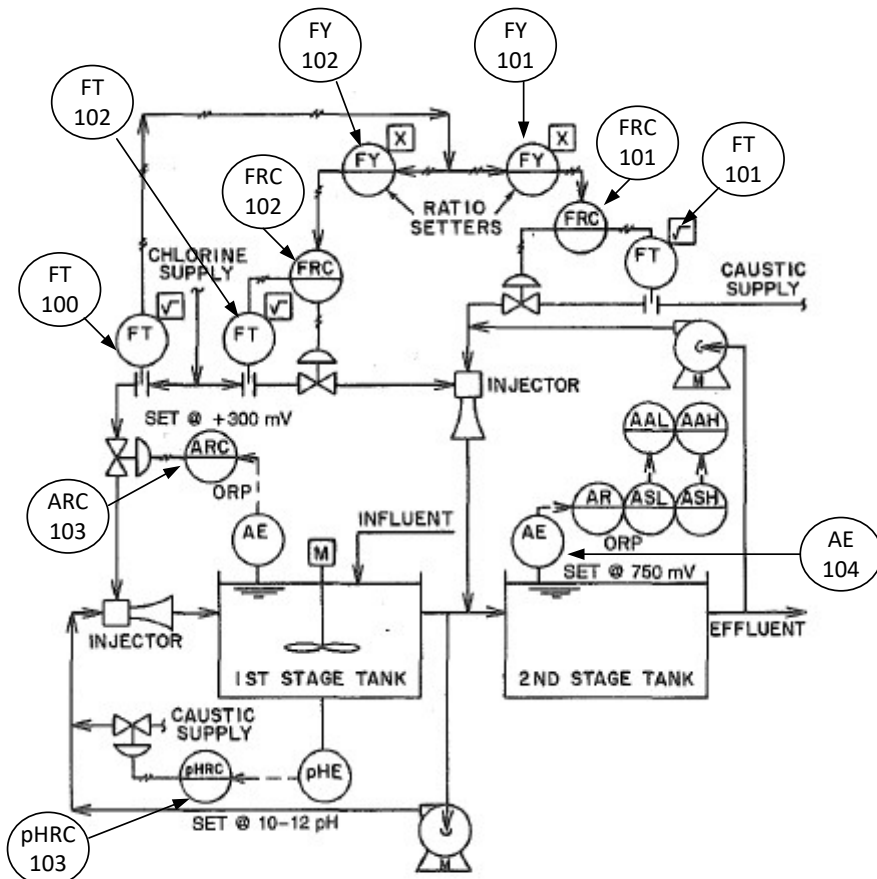
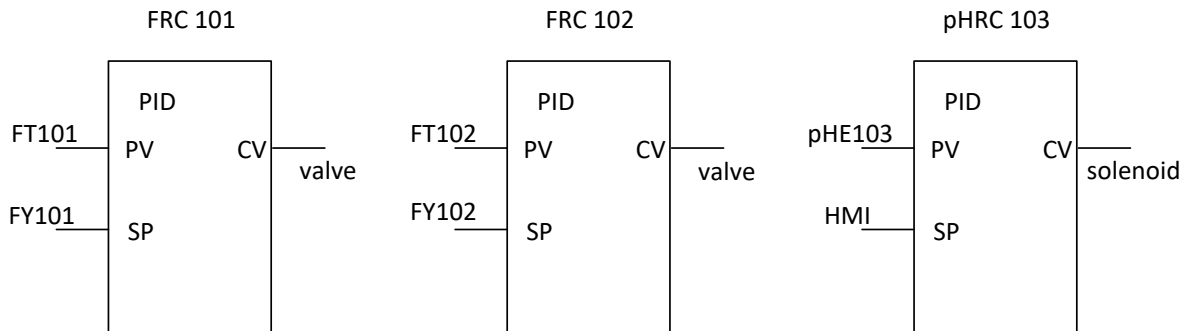


Fig. 19-104d

FIG. 8.29b

Continuous oxidation of cyanide waste with chlorine. Influent here has continuous constant flow rate and variable quality.



$$FY101 = FT100 * \text{const1}$$

$$FY102 = FT100 * \text{const2}$$

$$AAH104 = AE104 > \text{high alarm const}$$

$$AAL104 = AE104 < \text{low alarm const}$$

Example Programming for P&ID (The PLC program is left as an exercise for the student):

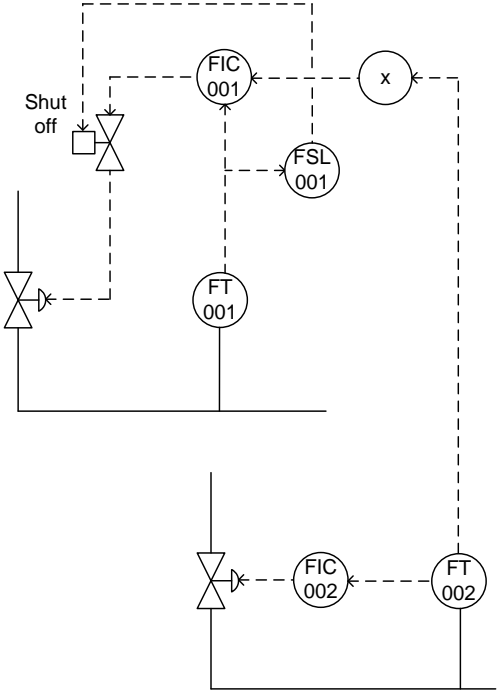


Fig. 19-105

The following three diagrams show more extensive P&ID drawings for a complete system.

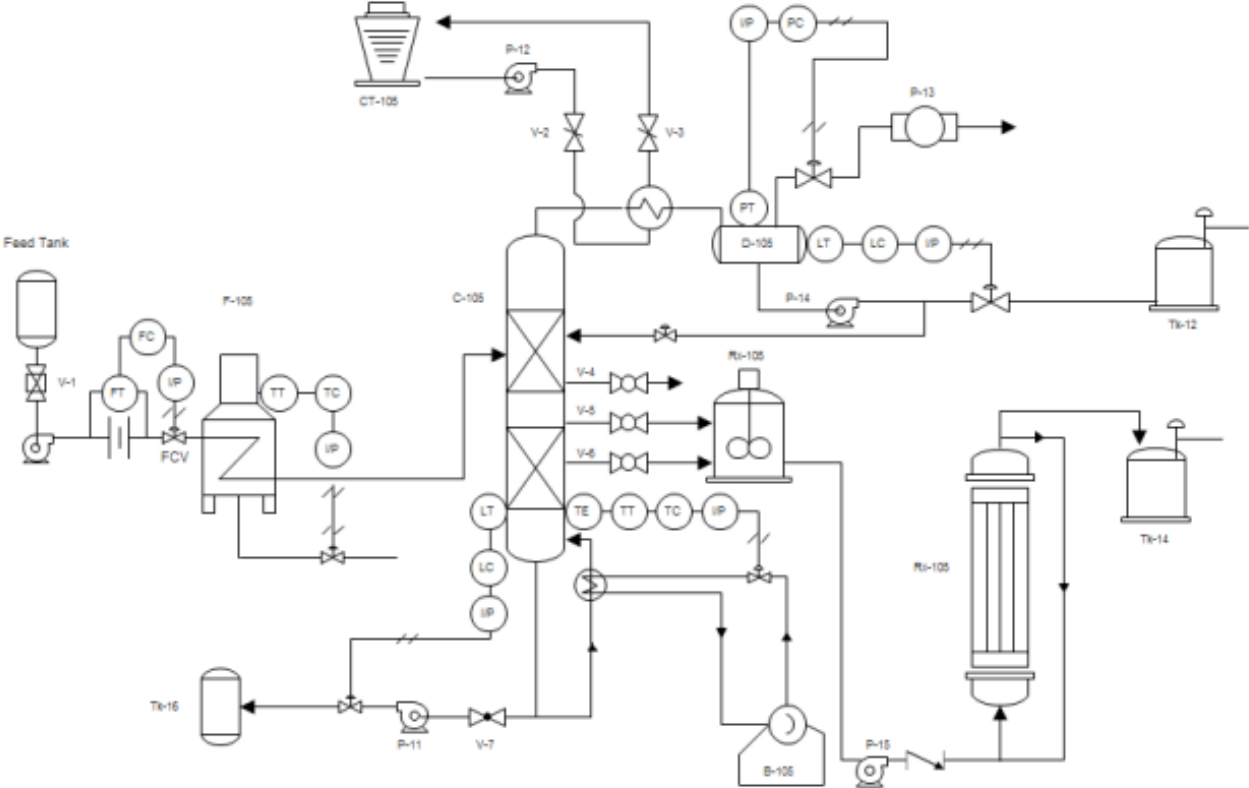


Fig. 19-106a

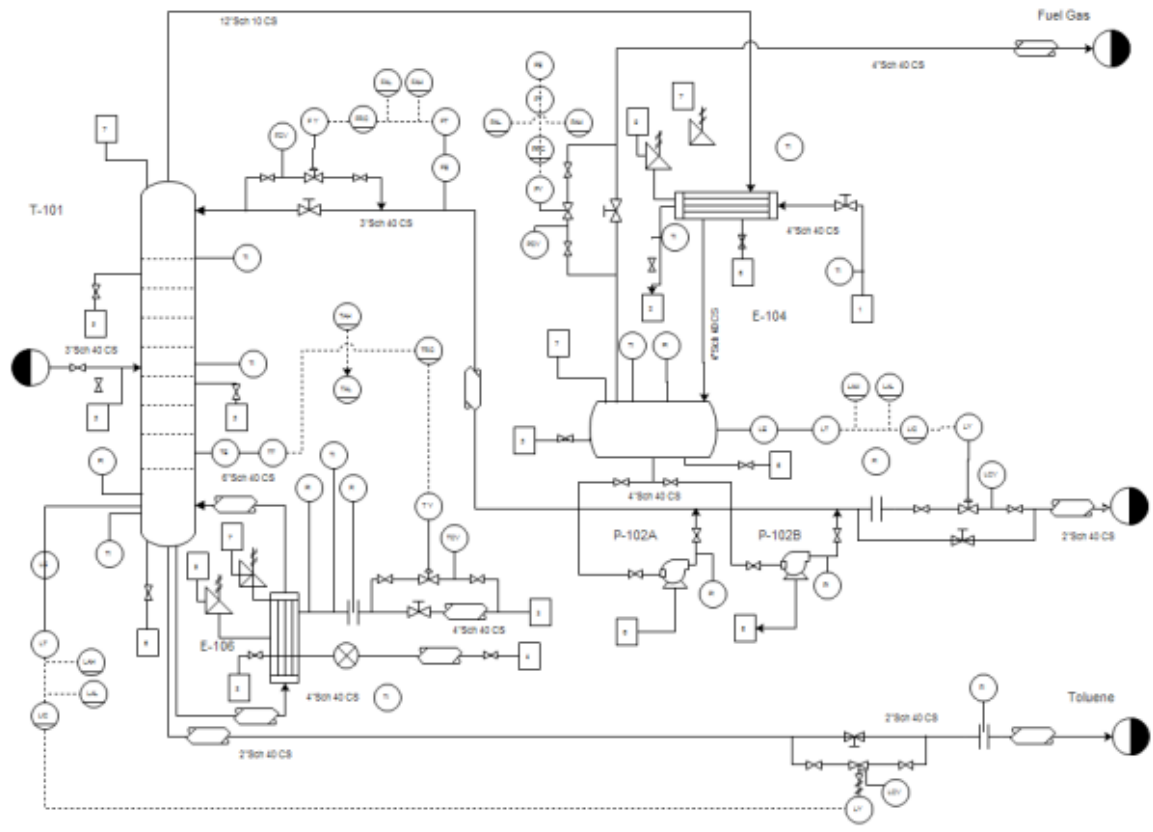


Fig. 19-106b

This diagram shows the system for generation of steam.

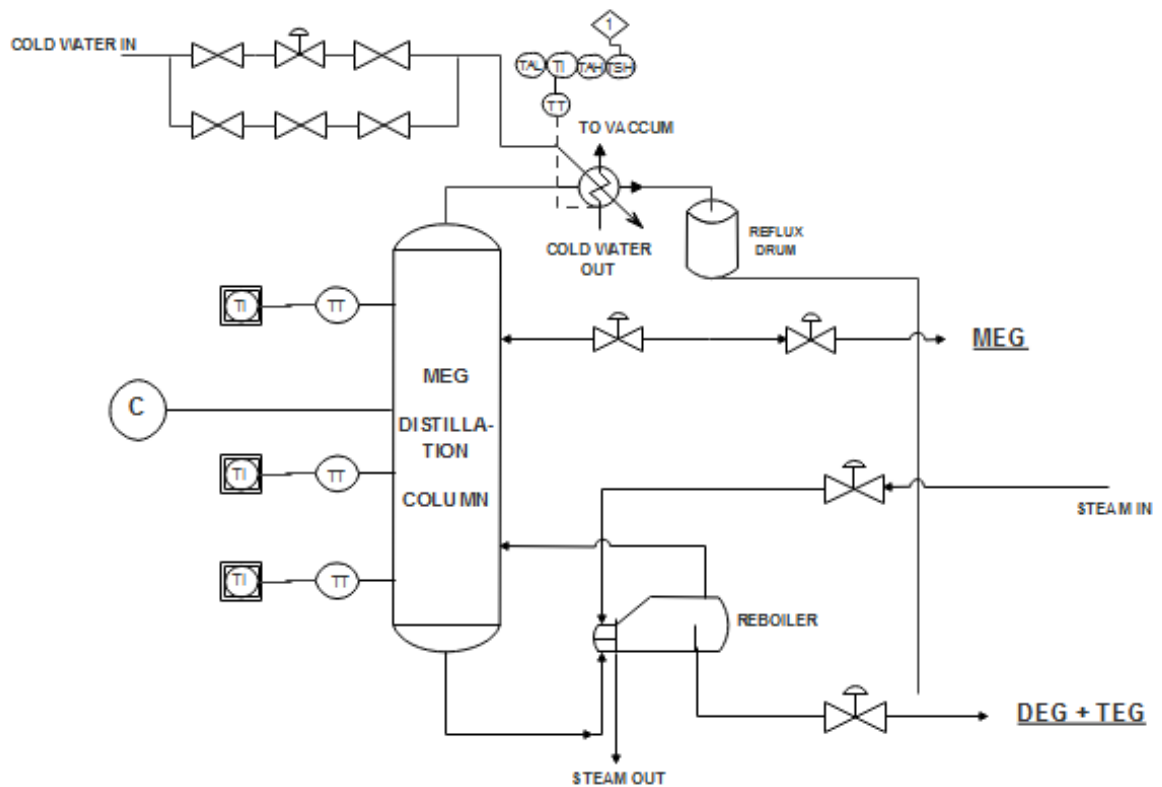


Fig. 19-106c

Specifications for P&ID design and the design of a process may be found at the ISA website. The following list is a partial list of design specifications used in constructing a modern process.

ANSI/ISA-75.01.01-2002 (60534-2-1 Mod)	Flow Equations for Sizing Control Valves
ANSI/ISA-75.02-1996	Control Valve Capacity Test Procedures
ANSI/ISA-TR75.04.01-1998	Control Valve Position Stability
ANSI/ISA-75.05.01-2000 (R2005)	Control Valve Terminology
ISA-75.07-1997	Laboratory Measurement of Aerodynamic Noise Generated by Control Valves
ANSI/ISA-75.08-1999	Installed Face-to-Face Dimensions for Flanged Clamp or Pinch Valves
ANSI/ISA-75.08.01-2002	Face-to-Face Dimensions for Integral Flanged Globe-Style Control Valve Bodies (Classes 125, 150, 250, 300, and 600)
ANSI/ISA-75.08.02-2003	Face-to-Face Dimensions for Flangeless Control Valves (Classes 150, 300, and 600)
ANSI/ISA-75.08.03-2001	Face-to-Face Dimensions for Socket Weld-End and Screwed-End Globe-Style Control Valves (Classes 150, 300, 600, 900, 1500, and 2500)
ANSI/ISA-75.11.01-1985 (R2002)	Inherent Flow Characteristic and Rangeability of Control Valves
ISA-75.13-1996	Method of Evaluating the Performance of Positioners with Analog Input Signals and Pneumatic Output
ISA-75.17-1989	Control Valve Aerodynamic Noise Prediction
ANSI/ISA-75.19.01-2001	Hydrostatic Testing of Control Valves
ISA-RP75.21-1989 (R1996)	Process Data Presentation for Control Valves
ANSI/ISA-75.22-1999	Face-to-Centerline Dimensions for Flanged Globe-Style Angle Control Valve Bodies (ANSI Classes 150, 300, and 600)
ISA-RP75.23-1995	Considerations for Evaluating Control Valve Cavitation
ANSI/ISA-75.25.01-2000	Test Procedure for Control Valve Response Measurement from Step Inputs
ANSI/ISA-TR75.25.02-2000	Control Valve Response Measurement from Step Inputs
ANSI/ISA-75.26.01-2006	Control Valve Diagnostic Data Acquisition and Reporting

Partial List of ANSI-ISA Specifications for Process Control

Using Visio for P&ID Drawings

Microsoft's Visio is useful for a flow-diagram generation and has provision for generating the P&ID drawings similar to those described above. An example below gives a description of how the drawing type is chosen in Visio.

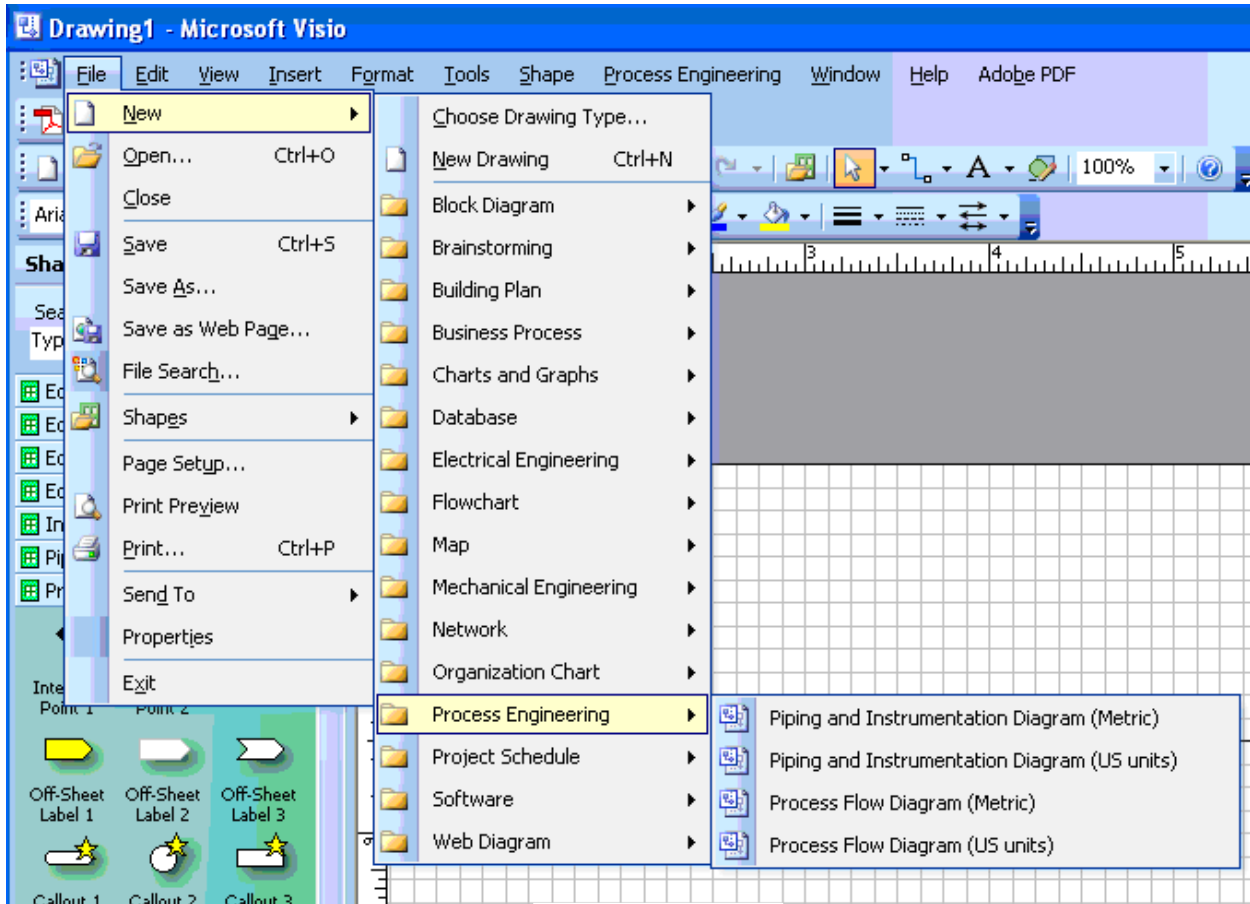


Fig. 19-107

The elements are automatically connected with piping (lines) and names are attached in sequential order.

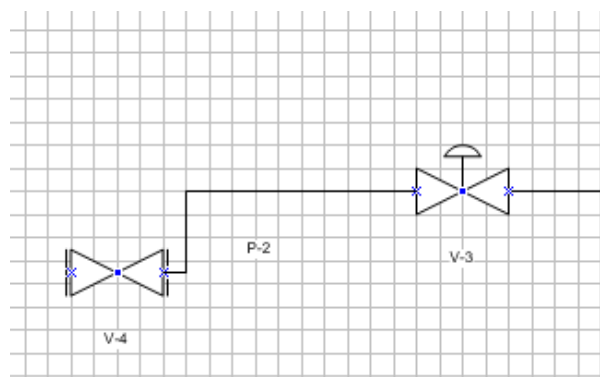


Fig. 19-108

Below the diagrams show a number of different pre-drawn figures for use in a P&ID. The diagrams follow ISA symbol standards.

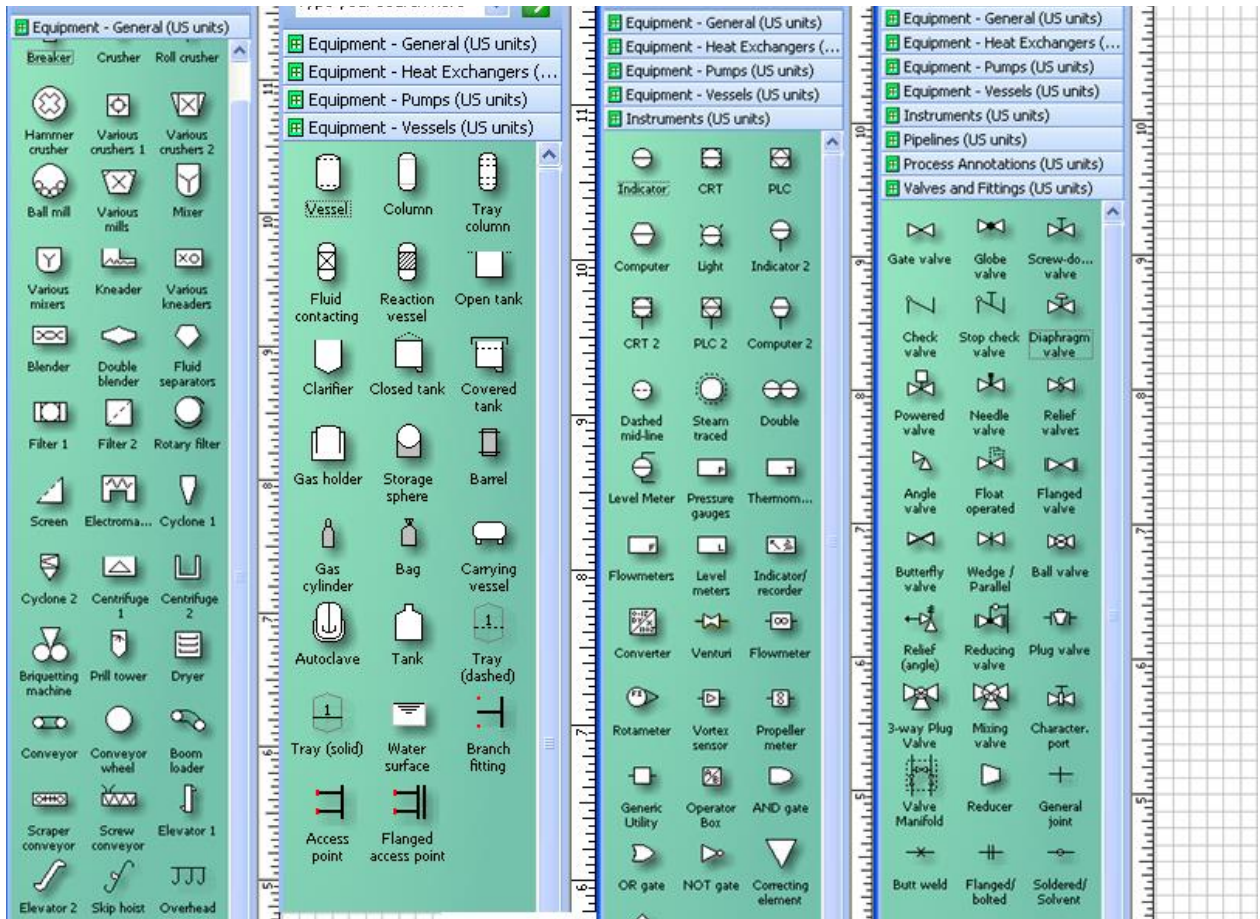


Fig. 19-109

The PIDE Function from Allen-Bradley

The PIDE is only available as a function block. Like the PID instruction, it is best to set it up in its own periodic task. The period of the task automatically becomes the sample rate of the PID loop. Just make sure when adding the new routine to the task to select Type as “Function Block Diagram – FBD”.

The PIDE (Enhanced PID) is an Allen-Bradley Logix5000 function block that improves on the standard PID found in all their controllers. First impressions tend to be intimidating. The advanced instruction boasts the following:

1. It uses the velocity form of the PID algorithm. This is especially useful for adaptive gains or multiloop selection
2. Control of the instruction can be switched between Program and Operator modes
3. Better support for cascading and ratio control
4. Built-in autotuner
5. Support for different timing modes
6. More limiting and fault handling selections

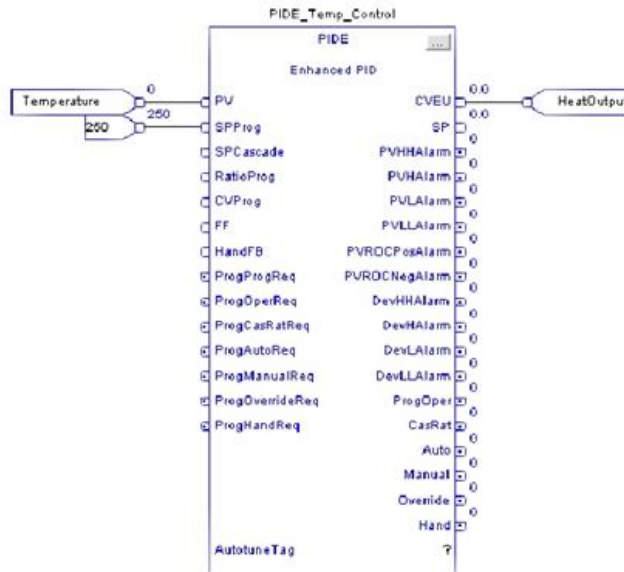


Fig. 19-110

Once a function block is created, the program tags for the function block must be created. With later versions of RSLogix 5000, the set-up box below gives a view of the variables required.

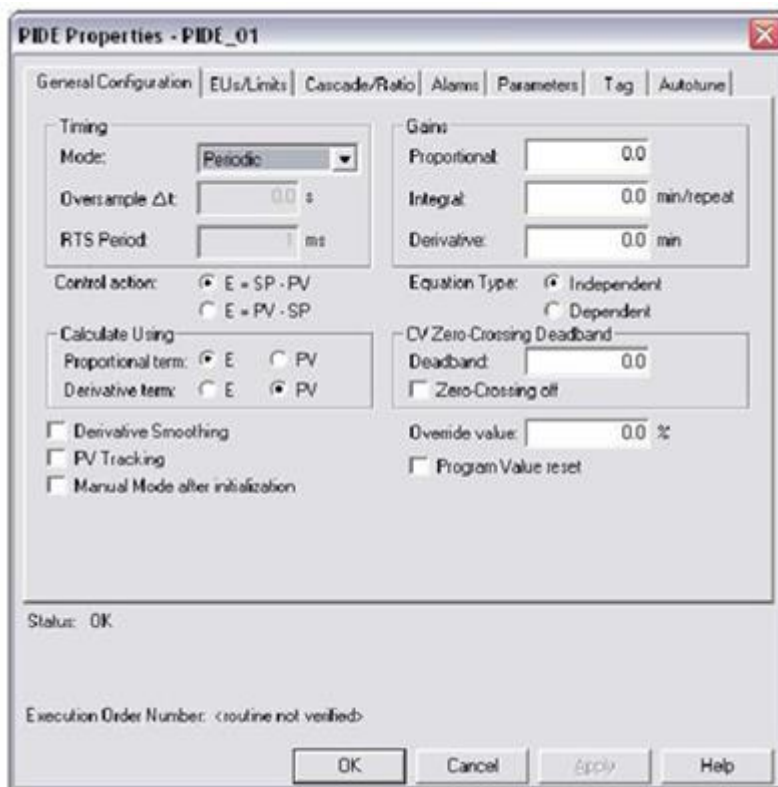


Fig. 19-111

Instead of control of control using the MultiState Button and the logic shown above, the PIDE shares program and operator control with control bits in the PIDE block. The following bits partially describe this control:

- .ProgProgReq Program request to go to Program Control
- .ProgOperReq Program request to go to Operator Control
- .OperProgReq Operator request to go to Program Control
- .OperOperReq Operator request to go to Operator Control

Operating Modes for the PIDE instruction include:

Manual:

While in Manual mode the instruction does not compute the change in CV. The value of CV is determined by the control. If in Program control, $CV = CV_{Prog}$ and if in Operator control, $CV = CV_{Oper}$. Select Manual mode using either OperManualReq or ProgManualReq. The Manual output bit is set when in Manual mode.

Auto:

While in Auto mode the instruction regulates CV to maintain PV at the SP value. If in program control, $SP = SP_{Prog}$ and if in Operator control, $SP = SP_{Oper}$. Select Auto mode using either OperAutoReq or PRogAutoReq. The Auto output bit is set when in Auto mode.

Cascade/Ratio:

While in Cascade/Ratio mode the instruction computes the change in CV. The instruction regulates CV to maintain PV at either SPCascade value or the SPCascade value multiplied by the Ratio value. SPCascade comes from either the CVEU of a primary PID loop for cascade control or from the “uncontrolled” flow of a ratio-controlled loop. Select Cascade/Ratio mode using either OperCasRatReq or ProgCasRatReq. The CasRat output bit is set when in Cascade/Ratio mode.

Override:

While in Override mode, the instruction does not compute the change in CV. $CV = CV_{Override}$, regardless of the control mode. Override mode is typically used to set a “safe state” for the PID loop. Select Override mode using ProgOverrideReq. The Override output bit is set when in Override mode.

Hand:

While in Hand mode, the PID algorithm does not compute the change in CV. $CV = Hand_{FB}$, regardless of the control mode. Hand mode is typically used to indicate that control of the final control element was taken over by a field hand/auto station. Select Hand mode using ProgHandReq. The Hand output but is set when in Hand mode.

The example below is of a PIDE block in FBD programming language:

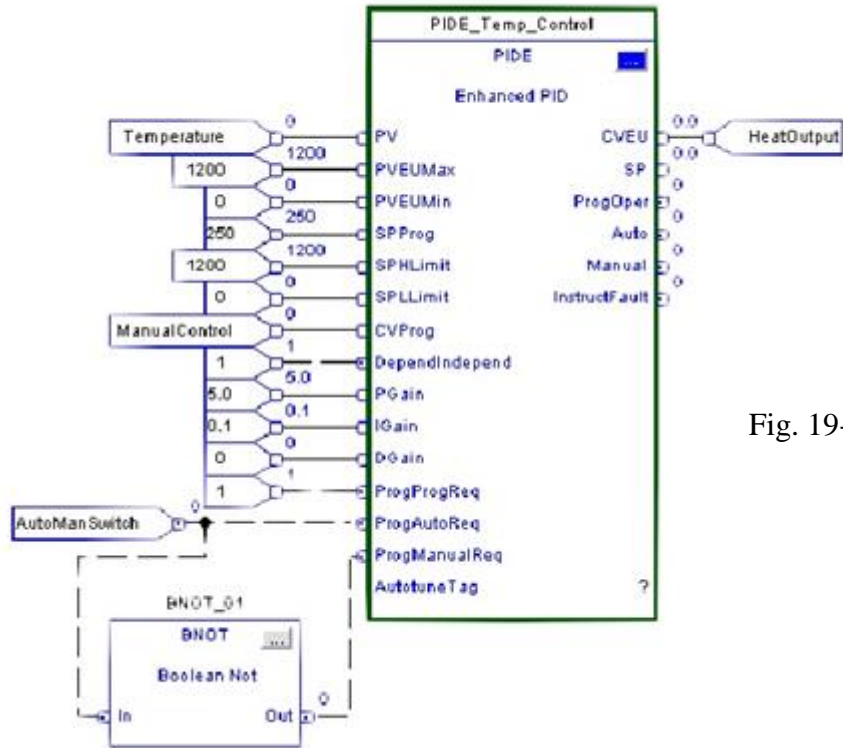


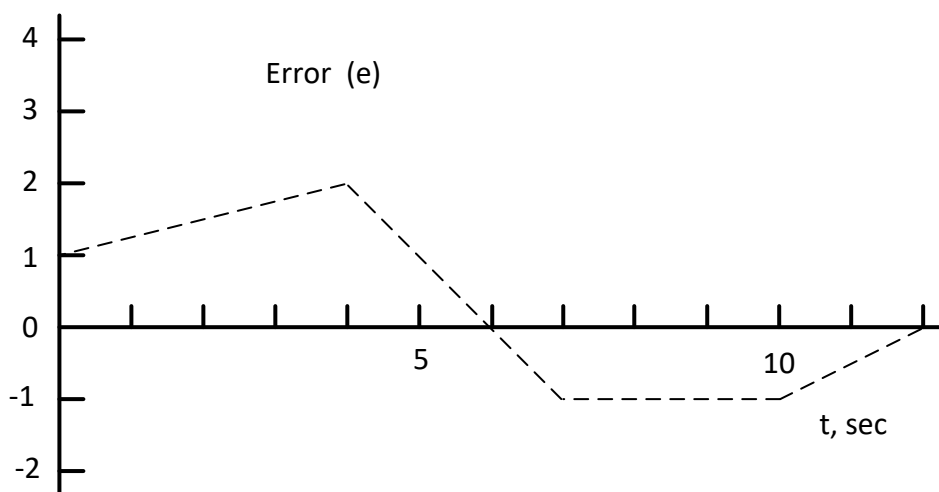
Fig. 19-112

The following problems give an example of how the PID algorithm is executed as time increases with a known error (e). The output C_v is calculated at each change of (e).

Test Problem:

A PI controller has a gain of 0.6, an integral action rate of 0.6 s^{-1} , and an initial value of v_o of 15%. The graph is of the error (e). Determine the value of the controller from $t = 0$ to $t = 12 \text{ sec}$ and graph the result below:

(remember that $C_v = P \cdot e + P \cdot I \cdot \sum e \cdot \Delta t + v_o$)



$$t=0$$

$$C_v = 15 + .6 \times 1 = 15.6$$

$$t=4$$

$$C_v = 15 + .6 \times 2 + .6 \times .6 \times \left(4 \times 1 + \frac{1}{2} \times 4 \times 1 \right) = 18.36$$

Area under curve

$$t=7$$

$$C_v = 15 + .6(-1) + .6 \times .6 \times (1.5 + 6) = 17.1$$

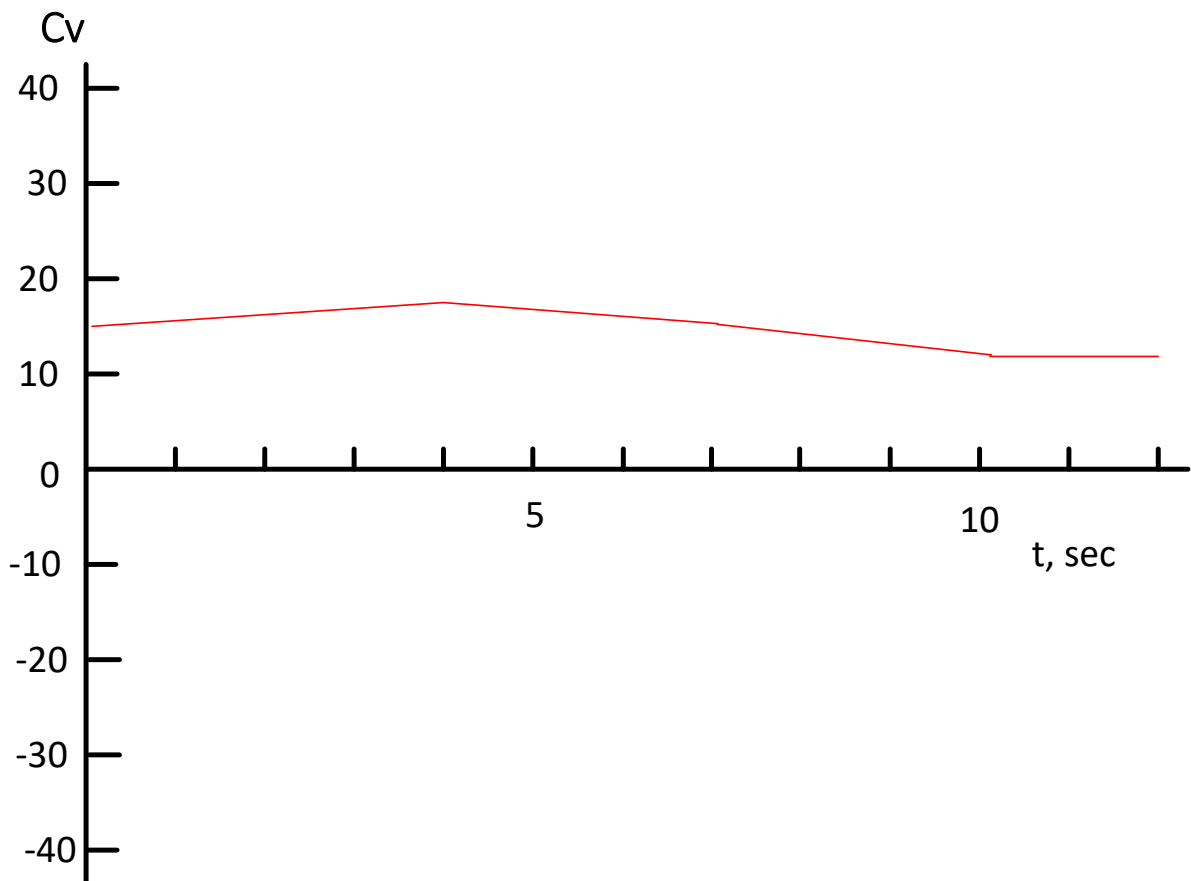
$$t=10$$

$$C_v = 15 + .6(-1) \times .6 \times .6 \times (1.5 + 6 - 3) = 16.02$$

$$t=12$$

$$C_v = 15 + 0 + .6 \times .6 \times (1.5 + 6 - 3 - 1) = 16.26$$

Graph

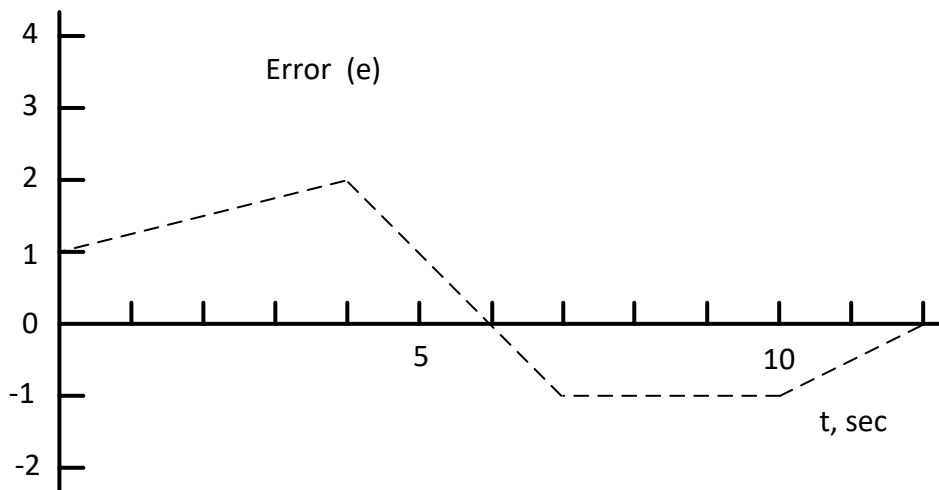


Remember the change in the integral term if the PID algorithm is moved from Auto to Manual. The integral term is cancelled (=0). If then the PID algorithm is changed back to Auto, the former sum is reset to 0. There is an assumption that the value of the output Cv does not change during the A-M-A change. This is commonly referred to as Bumpless Transfer. Bumpless transfer must be programmed in the PLC and is non-trivial. However, it is needed in most applications. The problem above is worked again here with an A-M-A pulse introduced to reset the sum of (e). The results show the difference with the problem above.

Test Problem:

A PI controller has a gain (P) of 0.6, an integral action rate (I) of 0.6 s^{-1} , and an initial value of v_o of 15%. The graph is of the error (e). At $t = 9$, a pulse occurs that drives the controller from auto to manual and then back to auto in a few msec. Determine the value of the controller (Cv) from $t = 0$ to $t = 12$ sec and graph the result below: (e is unitless and Cv is in percent)

(remember that $Cv = P \cdot e + P \cdot I \cdot \sum e \cdot \Delta t + v_o$)



$$t=0$$

$$C_v = 15 + .6 \times 1 = 15.6$$

$$t=4$$

$$C_v = 15 + .6 \times 2 + .6 \times .6 \times (4) = 18.36$$

$$t=7$$

$$C_v = 15 + .6(-1) + .6 \times .6 \times (1.5 + 6) = 17.1$$

$$t=9$$

$$C_v = 15 + .6(-1) + .6 \times .6 \times (1.5 + 6 - 2) = 16.38$$

$$t=9+$$

$$C_v = 15 + .6(-1) + 0 = 14.4$$

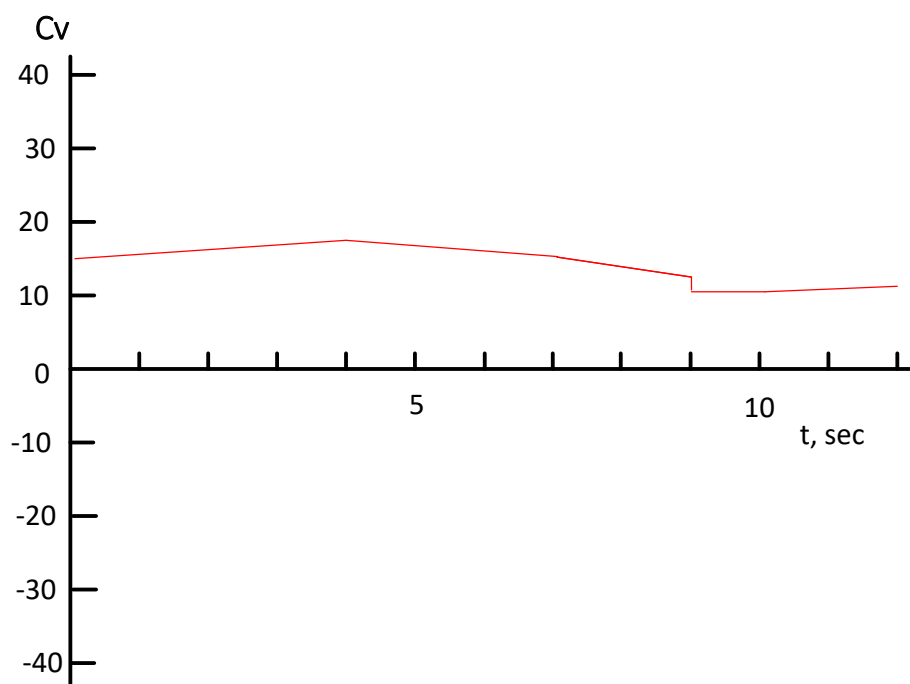
$$t=10$$

$$C_v = 15 + .6(-1) + .6 \times .6 \times (-1) = 14.04$$

$$t=12$$

$$C_v = 15 + .6 \times .6 \times (-2) = 14.28$$

Graph



Whether a PLC or a DCS is better at implementing Process Control:

“22 March 2011

It may surprise you to know that PLC, HMI and SCADA implementations today are consistently proving more expensive than DCS for the same process or batch application. CEE finds out more...

Traditionally, DCSs were large, expensive and very complex systems that were considered as a control solution for the continuous or batch process industries. In large systems this is, in principle, still true today, with engineers usually opting for PLCs and HMIs or SCADA for smaller applications, in order to keep costs down.

So what has changed? Integrating independent PLCs, the required operator interface and supervisory functionality, takes a lot of time and effort. The focus is on making the disparate technology work together, rather than improving operations, reducing costs, or improving the quality or profitability of a plant.

Yet a PLC/ SCADA system may have all or part of the following list of independent and manually coordinated databases.

- * Each controller and its associated I/O
- * Alarm management
- * Batch/recipe and PLI
- * Redundancy at all levels
- * Historian
- * Asset optimization
- * Fieldbus device management

See the following video from RealPars for a comparison of the PLC vs DCS system. The arguments may not fit exactly with your application but the video is worth the effort.

What is the Difference Between PLC and DCS?

<https://www.youtube.com/watch?v=iF99iKIDpxA>

Making changes

Every time a change is made in one database, the others usually need to be updated to reflect that change. For example, when an I/O point and some control logic are added there may be a need to change or add a SCADA element, the historian and the alarm database. This will require the plant engineer to make these changes in each of these databases, not just one – and get it right.”

While you may not be a proponent of either the DCS or PLC for Process control, the above is something worth thinking about. The arguments are not trivial. If one programs a process application with PLCs, then the objections mentioned in the above article must be dealt with and the negative effects of using the PLC minimized.

Other RealPars videos on PID include:

[How to Tune a PID Controller](#)

<https://www.youtube.com/watch?app=desktop&v=IB1lr4oCP5k&list=PLIn3BHg93SQ9SEN8jXvhyxcAeFcmkjTNS&index=41>

[PID Controller Explained](#)

<https://www.youtube.com/watch?app=desktop&v=fv6dLTEvl74&list=PLIn3BHg93SQ9SEN8jXvhyxcAeFcmkjTNS&index=85>

[PID Settings | Proportional-Only Control](#)

<https://www.youtube.com/watch?app=desktop&v=E780BPOjKwM&list=PLIn3BHg93SQ9SEN8jXvhycxvhyxcAeFcmkjTNS&index=94>

Summary

This chapter has the purpose of taking the programmer from the state of asking “What is a PID loop” to being able to program a PID loop, implement a faceplate, consider how more than one PID block can be combined to control complex processes and encourage the programming of at least one PID project complete with tuning and HMI panel.

A student should be able to accomplish each of the steps listed above from the examples in the chapter and implement a PID process in the laboratory.

Students should also be able to read a P&ID and interpret the parts of the P&ID that can be implemented in a controller including the PID algorithm.

HMI considerations also should be heeded and alarms that control the mode of the PID block were discussed.

1. **Lab 19.1 PID**

Use the Extruder/Mixing System making Dog Food of Fig. 19-60 to design a PID controller for the Fat Valve. A potentiometer may be present and (if present) may be used to represent the motor speed. Input the potentiometer into a second analog input. To simulate the change of speed of the motor, change the analog value from the pot. Demonstrate the running face-plate with auto-manual and local-remote to the instructor. When the PID algorithm crosses between auto and manual or between auto-remote and auto-local provide a bump-less transfer (optional). You may program the A-B and Siemens processors in either Ladder or FBD. Both processors must be demonstrated and their PID control discussed in a lab report. The Siemens process is the ball-in-tube and the A-B process is the water valve.

2. **Lab 19.2 Advanced PID**

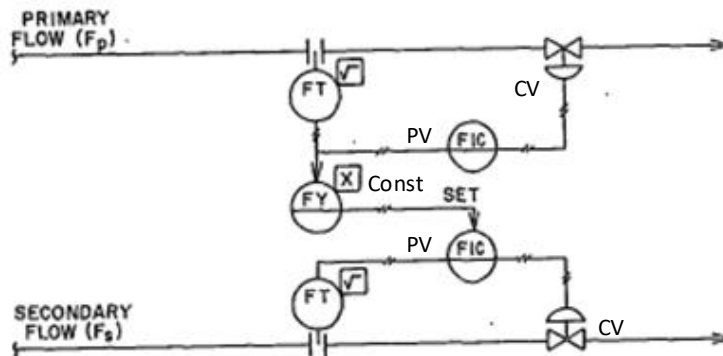
Add logic to PID Lab 17.1 to program to ramp from the old setpoint to a new setpoint using a ramping block. Program the ramping only for the remote mode (although the ramping function typically done in all automatic modes since it is needed to protect the process). When a new value is entered in the remote Sp entry location, the PID's Sp is not to immediately change to the new Sp, but rather it is to be ramped up or down from the present value (found in the Pv). Save the Pv when the new Sp is detected and determine whether the Pv is below or above the new Sp. Set a seal coil or latch coil to remember which way the ramp is going (either up or down). Also, start a timer to time out each 5 to 10 seconds. When the timer times out, add a small amount (delta) to the new Sp and then compare it to the Remote Sp. If the ramped Sp went past the Remote Sp, stop the ramp and put the Remote Sp in the PID's Sp. Then end the ramp program and wait for another Sp change. Also, stop the ramp if the PID loop is taken to manual from auto. Add a fault circuit that detects if the flow is dangerously low for the value of the output. If this kind of fault occurs, the PID algorithm might begin to wind up (read about anti-reset-windup in the PID section of the A-B book). If the low-flow fault occurs, blink an alarm light on the PanelView and turn the PID block to manual. Set the bit in the alarm banner.

Exercises

1. When a PID controller is in remote, is the mode in auto or manual?
2. T/F Windup of the controller is possible in manual mode?
3. T/F The controller performs exactly the same whether the controller is set for $E = PV - SP$ or $E = SP - PV$.
4. What is the purpose of the small triangles on the left and right side of the bar graphs of a faceplate?
5. List the function of the following ISA symbols:

LT
LIC
FIC
dTC
6. The process engineer says that you are to move the PID controller from auto to manual if any of the analog signals (4-20 mA) are invalid in the low range. Show with an example how to accomplish this in ladder logic. Assume the analog inputs are in slot 5. Label all rungs explaining your logic.
7. A temperature profile of two different TT's is to be added together in varying percentages to provide the PV for a PID controller. Show with an example how to accomplish this in ladder logic. Provide a mechanism so that if the percentage is not 100% that the PID block will only run in manual mode. Label all rungs explaining your logic. You should show the PID block but do not provide logic for the SP or CV. Assume the analog inputs are wired to a 4-20 mA analog card in slot 3.
8. A speed sensor has a high and low alarm attached to it. The signal from the sensor is transmitted to a computer. Draw a P&ID of the speed signal transmitter, high alarm and low alarm. Assume the signals are attached to a computer and are field mounted.
9. A differential pressure transducer transmits a signal that is used for flow. However, flow is proportional to the square root of the differential pressure. An analog input card is to be used with range 1-5V input for the PV and an analog output card is to be used for the CV, range 1-5V. The SP is to be input from an HMI. Draw the P&ID showing the mathematical calculation of the square root. Any symbol type is appropriate. Then write a program to control the flow using the analog cards listed. Assume the input card is in slot 4 and the output card is in slot 6.
10. In some temperature control, the output device is a switch that turns on or off a resistor to produce heat. If the output of a PID block is fed to a discrete output that can only turn the resistors on or off, write a program to turn the discrete output on or off a proportion of 10 seconds based on value of the CV. Assume the output CV can range only from 0 to 100 and its value is found in a storage location.
11. Build a lag controller capable of a 5 second lag with value changes each .5 second. Build a lag controller capable of an x second lag with value changes each y second.

12. Using either the PID blocks from A-B or Siemens, provide a program that will work in auto mode for the following P&ID. Use variables as inputs, outputs and internal variables as necessary. Describe these variables in a table. PV's and CV's are marked. Unmarked SP's are setpoints from the HMI screen.



13. Using either the PID blocks from A-B or Siemens, provide a program that will work in auto mode for the following P&ID. Use variables as inputs, outputs and internal variables as necessary. Describe these variables in a table.

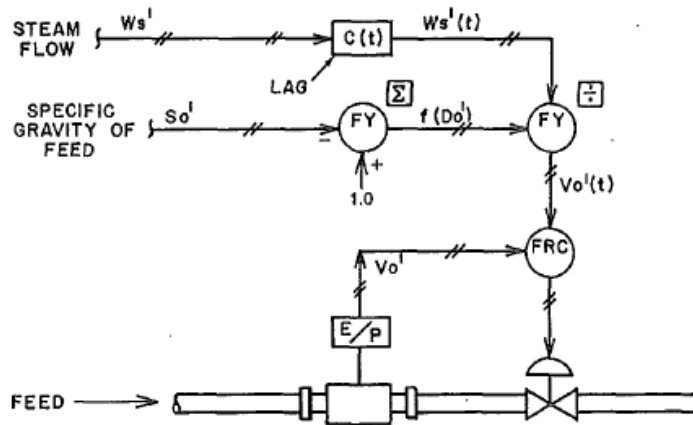


FIG. 8.16t
Feedforward control loop based on Equation 8.16(29) and provided with lag for dynamic compensation.

or if feed flow is the manipulated variable:

$$V_o' = \frac{W_s'}{1.11(1 - 0.437 S_o')} \quad 8.16(29)$$

14. Using either the PID blocks from A-B or Siemens, provide a program that will work in auto mode for the following P&ID. Use variables as inputs, outputs and internal variables as necessary. Describe these variables in a table. PV's and CV's are marked in the diagram. Unmarked SP's are setpoints from the HMI screen. Ignore sqrt operation. Assume that FT's give a flow rate and the sqrt function is performed at the transmitter, not the controller.

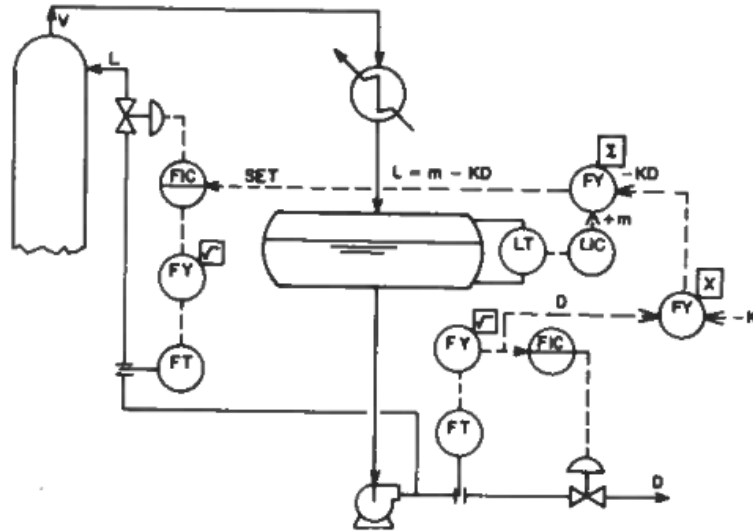


FIG. 8.13d
Reflux rate control system for overcoming accumulator lag.

15. Provide PID blocks for all the PID controllers in the following P&ID. Identify SP, PV and CV using labels from the P&ID or descriptors you made up.

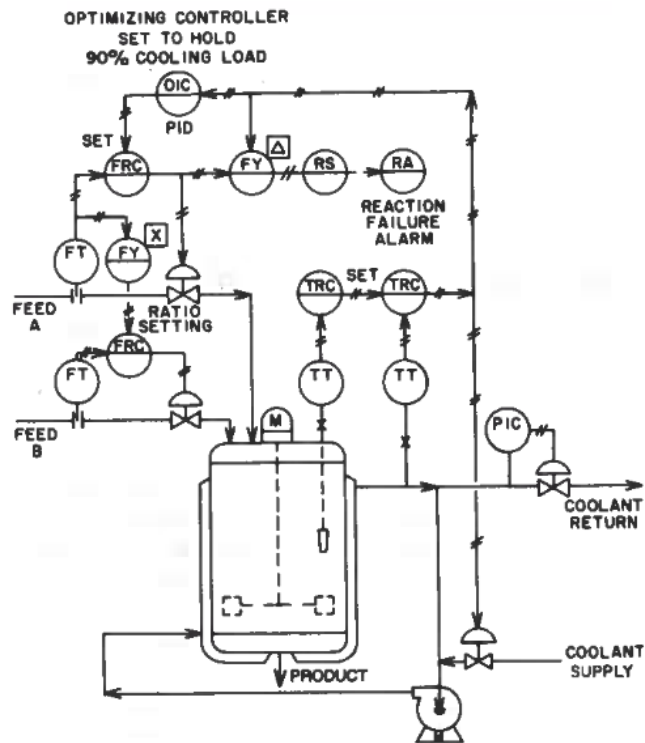


FIG. 8.24w
The reaction rate in continuous reactor is matched to the capacity of the cooling system.

16. Write logic to provide a 30-second lag given that the variable is to be updated each .1 second. Use A-B ladder format to demonstrate your answer.
17. At the end of Ch. 19 is an article:

A Discussion Comparing DCS and PLC/SCADA for Process Control

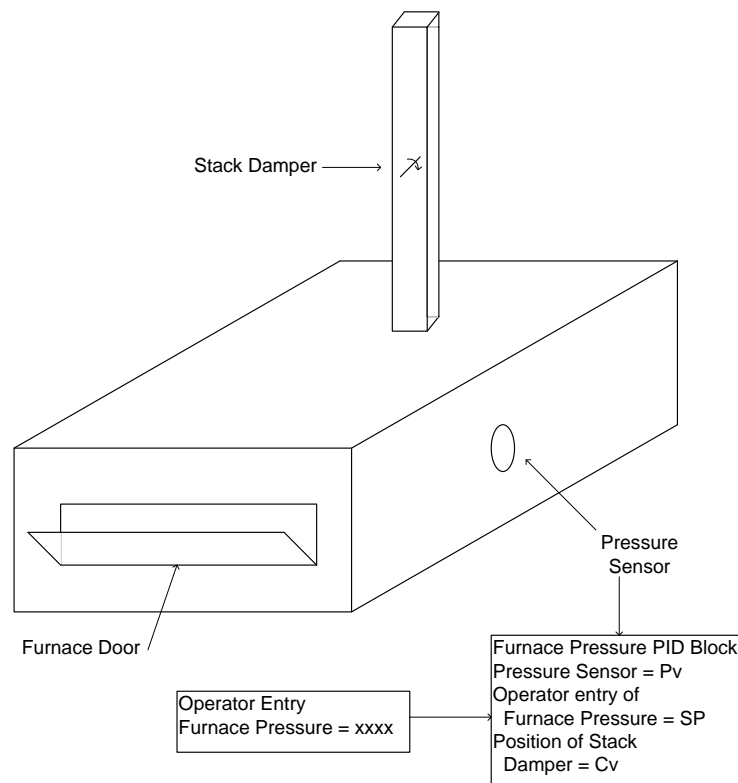
DCS and PLC/SCADA – a comparison in use

The author stipulates:

It may surprise you to know that PLC, HMI and SCADA implementations today are consistently proving more expensive than DCS for the same process or batch application. CEE finds out more...

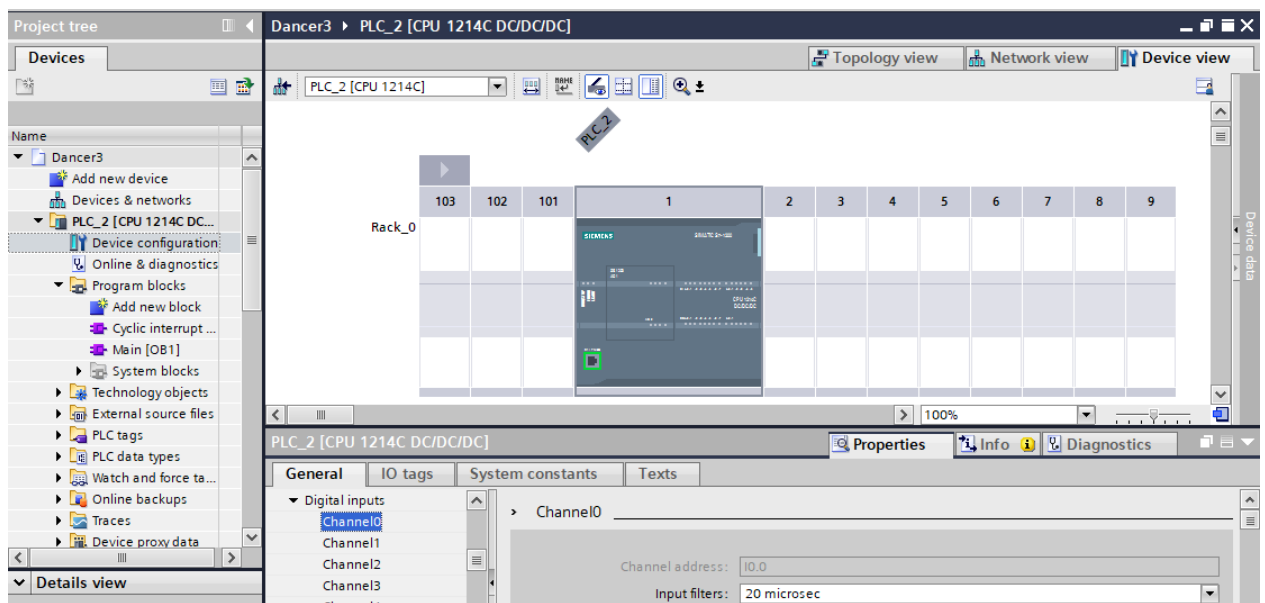
What does the author claim for the basis of his arguments and what would you do as a PLC programmer to counter these claims? Be specific:

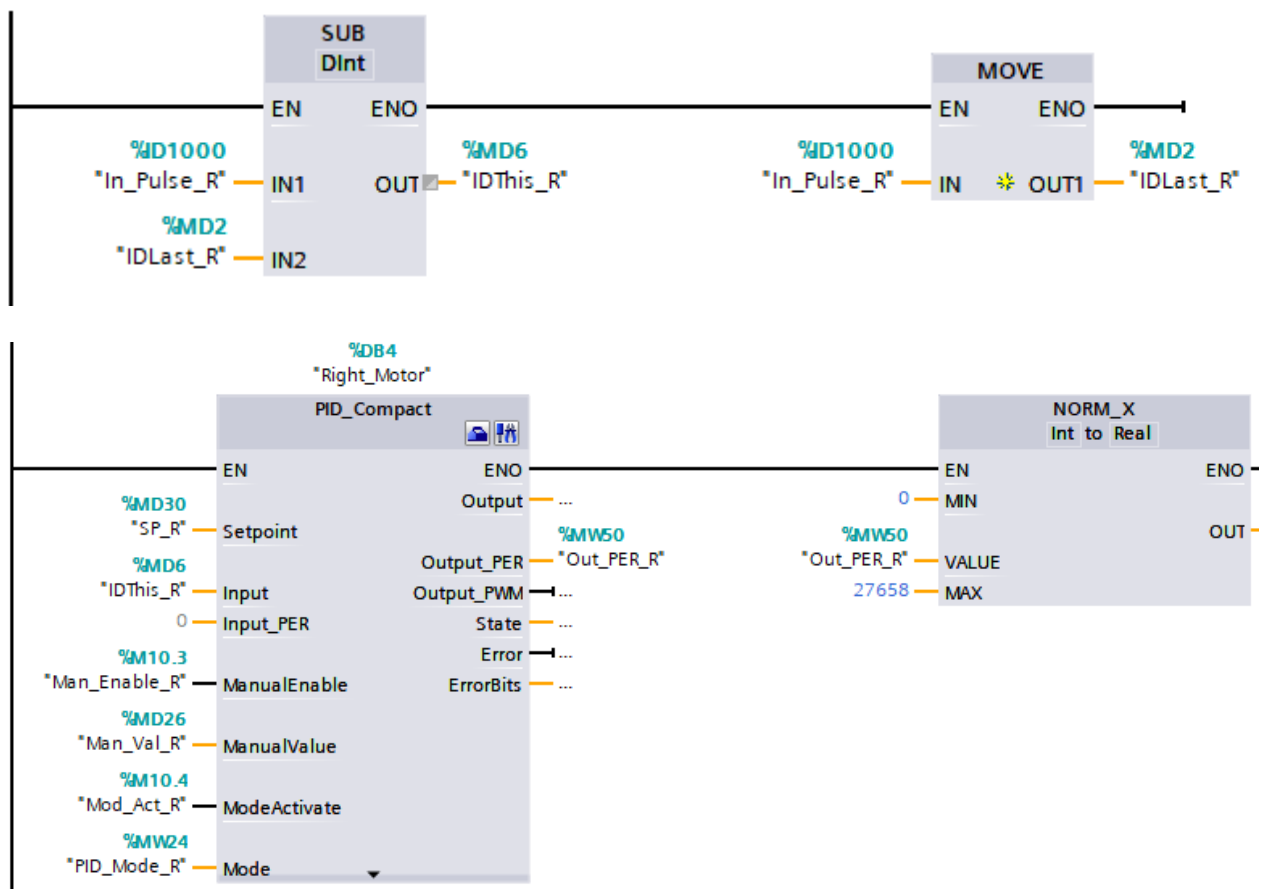
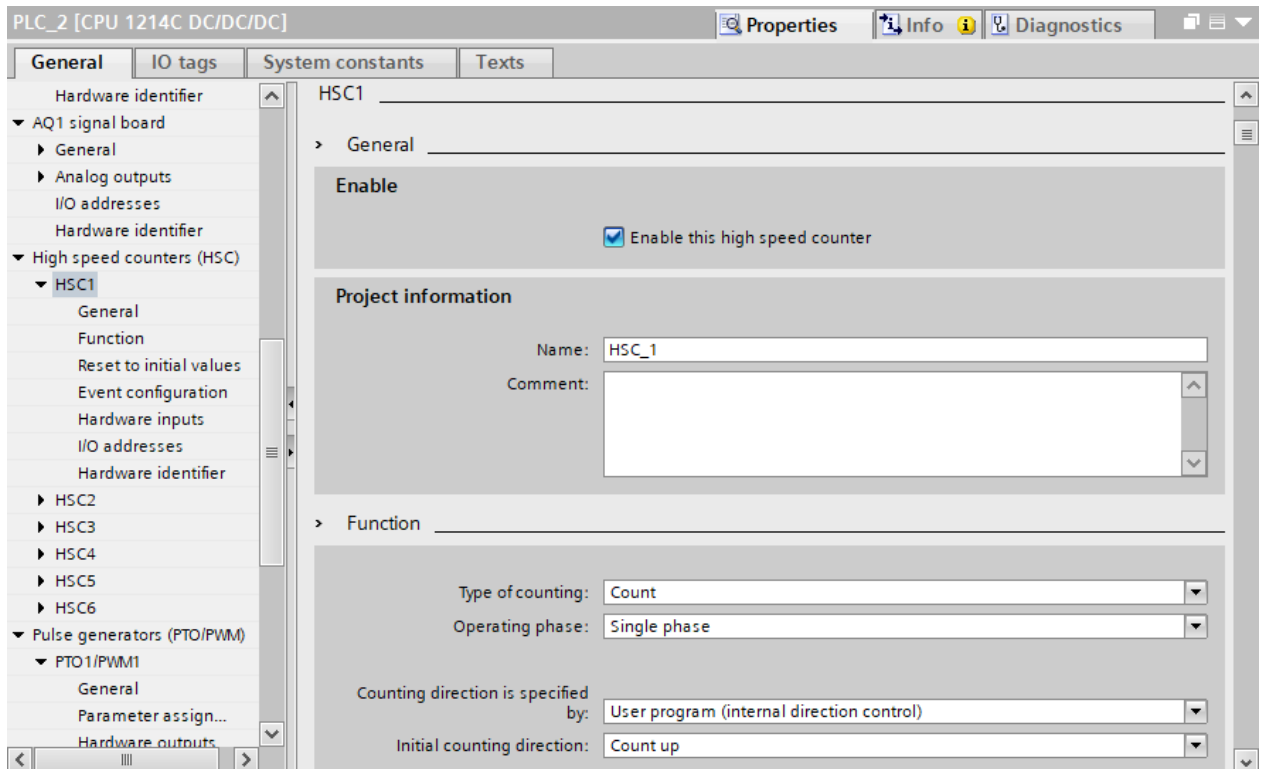
18. Give an example of multiple inputs being used instead of just one value for the PV (Process Variable) of a PID Loop. Write a program using either A-B or Siemens to demonstrate your answer.
19. An example was given in class describing how to control the pressure in a steel furnace even when the door was opened. Describe of how you would accomplish this. Be specific:

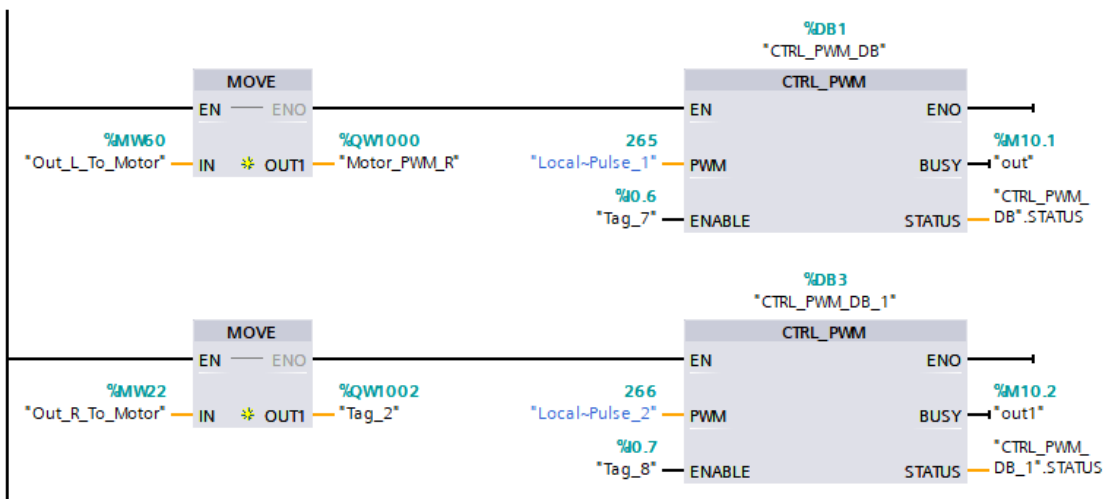
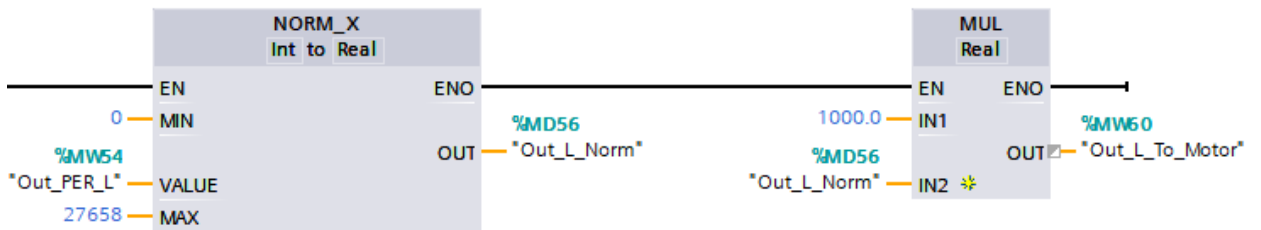
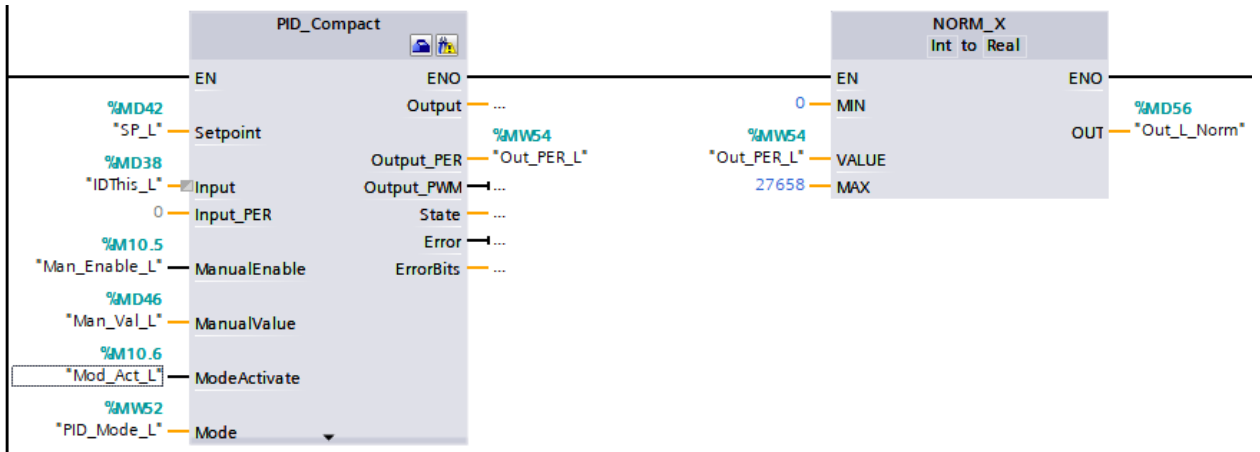
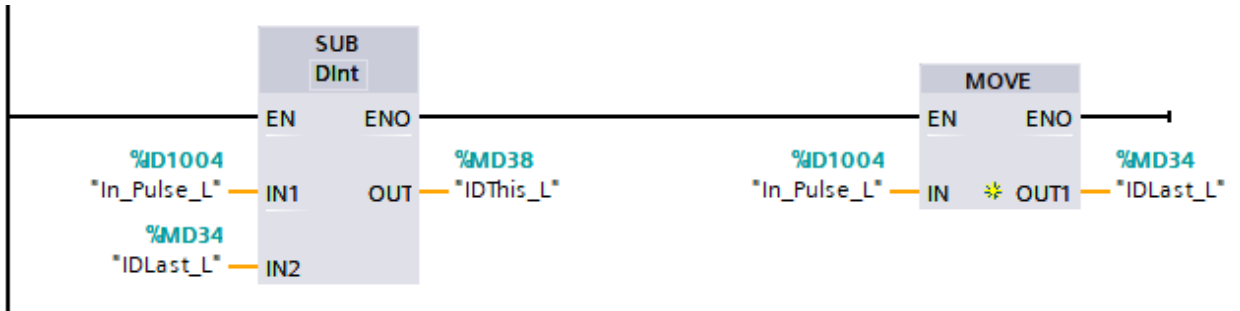
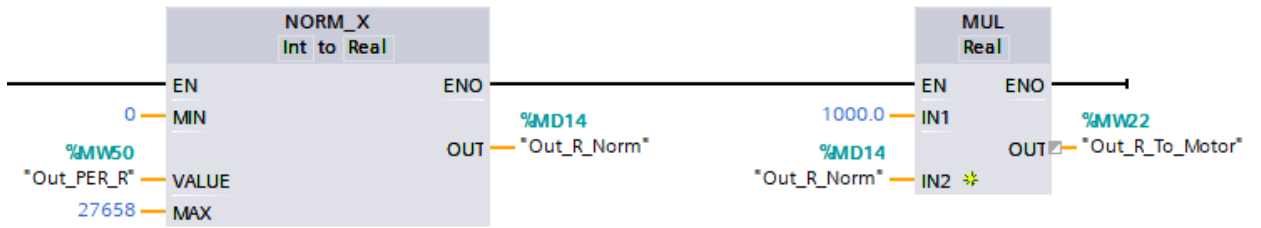


20. If an input range is listed as 0 mA to 21 mA range is from 0 to 32640 and we want a 4-20 mA. What is the numeric range of a 4-20 mA signal?
21. A good value for P for a servo: _____

22. A good cyclic time to update the PID Control for a servo: _____
23. A good value for P for a water loop: _____
24. A good cyclic time to update the PID control for a water loop: _____
25. A good value for P for a temperature loop: _____
26. A good cyclic time for update of the PID control for a temperature loop: _____
27. Name a PID control loop that does fine with no derivative component: _____
28. Name a PID control loop that is unstable if the derivative is left at zero: _____
29. The following program is a starter program to control the wind-up of the tape. To start understanding it, provide comments for each statement in the program listing. All statements are found in the cyclic interrupt program OB30 which is run each 100 ms. The individual motor programs can be used to control the speed portion of the gearmotor project discussed in the chapter.

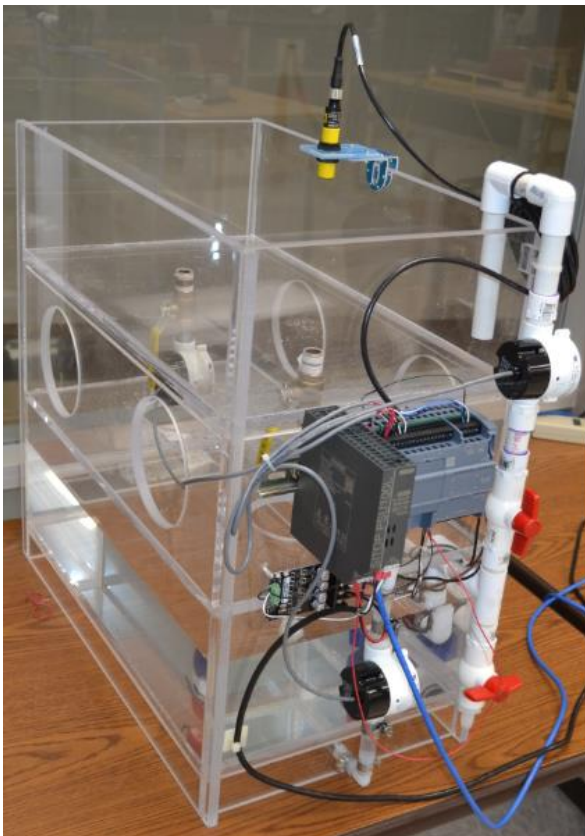






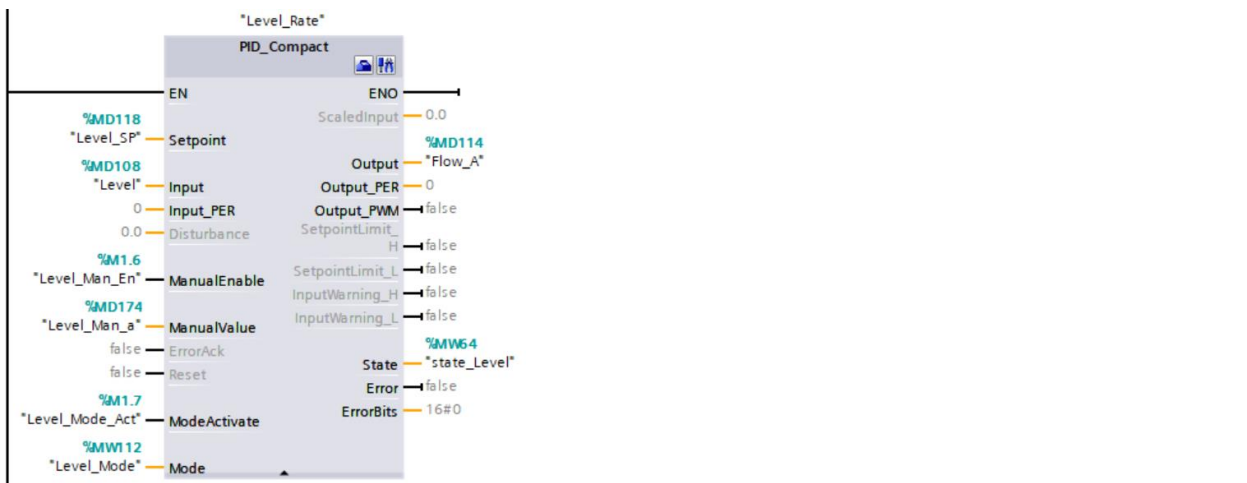
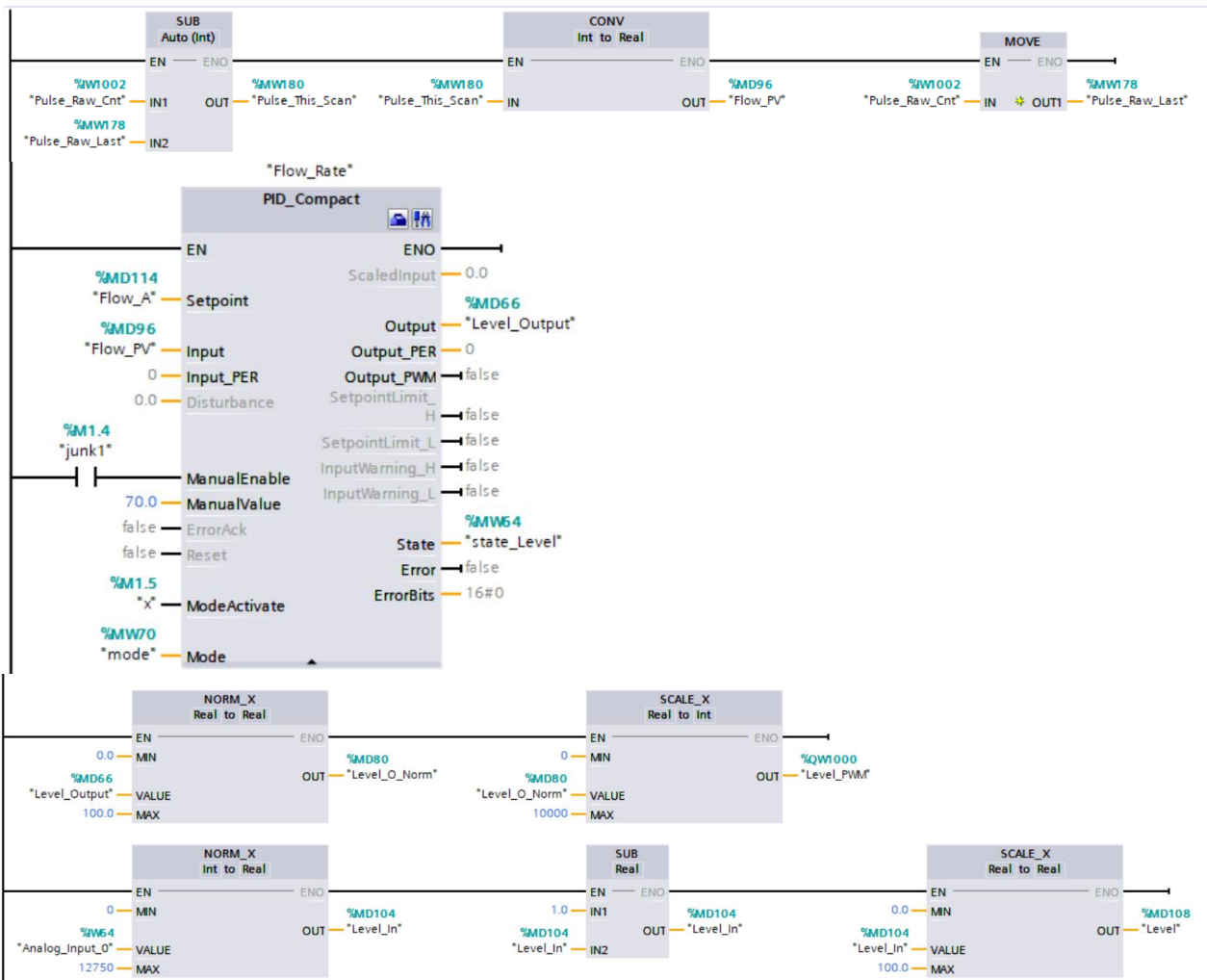


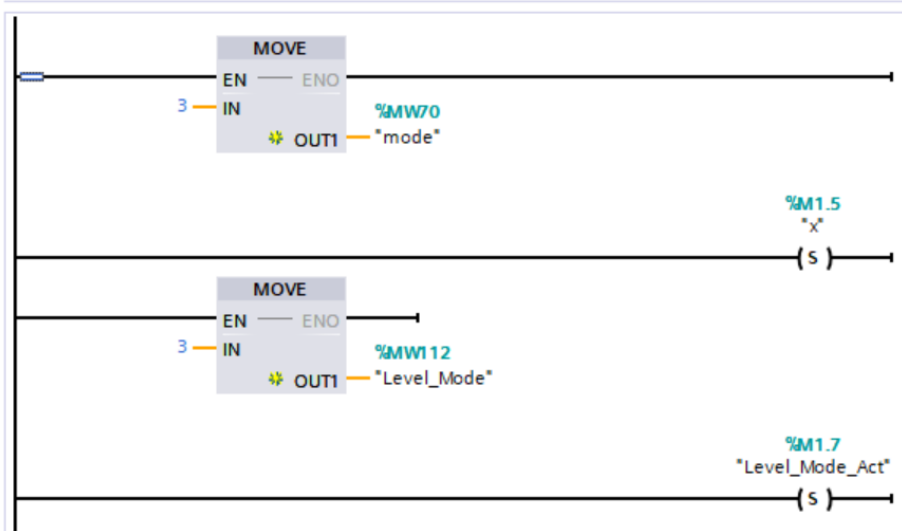
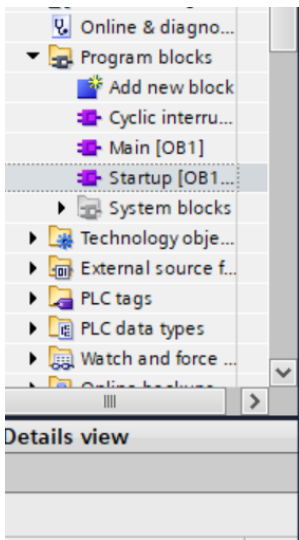
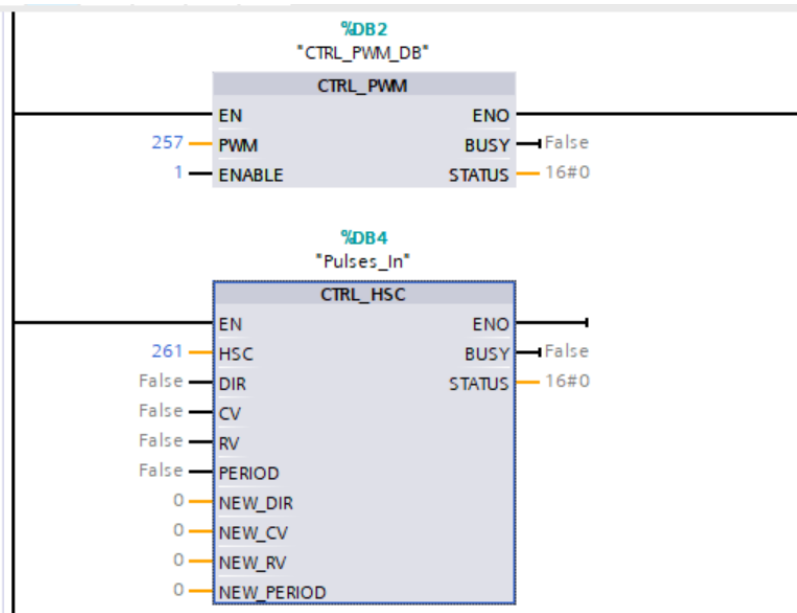
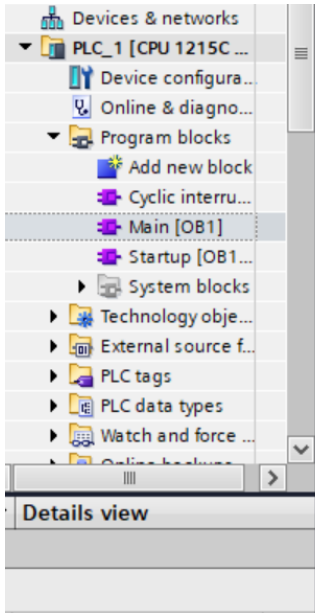
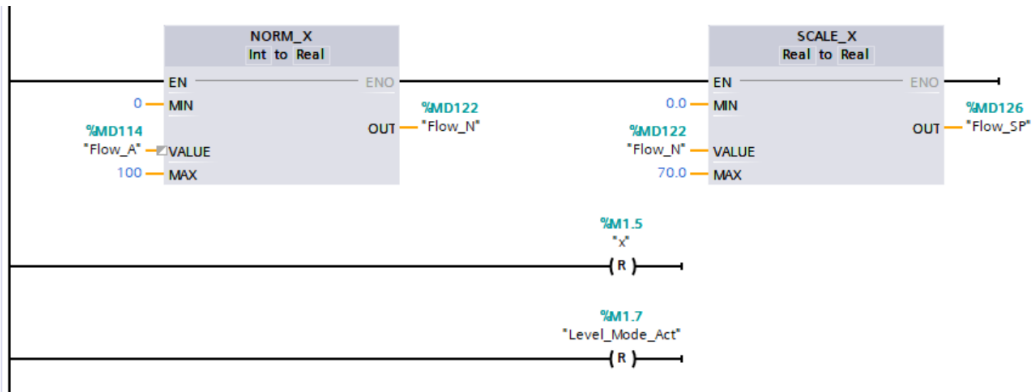
30. The following program is a starter program to control the control of water level in the top tank. To start understanding it, provide comments for each statement in the program listing. All statements are found in the cyclic interrupt program OB30 and start-up program OB100. OB30 is run each 1000 ms (1 sec).



Program blocks

- Add new block
- Cyclic interrupt [OB30]
- Main [OB1]
- Startup [OB100]





This work is licensed under a Creative Commons Attribution 4.0 International License.