PID Loop Operation

In This Chapter...

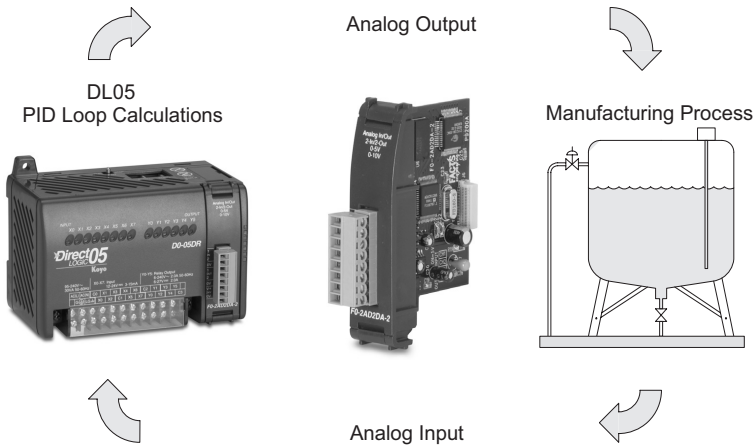
DL05 PID Control

DL05 PID Control Features

Along with control functions discussed in this manual, the DL05 PLC features PID process control capability. The DL05 PID process control loops offer the same features offered in much larger PLCs. The primary features are:

- Up to 4 PID loops, individual programmable sample rates
- Manual, Automatic and Cascade loop operation modes
- Two types of bumpless transfer available
- Full-featured alarms
- Ramp/soak generator with up to 16 segments
- Auto Tuning

The DL05 CPU has process control loop capability in addition to ladder program execution. Up to four loops can be selected and configured. All sensor and actuator wiring connects directly to DL05 analog I/O modules. All process variables, gain values, alarm levels, etc., associated with each loop reside in a Loop Variable Table in the CPU. The DL05 CPU reads process variable (PV) inputs during each scan. Then it makes PID loop calculations during a dedicated time slice on each PLC scan, updating the control output value. The control loops use a Proportional-Integral-Derivative (PID) algorithm to generate the control output. This chapter describes how the loops operate, and how to configure and tune the loops.



*Direct*SOFT programming software, release 5, or later, is used for configuring analog control loops in the DL05. *Direct*SOFT uses Intelligent boxes (IBoxes) to help you set up the individual loops. After completing the setup, you can use *Direct*SOFT's PID Trend View to tune each loop. The configuration and tuning selections you make are stored in the DL05's V-memory (RAM). The loop parameters also may be saved to disk for recall later.

PID Loop Feature	Specifications
Number of loops	Selectable, 4 maximum
CPU V-memory needed	32 words (V locations) per loop selected, 64 words if using ramp/soak
PID algorithm	Position or Velocity form of the PID equation
Control Output polarity	Selectable direct-acting or reverse-acting
Error term curves	Selectable as linear, square root of error, and error squared
Loop update rate (time between PID calculation)	0.05 to 99.99 seconds, user programmable
Minimum loop update rate	0.05 seconds for 1 to 4 loops
Loop modes	Automatic, Manual (operator control), or Cascade control
Ramp/Soak Generator	Up to 8 ramp/soak steps (16 segments) per loop with indication of ramp/soak step number
PV curves	Select standard linear, or square-root extract (for flow meter input)
Set Point Limits	Specify minimum and maximum setpoint values
Process Variable Limits	Specify minimum and maximum Process Variable values
Proportional Gain	Specify gains of 0.01 to 99.99
Integrator (Reset)	Specify reset time of 0.1 to 999.8 in units of seconds or minutes
Derivative (Rate)	Specify the derivative time from 0.01 to 99.99 seconds
Rate Limits	Specify derivative gain limiting from 1 to 20
Bumpless Transfer I	Automatically initialized bias and setpoint when control switches from manual to automatic
Bumpless Transfer II	Automatically set the bias equal to the control output when control switches from manual to automatic
Step Bias	Provides proportional bias adjustment for large setpoint changes
Freeze Bias (Anti-windup)	For position form of PID, this inhibits integrator action when the control output reaches 0% or 100 % (speeds up loop recovery when output recovers from saturation)
Error Deadband	Specify a tolerance (plus and minus) for the error term (SP–PV), so that no change in control output value is made

Alarm Feature	Specifications
PV Alarm Hysteresis	Specify 1 to 200 (word/binary) does not affect all alarms, such as, PV Rate-of-Change Alarm
PV Alarm Points	Select PV alarm settings for Low–low, Low, High, and High-high conditions
PV Deviation	Specify alarms for two ranges of PV deviation from the setpoint value
Rate of Change	Detect when PV exceeds a rate of change limit you specify

Introduction to PID Control

What is PID Control?

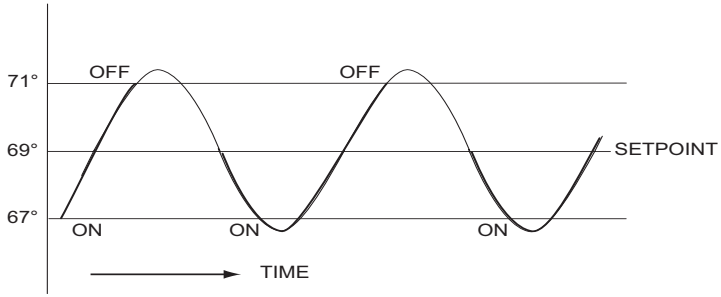
In this discussion, we will explain why PID control is used in process control instead of trying to provide control by simply using an analog input and a discrete output.

There are many types of analog controllers available, and the proper selection will depend upon the particular application. There are two types of analog controllers that are used throughout industry:

- 1. The ON-OFF controller, sometimes referred to as an open loop controller.
- 2. The PID controller, sometimes called a closed loop controller.

Regardless of type, analog controllers require input signals from electronic sensors such as pressure, differential pressure, level, flow meter or thermocouples. As an example, one of the most common analog control applications is located in your house for controlling either heat or air conditioning, the thermostat.

You wish for your house to be at a comfortable temperature so you set a thermostat to a desired temperature (setpoint). You then select the "comfort" mode, either heat or A/C. A temperature sensing device, normally a thermistor, is located within the thermostat. If the thermostat is set for heat and the setpoint is set for 69°, the furnace will be turned on to provide heat at, normally, 2° below the setpoint. In this case, it would turn on at 67°. When the temperature reaches 71°, 2° above setpoint, the furnace will turn off. In the opposite example, if the thermostat is set for A/C (cooling), the thermostat will turn the A/C unit on/off opposite the heat setting. For instance, if the thermostat is set to cool at 76°, the A/C unit will turn on when the sensed temperature reaches 2° above the setpoint or 78°, and turn off when the temperature reaches 74°. This would be considered to be an ON-OFF controller. The waveform below shows the action of the heating cycle. Note that there is a slight overshoot at the turn-off point, also a slight undershoot at the turn-on point.



The ON-OFF controller is used in some industrial control applications, but is not practical in the majority of industrial control processes.

The most common process controller that is used in industry is the PID controller.

The PID controller controls a continuous feedback loop that keeps the process output (control variable) flowing normally by taking corrective action whenever there is a deviation from the desired value (setpoint) of the process variable (PV) such as, rate of flow, temperature, voltage, etc. An "error" occurs when an operator manually changes the setpoint or when an event (valve opened, closed, etc.) or a disturbance (cold water, wind, etc.) changes the load, thus causing a change in the process variable.

The PID controller receives signals from sensors and computes corrective action to the actuator from a computation based on the error (Proportional), the sum of all previous errors (Integral) and the rate of change of the error (Derivative).

We can look at the PID controller in more simple terms. Take the cruise control on an automobile as an example. Let's say that we are cruising on an interstate highway in a car equipped with cruise control. The driver decides to engage the cruise control by turning it ON, then he manually brings the car to the desired cruising speed, say 70 miles per hour. Once the cruise speed is reached, the SET button is pushed fixing the speed at 70 mph, the setpoint. Now, the car is cruising at a steady 70 mph until it comes to a hill to go up. As the car goes up the hill, it tends to slow down. The speed sensor senses this and causes the throttle to increase the fuel to the engine. The vehicle speeds up to maintain 70 mph without jerking the car and it reaches the top at the set speed. When the car levels out after reaching the top of the hill it will speed up. The speed sensor senses this and signals the throttle to provide less fuel to the engine, thus, the engine slows down allowing the car to maintain the 70mph speed. How does this application apply to PID control? Lets look at the function of P, I and D terms:

- **Proportional** - is commonly referred to as Proportional Gain. The proportional term is the corrective action which is proportional to the error, that is, the change of the manipulated variable is equal to the proportional gain multiplied by the error (the activating signal). In mathematical terms:

  Proportional action = proportional gain X error
  Error = Setpoint (SP) - Process Variable (PV)

- Applying this to the cruise control, the speed was set at 70 mph which is the Setpoint. The speed sensor senses the actual speed of the car and sends this signal to the cruise controller as the Process Variable (PV). When the car is on a level highway, the speed is maintained at 70 mph, thus, no error since the error would be SP - PV = 0. When the car goes up the hill, the speed sensor detected a slow down of the car, SP-PV = error. The proportional gain would cause the output of the speed controller to bring the car back to the setpoint of 70 mph. This would be the Controlled Output.

- **Integral** - this term is often referred to as Reset action. It provides additional compensation to the control output, which causes a change in proportion to the value of the error over a period of time. In other words, the reset term is the integral sum of the error values over a period of time.

- **Derivative** - this term is referred to as rate. The Rate action adds compensation to the control output, which causes a change in proportion to the rate of change of error. Its job is to anticipate the probable growth of the error and generate a contribution to the output in advance.

Introducing DL05 PID Control

The DL05 is capable of controlling a process variable such as those already mentioned. As previously mentioned, the control of a variable, such as temperature, at a given level (setpoint) as long as there are no disturbances (cold water) in the process.

The DL05 PLC has the ability to directly accept signals from electronic sensors, such as thermocouples, pressure, VFDs, etc. These signals may be used in mathematically derived control systems.

In addition, the DL05 has built-in PID control algorithms that can be implemented. The basic function of PID closed loop process control is to maintain certain process characteristics at desired setpoints. As a rule, the process deviates from the desired setpoint reference as a result of load material changes and interaction with other processes. During this control, the actual condition of the process characteristics (liquid level, temperature, motor control, etc.) is measured as a *process variable* (PV) and compared with the target setpoint (SP). When deviations occur, an error is generated by the difference between the process variable (actual value) and the setpoint (desired value). Once an error is detected, the function of the control loop is to modify the control output in order to force the error to zero.

The DL05 PID control provides feedback loops using the PID algorithm. The control output is computed from the measured process variable as follows:

Let:

- K_c = proportional gain
- T_i = Reset or integral time
- T_d = Derivative time or rate
- SP = Setpoint
- PV(t) = Process Variable at time "t"
- e(t) = SP-PV(t) = PV deviation from setpoint at time "t" or PV error.

Then:

- M(t) = Control output at time "t"

$$M(t) = Kc \left[e(t) + 1/T_i \int_0^t e(x)\,dx + T_d\, d/dt\, e(t) \right] + M_o$$

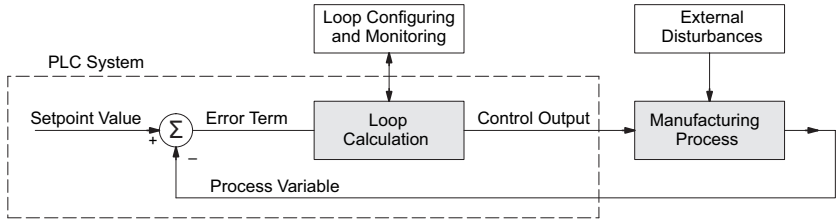
The analog input module receives the process variable in analog form along with an operator entered setpoint; the CPU computes the error. The error is used in the algorithm computation to provide corrective action at the control output. The function of the control action is based on an output control, which is proportional to the instantaneous error value. The *integral control action* (reset action) provides additional compensation to the control output, which causes a change in proportion to the value of the change of error over a period of time. The *derivative control action* (rate change) adds compensation to the control output, which causes a change in proportion to the rate of change of error. These three modes are used to provide the desired control action in Proportional (P), Proportional-Integral (PI), or Proportional-Integral-Derivative (PID) control fashion.

Standard DL05/06 analog input modules are used to interface to field transmitters to obtain the PV. These transmitters normally provide a 4-20 mA current or an analog voltage of various ranges for the control loop.

For temperature control, thermocouple or RTD can be connected directly to the appropriate module. The PID control algorithm, residing in the CPU memory, receives information from the user program, primarily control parameters and setpoints. Once the CPU makes the PID calculation, the result may be used to directly control an actuator connected to a 4-20mA current output module to control a valve.

With *Direct*SOFT programming software, additional ladder logic programming, both time proportioning (e.g. heaters for temperature control) and position actuator (e.g. reversible motor on a valve) type of control schemes can be easily implemented. This chapter will explain how to set up the PID control loop, how to implement the software and how to tune the loop.

The following block diagram shows the key parts of a PID control loop. The path from the PLC to the manufacturing process and back to the PLC is the closed loop control.



Process Control Definitions

Manufacturing Process – the set of actions that adds value to raw materials. The process can involve physical changes and/or chemical changes to the material. The changes render the material more useful for a particular purpose, ultimately used in a final product.

Process Variable – The controlled variable part of the process that you wish to control. It may be temperature, pressure, level, flow, composition, density, the ratio of two streams, etc. Also known as the actual value.

Setpoint – This is the target for the process variable. When all conditions of the process are correct, the process variable will equal the setpoint.

Control Output – The result of the loop calculation, which becomes a command for the process (such as the heater level in an oven). This is sometimes referred to as control variable.

Error Term – The algebraic difference between the process variable and the setpoint. This is the control loop error, and is equal to zero when the process variable is equal to the setpoint (desired) value. A well-behaved control loop is able to maintain a small error term magnitude.

Manipulated Variable – This is what is used to effect the controlled variable. For example, the fuel used in a furnace might be manipulated in order to control the temperature.

Disturbance – Something in the system that changes such that corrective action is required. For instance, when controlling a flow and the upstream pressure drops, the control valve must open wider in order to keep flow constant. The drop in upstream pressure is the disturbance.

Final Control Element – This is the physical device used to control the manipulated variable. Valves are probably the most widely used final control element.

Lag Time – The time it takes for the process to respond to a change in manipulated variable. This is also known as the capacitance of the system. When you're in the shower and you turn up the hot water a little, the time it takes before the water gets hot is the lag time.

Dead Time – The time it takes for a change in the process to be recognized. Composition analyzers and quality control are usually sources of significant dead time.

Loop Configuring – Operator-initiated selections which set up and optimize the performance of a control loop. The loop calculation function uses the configuration parameters in real time to adjust gains, offsets, etc.

Loop Monitoring – The function which allows an operator to observe the status and performance of a control loop. This is used in conjunction with the loop configuring to optimize the performance of a loop (minimize the error term).

PID Loop Operation

The Proportional–Integral–Derivative (PID) algorithm is widely used in process control. The PID method of control adapts well to electronic solutions, whether implemented in analog or digital (CPU) components. The DL05 CPU implements the PID equations digitally by solving the basic equations in software. I/O modules serve only to convert electronic signals into digital form (or vice versa).

The DL05 uses two types of PID controls: "position" and "velocity". These terms usually refer to motion control situations, but here we use them in a different sense:

- PID Position Algorithm – The control output is calculated so it responds to the displacement (position) of the PV from the SP (error term).
- PID Velocity Algorithm – The control output is calculated to represent the rate of change (velocity) for the PV to become equal to the SP.

Position Form of the PID Equation

Referring to the control output equation on page 8-6, the DL05 CPU approximates the output M(t) using a discrete position form of the PID equation.

- **Let:**

  Ts = Sample rate
  Kc = Proportional gain
  Ki = Kc * (Ts/Ti) = Coefficient of integral term
  Kr = Kc * (Td/Ts) = Coefficient of derivative term
  Ti = Reset or integral time
  Td = Derivative time or rate
  SP = Setpoint
  PV_n = Process variable at n^{th} sample
  en = SP – PV_n = Error at n^{th} sample
  M_o = Value to which the controller output has been initialized

- **Then:**

  M_n = Control output at n^{th} sample

$$M_n = Kc * e_n + Ki \sum_{i=1}^{n} e_i + Kr\,(e_n - e_{n-1}) + M_o$$

This form of the PID equation is referred to as the position form since the actual actuator position is computed. The velocity form of the PID equation computes the change in actuator position. The CPU modifies the standard equation slightly to use the derivative of the process variable instead of the error as follows:

$$M_n = Kc * e_n + Ki \sum_{i=1}^{n} e_i + Kr\,(PV_n - PV_{n-1}) + M_o$$

These two forms are equivalent unless the setpoint is changed. In the original equation, a large step change in the setpoint will cause a correspondingly large change in the error resulting in a bump to the process due to derivative action. This bump is not present in the second form of the equation.

Step Bias Proportional to Step Change in SP

This feature reduces oscillation caused by a step change in setpoint when the adjusting bias feature is used.

$Mx = Mx * SP_n / SP_{n-1}$ "if the loop is direct acting"

$Mx = Mx * SP_{n-1} / SP_n$ "if the loop is reverse acting"

$Mx_n = 0$ "if $Mx < 0$"

$Mx_n = Mx$ "if $0 \leq Mx \leq 1$"

$Mx_n = 1$ "if $M > 1$"

Eliminating Proportional, Integral or Derivative Action

It is not always necessary to run a full three mode PID control loop. Most loops require only the PI terms or just the P term. Parts of the PID equation may be eliminated by choosing appropriate values for the gain (Kc), reset (Ti) and rate (Td) yielding a P, PI, PD, I and even an ID and a D loop.

Eliminating Integral Action The effect of integral action on the output may be eliminated by setting Ti = 9999 or 0000. When this is done, the user may then manually control the bias term (Mx) to eliminate any steady-state offset.

Eliminating Derivative Action The effect of derivative action on the output may be eliminated by setting Td = 0 (most loops do not require a D parameter; it may make the loop unstable).

Eliminating Proportional Action Although rarely done, the effect of proportional term on the output may be eliminated by setting Kc = 0. Since Kc is also normally a multiplier of the integral coefficient (Ki) and the derivative coefficient (Kr), the CPU makes the computation of these values conditional on the value of Kc as follows:

$Ki = Kc * (Ts / Ti)$ "if Kc 0"

$Ki = Ts / Ti$ "if Kc = 0 (I or ID only)"

$Kr = Kc * (Td / Ts)$ "if Kc 0"

$Kr = Td / Ts$ "if Kc = 0 (ID or D only)"

Velocity Form of the PID Equation

The standard position form of the PID equation computes the actual actuator position. An alternative form of the PID equation computes the change in actuator position. This form of the equation is referred to as the velocity PID equation and is obtained by subtracting the equation at time "n" from the equation at time "n-1".

The velocity equation is given by:

$\Delta M_n = M - M_{n-1}$

$\Delta M_n = Kc * (e_n - e_{n-1}) + Ki * (PV_n - 2 * PV_{n-1} + PV_{n-2})$

The DL05 also combines the integral sum and the initial output into a single term called the bias (Mx). This results in the following set of equations:

$$Mx_o = M_o$$

$$Mx = Ki * e_n + Mx_{n-1}$$

$$M_n = Kc * e_n - Kr(PV_n - PV_{n-1}) + Mx_n$$

The DL05 by default will keep the normalized output M in the range of 0.0 to 1.0. This is done by clamping M to the nearer of 0.0 or 1.0 whenever the calculated output falls outside this range. The DL05 also allows you to specify the minimum and maximum output limit values (within the range 0 to 4095 in binary if using 12 bit unipolar).

NOTE: *The equations and algorithms , or parts of, in this chapter are only for references. Analysis of these equations can be found in most good text books about process control.*

Reset Windup Protection

Reset windup can occur if reset action (integral term) is specified and the computation of the bias term Mx is:

$$Mx = Ki * e_n + Mx_{n-1}$$

For example, assume the output is controlling a valve and the PV remains at some value greater than the setpoint. The negative error (e_n) will cause the bias term (Mx) to constantly decrease until the output M goes to 0 closing the valve. However, since the error term is still negative, the bias will continue to decrease becoming ever more negative. When the PV finally does come back down below the SP, the valve will stay closed until the error is positive for long enough to cause the bias to become positive again. This will cause the process variable to undershoot.

One way to solve the problem is to simply clamp the normalized bias between 0.0 and 1.0. The DL05 CPU does this. However, if this is the only thing that is done, then the output will not move off 0.0 (thus opening the valve) until the PV has become less than the SP. This will also cause the process variable to undershoot.

The DL05 CPU is programmed to solve the overshoot problem by either freezing the bias term, or by adjusting the bias term.

Freeze Bias

If the "Freeze Bias" option is selected when setting up the PID loop (discussed later) then the CPU simply stops changing the bias (Mx) whenever the computed normalized output (M) goes outside the interval 0.0 to 1.0.

$$Mx = Ki * e_n + Mx_{n-1}$$

$$M = Kc * e_n - Kr(PV_n - PV_{n-1}) + Mx$$

$M_n = 0$ "if $M < 0$"
$M_n = M$ "if $0 \leq M \leq 1$"
$M_n = 1$ "if $M > 1$"

$Mx_n = Mx$ "if $0 \leq M \leq 1$"
$Mx_n = Mx_{n-1}$ "otherwise"

Thus in this example, the bias will probably not go all the way to zero so that, when the PV does begin to come down, the loop will begin to open the valve sooner than it would have if the bias had been allowed to go all the way to zero. This action has the effect of reducing the amount of overshoot.

Adjusting the Bias

The normal action of the CPU is to adjust the bias term when the output goes out of range as shown below.

$$Mx = Ki * e_n + Mx_{n-1}$$

$$M = Kc * e_n - Kr(PV_n - PV_{n-1}) + Mx$$

$M_n = 0$ "if $M < 0$"
$M_n = M$ "if $0 \leq M \leq 1$"
$M_n = 1$ "if $M > 1$"

$Mx_n = Mx$ "if $0 \leq M \leq 1$"
$Mx_n = M_n - Kc * e_n - Kr(PV_n - PV_{n-1})$ "otherwise"

By adjusting the bias, the valve will begin to open as soon as the PV begins to come down. If the loop is properly tuned, overshoot can be eliminated entirely. If the output went out of range due to a setpoint change, then the loop probably will oscillate because we must wait for the bias term to stabilize again.

The choice of whether to use the default loop action or to freeze the bias is dependent on the application. **If large, step changes to the setpoint are anticipated, then it is probably better to select the freeze bias option** (see page 8-34).

Bumpless Transfer

The DL05 loop controller provides for bumpless mode changes. A bumpless transfer from manual mode to automatic mode is achieved by preventing the control output from changing immediately after the mode change.

When a loop is switched from Manual mode to Automatic mode, the setpoint and Bias are initialized as follows:

- **Position PID Algorithm**
  SP = PV
  Mx = M (Control Output)

  Velocity PID Algorithm
  SP = PV

The bumpless transfer feature of the DL05 is available in two types: Bumpless I and Bumpless II (see page 8-26). The transfer type is selected when the loop is set up.

Loop Alarms

The DL05 allows the user to specify alarm conditions that are to be monitored for each loop. Alarm conditions are reported to the CPU by setting up the alarms in ***Direct***SOFT programming software using the PID setup alarm dialog when the loop is setup. The alarm features for each loop are:

- **PV Limit** – Specify up to four PV alarm points.

High-High	PV rises above the programmed High-High Alarm Limit.
High	PV rises above the programmed High Alarm Limit.
Low	PV fails below the Low Alarm Limit.
Low-Low	PV fails below the Low-Low Limit.

- **PV Deviation Alarm** – Specify an alarm for High and Low PV deviation from the setpoint (Yellow Deviation). An alarm for High-High and Low-Low PV deviation from the setpoint (Orange Deviation) may also be specified. When the PV is further from the setpoint than the programmed Yellow or Orange Deviation Limit the corresponding alarm bit is activated.

- **Rate of Change** – This alarm is set when the PV changes faster than a specified rate-of-change limit.

- PV **Alarm Hysteresis** – The PV Limit Alarms and PV Deviation Alarms are programmed using threshold values. When the absolute value or deviation exceeds the threshold, the alarm status becomes true. Real-world PV signals have some noise on them, which can cause some fluctuation in the PV value in the CPU. As the PV value crosses an alarm threshold, its fluctuations will cause the alarm to be intermittent and annoy process operators. The solution is to use the PV Alarm Hysteresis feature.

Loop Operating Modes

The DL05 loop controller operates in one of two modes, either *Manual* or *Automatic*.

Manual

In manual mode, the control output is determined by the operator, not the loop controller. While in manual mode, the loop controller will still monitor all of the alarms including High-High, High, Low, Low-Low, Yellow deviation, Orange deviation and Rate-of-Change.

Automatic

In automatic mode, the loop controller computes the control output based on the programmed parameters stored in V-memory. All alarms are monitored while in automatic.

Cascade

Cascade mode is an option with the DL05 PLC and is used in special control applications. If the cascade feature is used, the loop will operate as it would if in automatic mode except for the fact that a cascaded loop has a setpoint which is the control output from another loop.

Special Loop Calculations

Reverse Acting Loop

Although the PID algorithm is used in a direct, or forward, acting loop controller, there are times when a reverse acting control output is needed. The DL05 loop controller allows a loop to operate as reverse acting. With a reverse acting loop, the output is driven in the opposite direction of the error. For example, if SP > PV, then a reverse acting controller will decrease the output to increase the PV.

$$Mx = -Ki * e_n + Mx_{n-1}$$

$$M = -Kc * e_n + Kr(PV_n - PV_{n-1}) + Mx_n$$

Square Root of the Process Variable

Square root is selected whenever the PV is from a device such as an orifice meter which requires this calculation.

Error Squared Control

Whenever error squared control is selected, the error is calculated as:

$$e_n = (SP - PV_n) * ABS(SP - PV_n)$$

A loop using the error squared is less responsive than a loop using just the error, however, it will respond faster with a large error. The smaller the error, the less responsive the loop. Error squared control would typically be used in a PH control application.

Error Deadband Control

With error deadband control, no control action is taken if the PV is within the specified deadband area around the setpoint. The error deadband is the same above and below the setpoint.

Once the PV is outside of the error deadband around the setpoint, the entire error is used in the loop calculation.

$e_n = 0$ "SP - Deadband_Below_SP < PV < SP - Deadband_Above_SP"
$e_n = P - PV_n$ "otherwise"

The error will be squared first if both Error Squared and Error Deadband is selected.

Derivative Gain Limiting

When the coefficient of the derivative term, Kr, is a large value, noise introduced into the PV can result in erratic loop output. This problem is corrected by specifying a derivative gain limiting coefficient, Kd. Derivative gain limiting is a first order filter applied to the derivative term computation, Y_n, as shown below.

$$Y_n = Y_{n-1} + \frac{Ts}{Ts + (\frac{Td}{Kd})} * (PV_n - Y_{n-1})$$

Position Algorithm

$Mx = Ki * e_n + Mx_{n-1}$

$M = Kc * e_n - Kr * (Y_n - Y_{n-1}) + Mx$

Velocity Algorithm

$\Delta M = Kc * (e_n - e_{n-1}) + Ki * e_n - Kr * (Y_n - 2 * Y_{n-1} + Y_{n-2})$

Ten Steps to Successful Process Control

Controllers such as the DL05 PLC provide sophisticated process control features. Automated control systems can be difficult to debug, because a given symptom can have many possible causes. We recommend a careful, step-by-step approach to bringing new control loops online:

Step 1: Know the Recipe

The most important knowledge is – how to produce your product. This knowledge is the foundation for designing an effective control system. A good process *recipe* will do the following:

- Identify all relevant Process Variables, such as temperature, pressure, or flow rates, etc. which need precise control.
- Plot the desired Setpoint values for each process variables for the duration of one process cycle.

Step 2: Plan Loop Control Strategy

This simply means choosing the method the machine will use to maintain control over the Process Variables to follow their Setpoints. This involves many issues and trade-offs, such as energy efficiency, equipment costs, ability to service the machine during production, and more. You must also determine how to generate the Setpoint value during the process, and whether a machine operator can change the SP.

Step 3: Size and Scale Loop Components

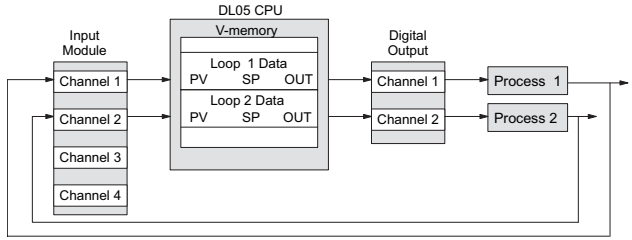
Assuming the control strategy is sound, it is still crucial to *properly size the actuator and properly scale the sensors.*

- Choose an actuator (heater, pump. etc.) which matches the size of the load. An oversized actuator will have an overwhelming effect on your process after a SP change. However, an undersized actuator will allow the PV to lag or drift away from the SP after a SP change or process disturbance.
- Choose a PV sensor which matches the range of interest (and control) for our process. Decide the resolution of control you need for the PV (such as within 2°C), and make sure the sensor input value provides the loop with at least 5 times that resolution (at LSB level). However, an over-sensitive sensor can cause control oscillations, etc. The DL05 provides 12-bit and 15-bit unipolar and bipolar data format options, and a 16–bit unipolar option. This selection affects SP, PV, Control Output and Integrator sum.

Step 4: Select I/O Modules

After deciding the number of loops, PV variables to measure, and SP values, you can choose the appropriate I/O module. Refer to the figure on the next page. In many cases, you will be able to share input or output modules, or use a analog I/O combination module, among several control loops. The example shown sends the PV and Control Output signals for two loops through the same set of modules.

AutomationDirect offers DL05 analog input modules with 4 channels per module that accept 0–20 mA or 4–20 mA signals. Also, analog input and output combination modules are now available. Thermocouple and RTD modules can also be used to maintain temperatures to a 10th of a degree. Refer to the sales catalog for further information on these modules, or find the modules on our website, **www.automationdirect.com.**



Step 5: Wiring and Installation

- After selection and procurement of all loop components and I/O module(s), you can perform the wiring and installation. Refer to the wiring guidelines in Chapter 2 of this manual, and to the **D0–OPTIONS–M** manual. The most common wiring errors when installing PID loop controls are:
- Reversing the polarity of sensor or actuator wiring connections.
- Incorrect signal ground connections between loop components.

Step 6: Loop Parameters

After wiring and installation, choose the loop setup parameters. The easiest method for programming the loop tables is using *Direct*SOFT programming software version (5.0 or later). This software provides PID Setup using dialog boxes to simplify the task. **Note: It is important to understand the meaning of all loop parameters mentioned in this chapter before choosing values to enter.**

Step 7: Check Open Loop Performance

With the sensor and actuator wiring done, and loop parameters entered, we must manually and carefully check out the new control system using the Manual mode).

- Verify that the PV value from the sensor is correct.
- If it is safe to do so, gradually increase the control output up above 0%, and see if the PV responds (and moves in the correct direction!).

Step 8: Loop Tuning

If the Open Loop Test (page 8–40) shows the PV reading is correct and the control output has the proper effect on the process; you can follow the closed loop tuning procedure (see page 8–46). In this step, the loop is tuned so the PV automatically follows the SP.

Step 9: Run Process Cycle

If the closed loop test shows the PV will follow small changes in the SP, consider running an actual process cycle. You will need to have completed the programming which will generate the desired SP in real time. In this step, you may want to run a small test batch of product through the machine, watching the SP change according to the recipe.

WARNING: Be sure the Emergency Stop and power-down provision is readily accessible, in case the process goes out of control. Damage to equipment and/or serious injury to personnel can result from loss of control of some processes.

Step 10: Save Parameters

When the loop tests and tuning sessions are complete, be sure to save all loop setup parameters to disk.

PID Loop Setup

Some Things to Do and Know Before Starting

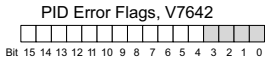
Have your analog module installed and operational before beginning the loop setup (refer to the DL05/06 Option Modules User Manual, D0-OPTIONS-M). The DL05 PLC gets its PID loop processing instructions from V-memory tables. There isn't a PID instruction that can be used in RLL, such as a block, to setup the PID loop control. Instead, the CPU reads the setup parameters from system V-memory locations. These locations are shown in the table below for reference only; they can be used in a RLL program if needed.

Address	Setup Parameter	Data type	Ranges	Read/Write
V7640	Loop Parameter Table Pointer	Octal	V1200 – V7340 V10000 – V17740	Write
V7641	Number of Loops	BCD	1 – 8	Write
V7642	Loop Error Flags	Bits	0 or 1	Read

NOTE: *The V-memory data is stored in SRAM memory. If power is removed from the CPU for an extended period of time, the PID Setup Parameters will be lost. It is recommended to use the MOV instruction, which places the data in non-volatile memory, when setting up the parameters in the ladder program.*

PID Error Flags

The CPU reports any programming errors of the setup parameters in V7640 and V7641. It does this by setting the appropriate bits in V7642 on program-to-run mode transitions.



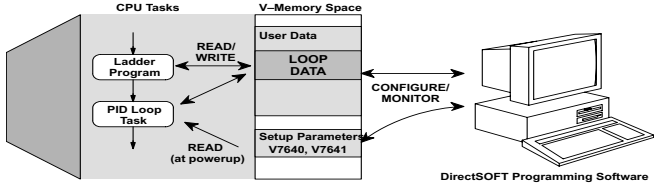
If you use the *Direct*SOFT programming software loop setup dialog box, its automatic range checking prohibits possible setup errors. However, the setup parameters may be written using other methods such as RLL, so the error flag register may be helpful in those cases. The following table lists the errors reported in V7642.

Bit	Error Description (0 = no error, 1 = error)
0	The starting address (in V7640) is out of the lower V-memory range.
1	The starting address (in V7640) is out of the upper V-memory range.
2	The number of loops selected (in V7641) is greater than 8.
3	The loop table extends past (straddles) the boundary at V7377. Use an address closer to V1200.

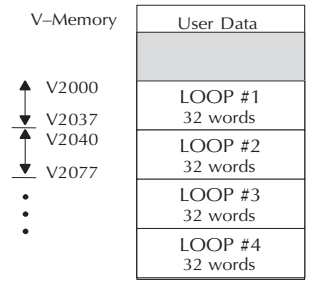
NOTE: *As a quick check, if the CPU is in Run mode and V7642=0000, there are no programming errors.*

Establishing the Loop Table Size and Location

On a PROGRAM-to-RUN mode transition, the CPU reads the loop setup parameters as pictured below. At that moment, the CPU learns the location of the loop table and the



number of loops it configures. Then during the ladder program scan, the PID Loop task uses the loop data to perform calculations, generate alarms, and so on. There are some loop table parameters the CPU will read or write on every loop calculation.



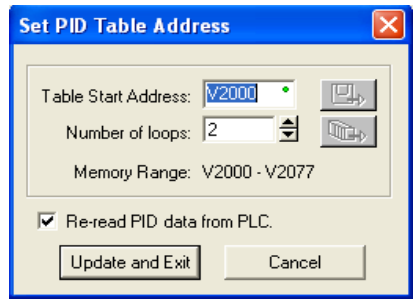
The Loop Table contains data for only the number of loops that are selected. The address for the table is stored in V7641. Each loop configuration occupies 32 words (0 to 37 octal) in the loop table.

For example, consider an application with 4 loops, and V2000 has been chosen as the starting location. The Loop Parameter will occupy V2000 – V2037 for loop 1, V2040 – V2077 for loop 2 and so on. Loop 4 occupies V2140 - V2177.

Determine the block of V-memory to be used for each PID loop. Besides being the beginning of the PID parameter memory block, the first address will be the start of loop 1 parameters. Remember, there are 32 words (0 to 37 octal) needed for each loop. Once you have determined the beginning V-memory address to be used, you can setup and store the PID parameters either directly in your RLL program or by the using PID Setup in *Direct*SOFT programming software.

NOTE: *Whether one or more loops are being setup, this block of V-memory will only be used for the PID loop parameters,* ***do not use this block of memory for anything else in your program****.*

Using *Direct*SOFT programming software is the simplest way to setup the parameters. The PID parameters can be setup either offline or online while developing the user program. The parameters can only be entered in PID setup when the PLC is in the Program mode. Once the parameters have been entered and saved for each loop, changes made through the PID setup can be made, but only in Program Mode. You can type the beginning address in the PID Table Address dialog found when the PID Setup is opened in *Direct*SOFT. This can be seen in the diagram at right. After the address has been entered, the memory range will appear. Also, entering the number of PID loops (1 to 8) will set the total V-memory range for the number of loops entered. After the V-memory address has been entered, the necessary PID parameters for a basic loop operation for each loop can be setup with the dialogs made available.



NOTE: *Have an edited program open, then click on PLC > Setup > PID to access the Setup PID dialog.*

Loop Table Word Definitions

These are the loop parameters associated with each of the four loops available in the DL05. The parameters are listed in the following table. The address offset is in octal, to help you locate specific parameters in the loop table. For example, if a table begins at V2000, then the location for the reset (integral) term is Addr+11, or V2011. Do not use the Word # (in the first column) to calculate addresses.

Word #	Address+Offset	Description	Format	Read on-the-fly***
1	Addr + 0	PID Loop Mode Setting 1	Bits	Yes
2	Addr + 1	PID Loop Mode Setting 2	Bits	Yes
3	Addr + 2	Setpoint Value (SP)	Word/Binary	Yes
4	Addr + 3	Process Variable (PV)	Word/Binary	Yes
5	Addr + 4	Bias (Integrator) Value	Word/Binary	Yes
6	Addr + 5	Control Output Value	Word/Binary	Yes
7	Addr + 6	Loop Mode and Alarm Status	Bits	–
8	Addr + 7	Sample Rate Setting	Word/BCD	Yes
9	Addr + 10	Gain (Proportional) Setting	Word/BCD	Yes
10	Addr + 11	Reset (Integral) Time Setting	Word/BCD	Yes
11	Addr + 12	Rate (Derivative) Time Setting	Word/BCD	Yes
12	Addr + 13	PV Value, Low-low Alarm	Word/Binary	No*
13	Addr + 14	PV Value, Low Alarm	Word/Binary	No*
14	Addr + 15	PV Value, High Alarm	Word/Binary	No*
15	Addr + 16	PV Value, High-high Alarm	Word/Binary	No*
16	Addr + 17	PV Value, deviation alarm (YELLOW)	Word/Binary	No*
17	Addr + 20	PV Value, deviation alarm (RED)	Word/Binary	No*
18	Addr + 21	PV Value, rate-of-change alarm	Word/Binary	No*
19	Addr + 22	PV Value, alarm hysteresis setting	Word/Binary	No*
20	Addr + 23	PV Value, error deadband setting	Word/Binary	Yes
21	Addr + 24	PV low-pass filter constant	Word/BCD	Yes
22	Addr + 25	Loop derivative gain limiting factor setting	Word/BCD	No**
23	Addr + 26	SP value lower limit setting	Word/Binary	Yes
24	Addr + 27	SP value upper limit setting	Word/Binary	Yes
25	Addr + 30	Control output value lower limit setting	Word/Binary	No**
26	Addr + 31	Control output value upper limit setting	Word/Binary	No**
27	Addr + 32	Remote SP Value V-Memory Address Pointer	Word/Hex	Yes
28	Addr + 33	Ramp/Soak Setting Flag	Bit	Yes
29	Addr + 34	Ramp/Soak Programming Table Starting Address	Word/Hex	No**
30	Addr + 35	Ramp/Soak Programming Table Error Flags	Bits	No**
31	Addr + 36	PV direct access, channel number	Word/Hex	Yes
32	Addr + 37	Control output direct access, channel number	Word/Hex	Yes

* Read data only when alarm enable bit transitions from 0 to 1.
** Read data only on PLC Mode change.
*** Read on-the-fly means that the content of V-memory can be changed while the PID loop is in operation.

PID Mode Setting 1 Bit Descriptions (Addr + 00)

The individual bit definitions of the PID Mode Setting 1 word (Addr+00) are listed in the following table.

Bit	PID Mode Setting 1 Description	Read/Write	Bit=0	Bit=1
0	Manual Mode Loop Operation request	Write	–	01 request
1	Automatic Mode Loop Operation request	Write	–	01 request
2	Cascade Mode Loop Operation request	Write	–	01 request
3	Bumpless Transfer select	Write	Mode I	Mode II
4	Direct or Reverse-Acting Loop select	Write	Direct	Reverse
5	Position / Velocity Algorithm select	Write	Position	Velocity
6	PV Linear / Square Root Extract select	Write	Linear	Sq. root
7	Error Term Linear / Squared select	Write	Linear	Squared
8	Error Deadband enable	Write	Disable	Enable
9	Derivative Gain Limit select	Write	Off	On
10	Bias (Integrator) Freeze select	Write	Off	On
11	Ramp/Soak Operation select	Write	Off	On
12	PV Alarm Monitor select	Write	Off	On
13	PV Deviation alarm select	Write	Off	On
14	PV rate-of-change alarm select	Write	Off	On
15	Loop mode is independent from CPU mode when set	Write	Loop with CPU mode	Loop Independent of CPU mode

PID Mode Setting 2 Bit Descriptions (Addr + 01)

The individual bit definitions of the PID Mode Setting 2 word (Addr+01) are listed in the following table.

Bit	PID Mode 2 Word Description	Read/Write	Bit=0	Bit=1
0	Input (PV) and Control Output Range Unipolar/Bipolar select (See Notes 2 and 3)	Write	unipolar	bipolar
1	Input/Output Data Format select (See Notes 2 and 3)	Write	12 bit	15 bit
2	Analog Input filter	Write	off	on
3	SP Input limit enable	Write	disable	enable
4	Integral Gain (Reset) units select	Write	seconds	minutes
5	Select Auto tune PID algorithm	Write	closed loop	open loop
6	Auto tune selection	Write	PID	PI only (rate = 0)
7	Auto tune start (See Note 1)	Read/Write	auto tune cancel/done	force start
8	PID Scan Clock (internal use)	Read	–	–
9	Input/Output Data Format 16 bit select (See Notes 2 and 3)	Write	not 16 bit	select 16 bit
10	Select separate data format for input and output (See Notes 3, and 4)	Write	same format	separate formats
11	Control Output Range Unipolar/Bipolar select See Notes 3, and 4)	Write	unipolar	bipolar
12	Output Data Format select (See Notes 3, and 4)	Write	12 bit	15 bit
13	Output data format 16-bit select (See Notes 3, and 4)	Write	not 16 bit	select16 bit
14–15	Reserved for future use	–	–	–

NOTE 1: *Bit 7 can be used to cancel Autotune mode by setting it to 0.*

NOTE 2: *If the value in bit 9 is 0, then the values in bits 0 and 1 are read. If the value in bit 9 is 1, then the values in bits 0 and 1 are not read, and bit 9 defines the data format (the range is automatically unipolar).*

NOTE 3: *If the value in bit 10 is 0, then the values in bits 0, 1, and 9 define the input and output ranges and data formats (the values in bits 11, 12, and 13 are not read). If the value in bit 10 is 1, then the values in bits 0, 1, and 9 define only the input range and data format, and bits 11, 12, and 13 are read and define the output range and data format.*

NOTE 4: *If bit 10 has a value of 1 and bit 13 has a value of 0, then bits 11 and 12 are read and define the output range and data format. If bit 10 and bit 13 each have a value of 1, then bits 11 and 12 are not read, and bit 13 defines the data format (the output range is automatically unipolar).*

Mode/Alarm Monitoring Word (Addr + 06)

The individual bit definitions of the Mode / Alarm monitoring (Addr+06) word is listed in the following table.

Bit	Mode/Alarm Bit Description	Read/Write	Bit=0	Bit=1
0	Manual Mode Indication	Read	–	Manual
1	Automatic Mode Indication	Read	–	Auto
2	Cascade Mode Indication	Read	–	Cascade
3	PV Input LOW–LOW Alarm	Read	Off	On
4	PV Input LOW Alarm	Read	Off	On
5	PV Input HIGH Alarm	Read	Off	On
6	PV Input HIGH–HIGH Alarm	Read	Off	On
7	PV Input YELLOW Deviation Alarm	Read	Off	On
8	PV Input RED Deviation Alarm	Read	Off	On
9	PV Input Rate-of-Change Alarm	Read	Off	On
10	Alarm Value Programming Error	Read	–	Error
11	Loop Calculation Overflow/Underflow	Read	–	Error
12	Loop in Auto-Tune indication	Read	Off	On
13	Auto-Tune error indication	Read	–	Error
14–15	Reserved for Future Use	–	–	–

Ramp/Soak Table Flags (Addr + 33)

The individual bit definitions of the Ramp/Soak Table Flag (Addr+33) word is listed in the following table.

Bit	Ramp/Soak Flag Bit Description	Read/Write	Bit=0	Bit=1
0	Start Ramp / Soak Profile	Write	–	01 Start
1	Hold Ramp / Soak Profile	Write	–	01 Hold
2	Resume Ramp / soak Profile	Write	–	01 Resume
3	Jog Ramp / Soak Profile	Write	–	01 Jog
4	Ramp / Soak Profile Complete	Read	–	Complete
5	PV Input Ramp / Soak Deviation	Read	Off	On
6	Ramp / Soak Profile in Hold	Read	Off	On
7	Reserved	Read	–	–
8–15	Current Step in R/S Profile	Read	decode as byte (hex)	

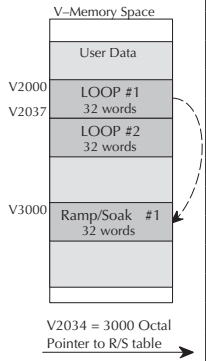
Bits 8–15 must be read as a byte to indicate the current segment number of the Ramp/Soak generator in the profile. This byte will have the values 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, and 10, which represent segments 1 to 16 respectively. If the byte=0, then the Ramp/Soak table is not active.

Ramp/Soak Table Location (Addr + 34)

Each loop that you configure has the option of using a built-in Ramp/Soak generator dedicated to that loop. This feature generates SP values that follow a profile. To use the Ramp Soak feature, you must program a separate table of 32 words with appropriate values. A *Direct*SOFT dialog box makes this easy to do.

In the loop table, the Ramp/Soak Table Pointer at Addr+34 must point to the start of the ramp/soak data for that loop. This may be anywhere in user memory, and does not have to adjoin to the Loop Parameter table, as shown to the left. Each R/S table requires 32 words, regardless of the number of segments programmed.

The ramp/soak table parameters are defined in the table below. Further details are in the section on Ramp/Soak Operation in this chapter.



Addr Offset	Step	Description	Addr Offset	Step	Description
+ 00	1	Ramp End SP Value	+ 20	9	Ramp End SP Value
+ 01	1	Ramp Slope	+ 21	9	Ramp Slope
+ 02	2	Soak Duration	+ 22	10	Soak Duration
+ 03	2	Soak PV Deviation	+ 23	10	Soak PV Deviation
+ 04	3	Ramp End SP Value	+ 24	11	Ramp End SP Value
+ 05	3	Ramp Slope	+ 25	11	Ramp Slope
+ 06	4	Soak Duration	+ 26	12	Soak Duration
+ 07	4	Soak PV Deviation	+ 27	12	Soak PV Deviation
+ 10	5	Ramp End SP Value	+ 30	13	Ramp End SP Value
+ 11	5	Ramp Slope	+ 31	13	Ramp Slope
+ 12	6	Soak Duration	+ 32	14	Soak Duration
+ 13	6	Soak PV Deviation	+ 33	14	Soak PV Deviation
+ 14	7	Ramp End SP Value	+ 34	15	Ramp End SP Value
+ 15	7	Ramp Slope	+ 35	15	Ramp Slope
+ 16	8	Soak Duration	+ 36	16	Soak Duration
+ 17	8	Soak PV Deviation	+ 37	16	Soak PV Deviation

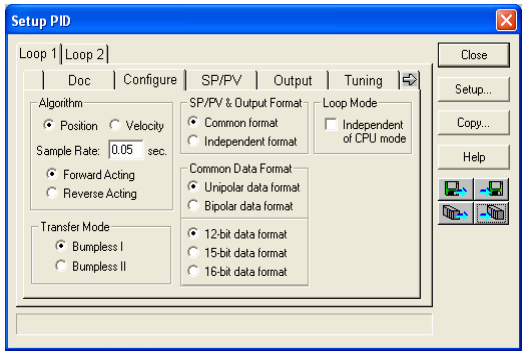
Ramp/Soak Table Programming Error Flags (Addr + 35)

The individual bit definitions of the Ramp/Soak Table **Programming Error Flags** word (Addr+35) is listed in the following table. Further details are given in the PID Loop Mode section and in the PV Alarm section later in chapter 8.

Bit	R/S Error Flag Bit Description	Read/Write	Bit=0	Bit=1
0	Starting Addr out of lower V-memory range	Read	–	Error
1	Starting Addr out of upper V-memory range	Read	–	Error
2–3	Reserved for Future Use	–	–	–
4	Starting Addr in System Parameter V-memory Range	Read	–	Error
5–15	Reserved for Future Use	–	–	–

Configure the PID Loop

Once the PID table is established in V-memory, configuring the PID loop continues with the *Direct*SOFT PID setup configuration dialog. You will need to check and fill in the data required to control the PID loop. Select Configure and the following dialog will appear for this process.

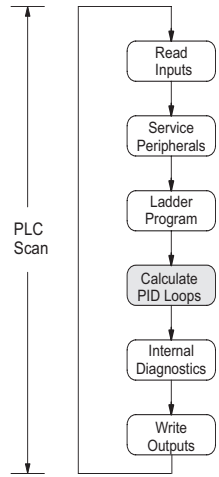


Select the Algorithm Type

Chose either *Position* or *Velocity*. The default algorithm is Position. This is the choice for most applications which include heating and cooling loops as well as most position an level control loops. A typical velocity control will consist of a process variable such as a flow totalizer in a flow control loop.

Enter the Sample Rate

The main tasks of the CPU fall into categories as shown to the right. The list represents the tasks done when the CPU is in Run Mode, on each PLC scan. Note that PID loop calculations occur after the ladder logic task.



The *sample rate* of a control loop is simply the frequency of the PID calculation. Each calculation generates a new control output value. With the DL05 CPU, you can set the sample rate of a loop from 50 ms to 99.99 seconds. Most loops do not require a fresh PID calculation on every PLC scan. Some loops may need to be calculated only once in 1000 scans.

Enter 0.05 sec., or the sample rate of your choice, for each loop, and the CPU automatically schedules and executes PID calculations on the appropriate scans.

Select Forward/Reverse

It is important to know which direction the control output will respond to the error (SP-PV), either *forward* or *reverse.* A forward (direct) acting control loop means that whenever the control output increases, the process variable will also increase. The control output of most PID loops are forward acting, such as a heating control loop. An increase in heat applied will increase the PV (temperature).

A reverse acting control loop is one where an increase in the control output results in a decrease in the PV. A common example of this would be a refrigeration system, where an increase in the cooling input causes a decrease in the PV (temperature).

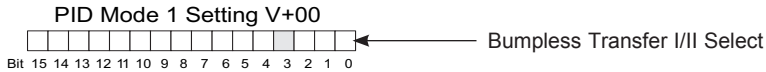
The Transfer Mode

Choose either Bumpless I or Bumpless II to provide a smooth transition of the control output from Manual Mode to Auto Mode. Choosing Bumpless I will set the SP equal to the PV when the control output is switched from Manual to Auto. If this is not desired, choose BumplessII.

The characteristics of Bumpless I and II transfer types are listed in the chart below. Note that their operation also depends on which PID algorithm you are using, the position or velocity form of the PID equation. Note that you must use Bumpless Transfer type I when using the velocity form of the PID algorithm.

TransferType	Transfer Select Bit 3	PID Algorithm	Manual-to-Auto Transfer Action	Auto-to-Cascade Transfer Action
Bumpless Transfer I	0	Position	Forces Bias = Control Output Forces SP = PV	Forces Major Loop Output = Minor Loop PV
		Velocity	Forces SP = PV	Forces Major Loop Output = Minor Loop PV
Bumpless Transfer II	1	Position	Forces Bias = Control Output	None
		Velocity	None	None

The transfer type can also be selected in a RLL program by setting bit 3 of PID Mode 1, V+00 setting as shown.

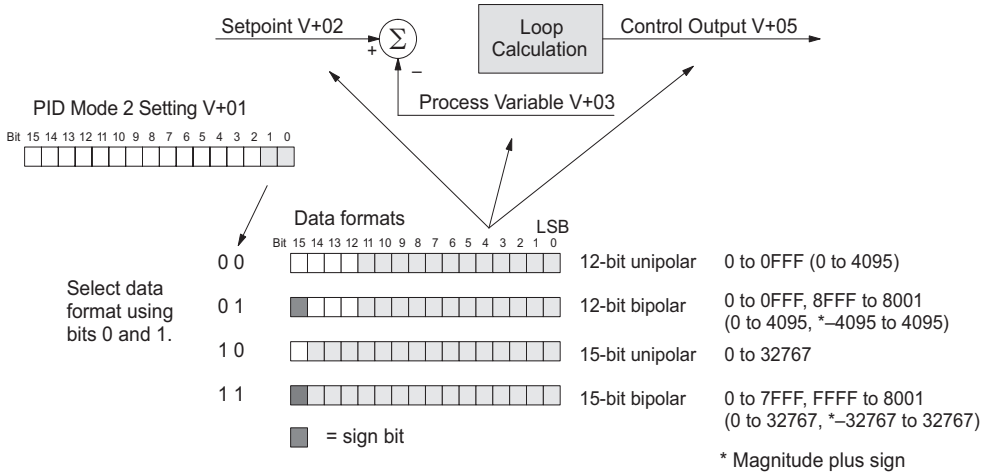


SP/PV & Output Format

This block allows you to select either *Common format* or *Independent format*. Common format is the default and is most commonly used. With this format both SP/PV and Output will have the same data structure. Both will have the same number of bits and either bipolar or unipolar. If Independent format is selected, the data structure selections will be grayed out. The reason for this is that they become independently selectable in the *SP/PV* and the *Output* dialogs.

Common Data Format

Select either *Unipolar data format* (which is positive data only) in 12 bit (0 to 4095), 15 bit (0 to 32767) or 16 bit (0 to 65535) format, or *Bipolar data format* which ranges from negative to positive (-4095 to 4095 or -32767 to 32767) and requires a sign bit. Bipolar selection displays input/output as magnitude plus sign, not two's compliment. The bipolar selection is not available when 16-bit data format is selected.



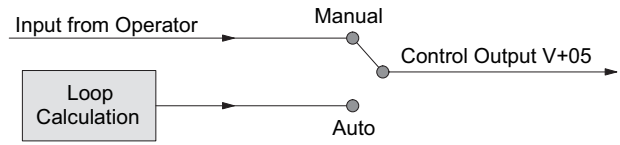
The data format determines the numerical interface between the PID loop and the PV sensor, and the control output device. This selects the data format for both the SP and the PV.

Loop Mode

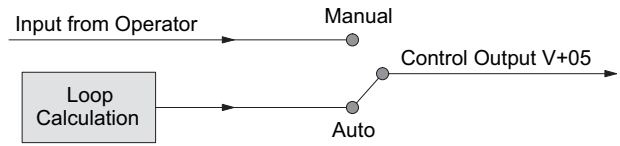
Loop Mode is a special feature that allows the PID loop controller to perform closed-loop control while the CPU is in the Program Mode. Careful thought must be taken before using this feature called *Independent of CPU mode* in the dialog. Before continuing with the PID setup, a knowledge of the three PID loop modes will be helpful.

The DL05 provides the three standard control modes: *Manual, Automatic,* and *Cascade*. The sources of the three basic variables SP, PV and control output are different for each mode.

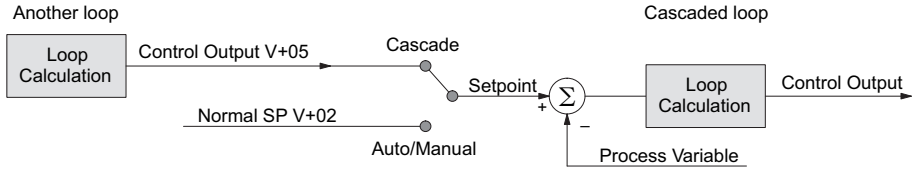
In Manual Mode, the loop is not executing PID calculations (however, loop alarms are still active). With regard to the loop table, the CPU stops writing values to location V+05 (control output) for that loop. *It is expected that an operator or other intelligent source* is manually controlling the output by observing the PV and writing data to the control output as necessary to keep the process under control. The drawing below shows the equivalent schematic diagram of manual mode operation.



In Automatic Mode, the loop operates normally and generates new control output values. It calculates the PID equation and writes the result in location V+05 every sample period of that loop. The equivalent schematic diagram is shown below.

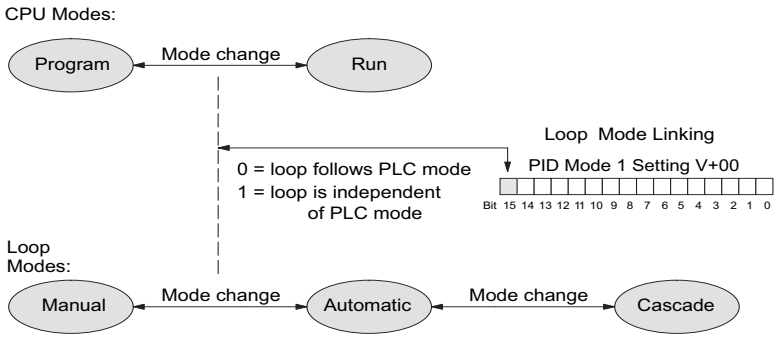


In Cascade Mode, the loop operates as it does in Automatic Mode, with one important difference. The data source for the SP changes from its normal location at V+02 to using the control output value, V+05, from another loop. So in Auto or Manual modes, the loop calculation uses the data at V+02. In Cascade Mode, the loop calculation reads the control output from another loop's parameter table, V+05.



As pictured below, a loop can be changed from one mode to another, but *cannot go from Manual Mode directly to Cascade, or vice versa*. This mode change is prohibited because a loop would be changing two data sources at the same time, and could cause a loss of control.

Once the CPU is operating in the Run Mode, the normal operation of the PID loop controller is to read the loop data and perform calculations on each scan of the RLL program. When the CPU is placed in the Program Mode, the RLL program halts operation and all PID loops are automatically put into the Manual Mode. The PID parameters can then be changed if desired. Similarly, by placing the CPU in the Run mode, the PID loops are returned to the operational mode which they were previously in, i.e., Manual, Automatic and Cascade. With this selection you automatically affect the modes by changing the CPU mode.



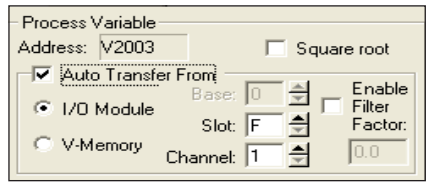
If bit 15 is set to one, then the loops will run independent of the CPU mode. It is like having two independent processors in the CPU...one is running the RLL program and the other is running the process loops.

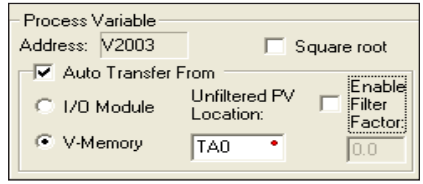
Having the ability to run loops independently of the RLL program makes it feasible to make a ladder logic change while the process is still running. This is especially beneficial for large-mass continuous processes that are difficult or costly to interrupt. The independent of CPU is the feature used for this.

If you need to operate the PID loops while the RLL program is halted, in Program Mode, either select the Independent of CPU mode in the dialog or edit your program to set and reset bit 15 of PID Mode 1 word (V+00) in your RLL program. If the bit is set to a zero, the loop will follow the CPU mode, then when the CPU is placed in the Program Mode, all loops will be forced into the Manual Mode.

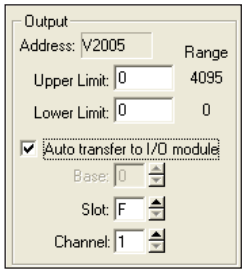
When Independent of CPU mode is used, you should also set the PV to be read directly from an analog input module. This can easily be done in the PID setup dialog, SP/PV.

The SP/PV dialog has a block entitled Process Variable. There is a block within this block called Auto Transfer From (from analog input) with the information grayed out. Checking the box to the left of the Auto Transfer From will highlight the information. There are two choices to select for the auto transfer. The first choice is I/O Module. If this is selected, the Slot number "F" and the Channel number will automatically appear. Since there is only one option slot in the DL05, the letter F represents the default option slot. Also, select the analog input channel of your choice.





The second choice is V-Memory. When this is selected, the V-memory address from where the PV is transferred from must be specified.

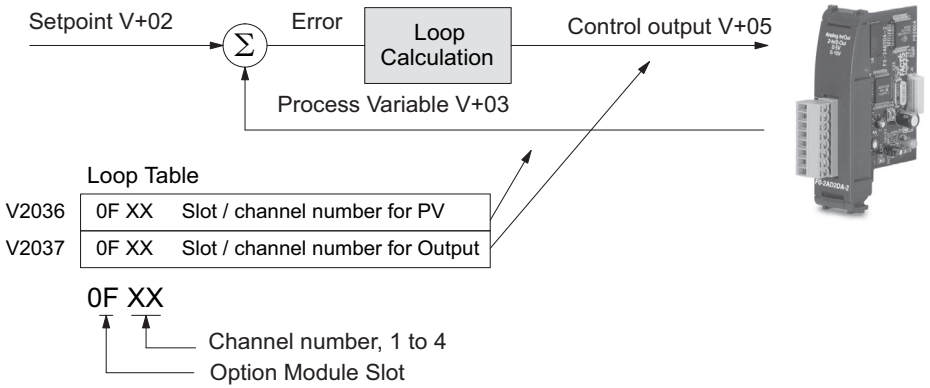


Whichever method of auto transfer is used, it is recommended to check the Enable Filter Factor (a low pass filter) and specify the coefficient.

You should also select the analog output for the control output to be transferred to. This is done in the PID setup Output dialog shown here. The block of information in this dialog is "grayed-out" until the box next to Auto transfer to I/O module is checked. Once checked, the slot number defaults to "F" like the input, and the analog output channel can be selected.

NOTE: *To make changes to any loop table parameters, the PID loop must be in Manual mode and the PLC must be stopped. If you have selected to operate the PID loop independent of the CPU mode, then you must take certain steps to make it possible to make loop parameter changes. You can temporarily make the loops follow the CPU mode by changing bit 15 to 0. Then you will be able to place the loop into Manual Mode using* ***Direct****SOFT . After you change the loop's parameter setting/s, just restore bit 15 to a value of 1 to re-establish PID operation independent of CPU.*

You may optionally configure each loop to access its analog I/O (PV and control output) by placing proper values in the associated loop table registers in your RLL program. The following figure shows the loop table parameters at V+36 and V+37 and their auto transfer role to access the analog values directly.

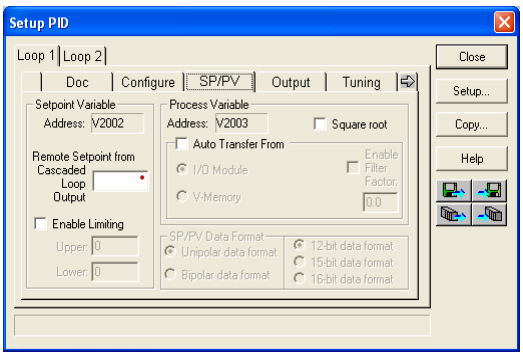


When these loop table parameters are programmed directly, a value of "0F02" in register V2036 directs the loop controller to read the PV data from channel 1 of the analog input. A value of "0000" in either register tells the loop controller not to access the corresponding analog value directly. In that case, ladder logic must be used to transfer the value between the analog input and the loop table.

NOTE: *When auto transfer to/from I/O is used, the analog data for all of the channels on the analog module cannot be accessed by any other method, i.e., pointer or multiplex.*

SP/PV Addresses

An SP/PV dialog will be made available to setup how the setpoint (SP) and the process variable (PV) will be used in the loop. If this loop is the minor loop of a cascaded pair, enter that control output address in the *Remote SP from Cascaded Loop Output* area. It is sometimes desirable to limit the range of setpoint values allowed to be entered. To activate this feature, check the box next to *Enable Limiting*. This will activate the *Upper* and *Lower* fields for the values to be entered. Set the limits around the SP value to prevent an operator from entering a setpoint value outside of a safe range. The *Square root* box is only checked for certain PID loops, such as a flow control loop. If the *Auto transfer from I/O module* is selected, a first-order low-pass filter can be used by checking the *Enable Filter* box. The filter coefficient is user specified.

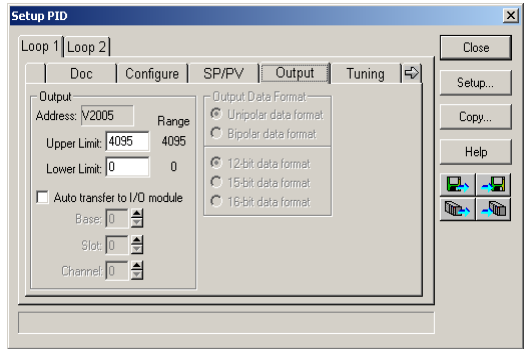


The use of this filter is recommended during closed loop auto-tuning. If the Independent format had been checked previously, make the Data format selections here.

NOTE: *The SP/PV dialog can be left as it first appears for basic PID operation.*

Set Control Output Limits

Another dialog that will be available in the PID setup will be the Output dialog. The control output address, V+05, (determined by the PID loop table beginning address) will be in view. Enter the output range limits, *Upper Limit* and *Lower Limit,* that will meet the requirement of the process and which will agree with the data format that has been selected. For a basic PID operation using a 12-bit output module, set the Upper Limit to 4095 and leave the Lower Limit set to 0. Check the box next for *Auto transfer to I/O module* if there is a need to send the control output to a certain analog output module, as in the case of using the Loop Mode independent of CPU Mode; otherwise, the PID output signal cannot control the analog output when the PLC is not in RUN Mode. If the *Auto transfer to I/O module* feature is checked, all channels of the module must be used for PID control outputs. If Independent format has been previously chosen, the *Output Data Format* will need to be setup here, that is, select Unipolar or Bipolar format and the bit structure. This area is not available and is grayed out if *Common format* has been chosen (see page 8-26).

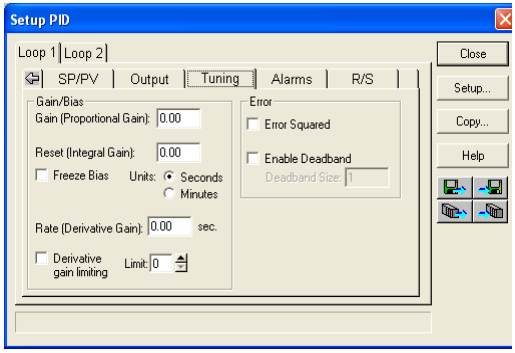


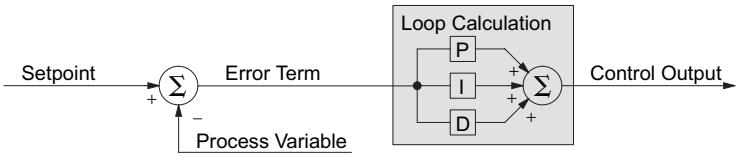
WARNING: If the Upper Limit is set to zero, the output will never get above zero. In effect, there will be no control output. The default value is 0, so this value MUST be changed.

Enter PID Parameters

Another PID setup dialog, Tuning, is for entering the PID parameters shown as: Gain (Proportional Gain), Reset (Integral Gain) and Rate (Derivative Gain).

Recall the position and velocity forms of the PID loop equations which were introduced earlier. The equations basically show the three components of the PID calculation: Proportional Gain (P), Integral Gain (I) and Derivative Gain (D). The following diagram shows a form of the PID calculation in which the control output is the sum of the proportional gain, integral gain and derivative gain. With each calculation of the loop, each term receives the same error signal value.





The P, I and D gains are 4-digit BCD numbers with values from 0000 to 9999. They contain an implied decimal point in the middle, so the values are actually 00.00 to 99.99. Some gain values have units – Proportional gain has no unit, Integral gain may be selected in seconds or in minutes, and Derivative gain is in seconds.

Gain (Proportional Gain) – This is the most basic gain of the three. Values range from 0000 to 9999, but they are used internally as xx.xx. An entry of "0000" effectively removes the proportional term from the PID equation. This accommodates applications which need integral-only loops.

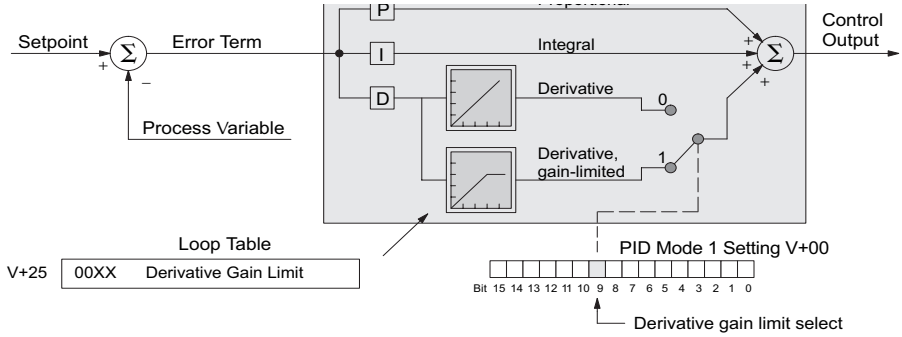
Reset (Integral Gain) – Values range from 0001 to 9998, but they are used internally as xx.xx. An entry of "0000" or "9999" causes the integral gain to be "∞", effectively removing the integrator term from the PID equation. This accommodates applications which need proportional-only loops. The units of integral gain may be either seconds or minutes, as shown in the above dialog.

Rate (Derivative Gain) – Values which can be entered range from 0001 to 9999, but they are used internally as XX.XX. An entry of "0000" allows removal of the derivative term from the PID equation (a common practice). This accommodates applications which require only proportional and/or integral loops. Most control loops will operate as a PI loop.

NOTE: *You may elect to leave the tuning dialog blank and enter the tuning parameters in the* ***Direct****SOFT programming software PID View.*

Derivative Gain Limiting

The derivative gain (rate) has an optional gain-limiting feature. This is provided because the derivative gain reacts badly to PV signal noise or other causes of sudden PV fluctuations. The function of the gain-limiting is shown in the diagram below.



The gain limit can be particularly useful during loop tuning. Most loops can tolerate only a little derivative gain without going into uncontrolled oscillations.

If this option is checked, a *Limit* of 0 to 20 must also be entered.

***NOTE**: When first configuring a loop, it's best to use the standard error term until after the loop is tuned. Once the loop is tuned, you will be able to tell if these functions will enhance control. The Error Squared and/or Enable Deadband can be selected later in the PID setup. Also, values are not required to be entered in the Tuning dialog, but they can be set later in the **Direct**SOFT PID View.*

Error Term Selection

The error term is internal to the CPUs PID loop controller, and is generated again in each PID calculation. Although its data is not directly accessible, you can easily calculate it by subtracting: Error = (SP–PV). The PID calculation operates on this value linearly to give the result. However, a few applications can benefit from non-linear control. The Error-squared method of non-linear control exaggerates large errors and diminishes small error.

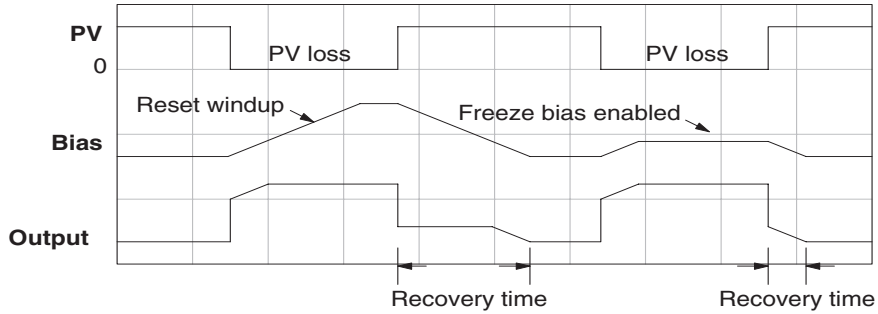
Error Squared – When selected, the squared error function simply squares the error term (but preserves the original algebraic sign), which is used in the calculation. This affects the Control Output by diminishing its response to smaller error values, but maintaining its response to larger errors. Some situations in which the error squared term might be useful:

- Noisy PV signal – using a squared error term can reduce the effect of low-frequency electrical noise on the PV, which will make the control system jittery. A squared error maintains the response to larger errors.
- Non-linear process – some processes (such as chemical pH control) require non-linear controllers for best results. Another application is surge tank control, where the Control Output signal must be smooth.

Enable Deadband – When selected, the enable deadband function takes a range of small error values near zero, and simply substitutes zero as the value of the error. If the error is larger than the deadband range, then the error value is used normally.

Freeze Bias

The term *reset windup* refers to an undesirable characteristic of integrator behavior which occurs naturally under certain conditions. Refer to the figure below. Suppose the PV signal becomes disconnected, and the PV value goes to zero. While this is a serious loop fault, it is made worse by reset windup. Notice the bias (reset) term keeps integrating normally during the PV disconnect, until its upper limit is reached. When the PV signal returns, the bias value is saturated (windup) and takes a long time to return to normal. The loop output consequently has an extended recovery time. Until recovery, the output level is wrong and causes further problems.



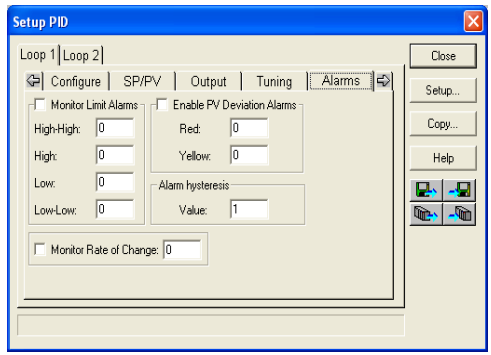
In the second PV signal loss episode in the figure, the freeze bias feature is enabled. It causes the bias value to freeze when the control output goes to its range limit. Much of the reset windup is thus avoided, and the output recovery time is much less.

For most applications, the freeze bias feature will work with the loop as described above. It is suggested to enable this feature by selecting it in the dialog. Bit 10 of PID Mode 1 Setting (V+00) word can also be set in RLL.

NOTE: *The freeze bias feature stops the bias term from changing when the control output reaches the end of the data range. If you have set limits on the control output other than the range (i.e, 0–4095 for a unipolar/12-bit loop), the bias term still uses the end of range for the stopping point and bias freeze will not work.*

Setup the PID Alarms

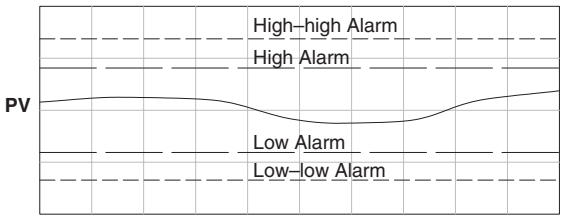
Although the setup of the PID alarms is optional, you surely would not want to operate a process without monitoring it. The performance of a process control loop may generally be measured by how closely the process variable matches the setpoint. Most process control loops in industry operate continuously, and will eventually lose control of the PV due to an error condition. Process alarms are vital in early discovery of a loop error condition and can alert plant personnel to manually control a loop or take other measures until the error condition has been repaired.



The alarm thresholds are fully programmable, and each type of alarm may be independently enabled and monitored. The following diagram shows the Alarm dialog in the PID setup which simplifies the alarm setup.

Monitor Limit Alarms

Checking this box will allow all of the PV limit alarms to be monitored once the limits are entered. The PV absolute value alarms are organized as two upper and two lower alarms. The alarm status is false as long as the PV value remains in the region between the upper and lower alarms, as shown below. The alarms nearest the safe zone are named *High Alarm* and *Low Alarm*. If the loop loses control, the PV will cross one of these thresholds first. Therefore, you can program the appropriate alarm threshold values in the loop table locations shown below to the right. The data format is the same as the PV and SP (12-bit or 15-bit). The threshold values for these alarms should be set to give an operator an early warning if the process loses control.



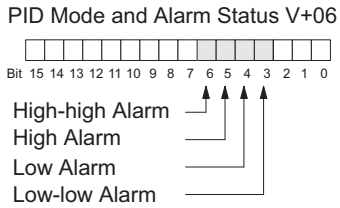
Loop Table

V+16	XXXX	High-high Alarm
V+15	XXXX	High Alarm
V+14	XXXX	Low Alarm
V+13	XXXX	Low-low Alarm

NOTE: *The Alarm dialog can be left as it first appears, without alarm entries. The alarms can then be setup in the* ***Direct****SOFT PID View.*

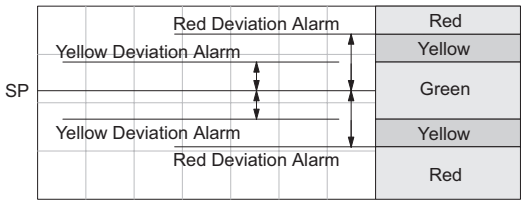
If the process remains out of control for some time, the PV will eventually cross one of the outer alarm thresholds, named High-high alarm and Low-low alarm. Their threshold values are programmed using the loop table registers listed above. A High-high or Low-low alarm indicates a serious condition exists, and needs the immediate attention of the operator.

The PV Absolute Value Alarms are reported in the four bits in the PID Mode and Alarm Status word in the loop table, as shown to the right. We highly recommend using ladder logic to monitor these bits. The bit-of-word instructions make this easy to do. Additionally, you can monitor PID alarms using *Direct*SOFT programming software.



PV Deviation Alarms

The PV Deviation Alarms monitor the PV deviation with respect to the SP value. The deviation alarm has two programmable thresholds, and each threshold is applied equally above and below the current SP value. In the figure below, the smaller deviation alarm is called the "Yellow Deviation", indicating a cautionary condition for the loop. The larger deviation alarm is called the "Red Deviation", indicating a strong error condition for the loop. The threshold values use the loop parameter table locations V+17 and V+20 as shown.

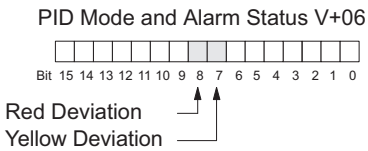


Loop Table

V+17	XXXX	Yellow Deviation Alarm
V+20	XXXX	Red Deviation Alarm

The thresholds define zones, which fluctuate with the SP value. The green zone which surrounds the SP value represents a safe (no alarm) condition. The yellow zones lie outside the green zone, and the red zones are beyond those.

The PV Deviation Alarms are reported in the two bits in the PID Mode and Alarm Status word in the loop table, as shown to the right. We highly recommend using ladder logic to monitor these bits. The bit-of-word instructions make this easy to do. Additionally, you can monitor PID alarms using *Direct*SOFT programming software.



The PV Deviation Alarm can be independently enabled and disabled from the other PV alarms, using bit 13 of the PID Mode 1 Setting V+00 word.

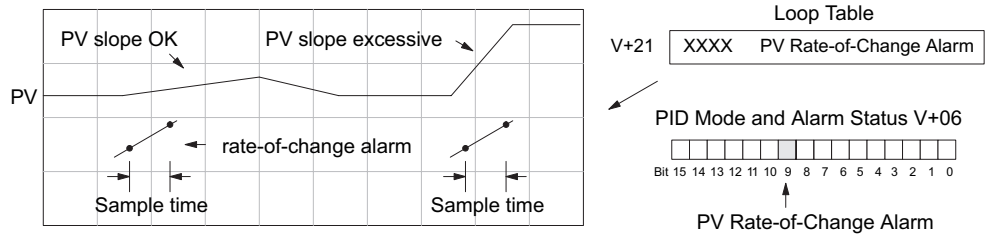
Remember the alarm hysteresis feature works in conjunction with both the deviation and absolute value alarms, and is discussed at the end of this section.

***NOTE**: PID deviation alarm only works in Auto mode.*

PV Rate-of-Change Alarm

An excellent way to get an early warning of a process fault is to monitor the *rate-of-change* of the PV. Most batch processes have large masses and slowly-changing PV values. A relatively fast-changing PV will result from a broken signal wire for either the PV or control output, a SP value error, or other causes. If the operator responds to a PV Rate-of-Change Alarm quickly and effectively, the PV absolute value will not reach the point where the material in process would be ruined.

The DL05 loop controller provides a programmable PV Rate-of-Change Alarm, as shown below. The rate-of-change is specified in PV units change per loop sample time. This value is programmed into the loop table location V+21.



As an example, suppose the PV is the temperature for your process, and you want an alarm whenever the temperature changes faster than 15 degrees/minute. The PV counts per degree and the loop sample rate must be known. Then, suppose the PV value (in V+03 location) represents 10 counts per degree, and the loop sample rate is 2 seconds. Use the formula below to convert our engineering units to counts/sample period:

$$\text{Alarm Rate-of-Change} = \frac{15 \text{ degrees}}{1 \text{ minute}} \text{ X } \frac{10 \text{ counts / degree}}{30 \text{ loop samples / min.}} = \frac{150}{30} = 5 \text{ counts / sample period}$$

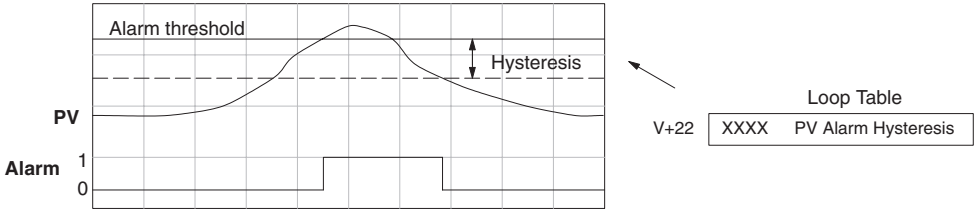
From the calculation result, you would program the value 5 in the loop table for the rate-of-change. The PV Rate-of-Change Alarm can be independently enabled and disabled from the other PV alarms, using bit 14 of the PID Mode 1 Setting V+00 word.

The alarm hysteresis feature (discussed next) does not affect the Rate-of-Change Alarm.

PV Alarm Hysteresis

The PV Absolute Value Alarm and PV Deviation Alarm are programmed using threshold values. When the absolute value or deviation exceeds the threshold, the alarm status becomes true. Real-world PV signals have some noise on them, which can cause some fluctuation in the PV value in the CPU. As the PV value crosses an alarm threshold, its fluctuations cause the alarm to be intermittent and annoy process operators. The solution is to use the PV Alarm Hysteresis feature.

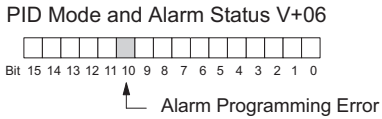
The PV Alarm Hysteresis amount is programmable from 1 to 200 (binary/decimal). When using the PV Deviation Alarm, the programmed hysteresis amount must be less than the programmed deviation amount. The figure below shows how the hysteresis is applied when the PV value goes past a threshold and descends back through it.



The hysteresis amount is applied after the threshold is crossed, and toward the safe zone. In this way, the alarm activates immediately above the programmed threshold value. It delays turning off until the PV value has returned through the threshold by the hysteresis amount.

Alarm Programming Error

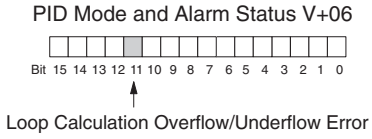
The PV Alarm threshold values must have certain mathematical relationships to be valid. The requirements are listed below. If not met, the Alarm Programming Error bit will be set, as indicated to the right.



- PV Absolute Alarm
  value requirements: Low-low < Low < High < High-high
- PV Deviation Alarm requirements: Yellow < Red

Loop Calculation Overflow/Underflow Error

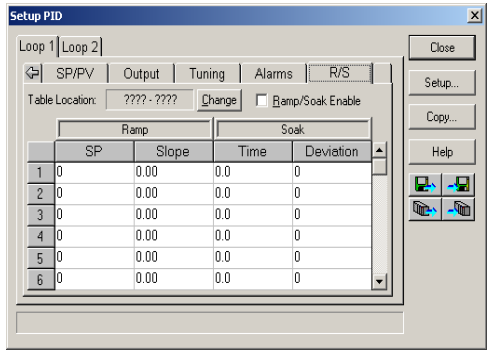
This error occurs whenever the output reaches its upper or lower limit and the PV does not reach the setpoint. A typical example might be when a valve is stuck, the output is at its limit, but the PV has not reached setpoint.



NOTE: *Overflow/underflow can be alarmed in PID View. The optional C-more IO panel (see the automationdirect.com website) can also be set up to read these error bits using the PID Faceplate templates.*

Ramp/Soak

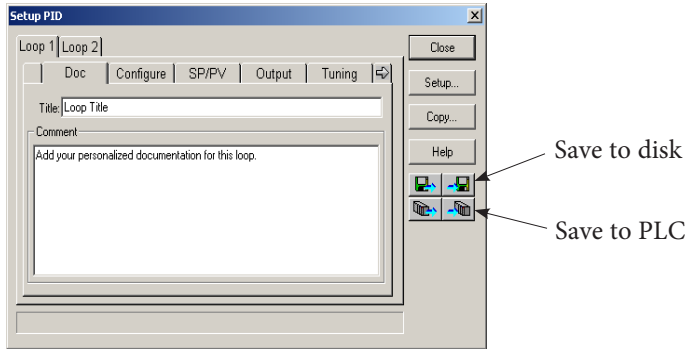
R/S (Ramp/Soak) is the last dialog available in the PID setup. The basic PID does not require any entries to be made in order to operate the PID loop. Ramp/Soak will be discussed in another section.



Complete the PID Setup

Once you have filled in the necessary information for the basic PID setup, the configuration should be saved. The icons on the Setup PID dialog will allow you to save the configuration to the PLC and to disk. The save to icons have the arrow pointing to the PLC and disk. The read from icons have the arrows pointing away from the PLC and disk.

An optional feature is available with the Doc tab in the Setup PID window. You enter a name and description for the loop. This is useful if there is more than one PID loop in your application.



NOTE: *It is good practice to save your project after setting up the PID loop by selecting* ***File*** *from the menu toolbar, then* ***Save project > to disk****. In addition to saving your entire project, all the PID parameters are also saved.*

PID Loop Tuning

Once you have set up a PID loop, it must be tuned in order for it to work. The goal of loop tuning is to adjust the loop gains so the loop has optimal performance in dynamic conditions. The quality of a loop's performance may generally be judged by how well the PV follows the SP after a SP step change. It is important to keep in mind that understanding the process is fundamental to getting a well designed control loop. Sensors must be in appropriate locations and valves must be sized correctly with appropriate trim. **PID control does not have *typical* values**. There isn't one control process that is identical to another.

Manual Tuning vs. Auto Tuning

You may enter the PID gain values to tune your loops (manual tuning), or you can rely on the PID processing "engine" in the CPU to automatically calculate the gain values (auto tuning). Most experienced process engineers will have a favorite method; the DL05 will accommodate either preference. The use of auto tuning can eliminate much of the trial-and-error of the manual tuning approach, especially if you do not have a lot of loop tuning experience. However, performing the auto tuning procedure will get the gains close to optimal values, but additional manual tuning can get the gain values to their optimal values.

WARNING: Only authorized personnel fully familiar with all aspects of the process should make changes that affect the loop tuning constants. Using the loop auto tune procedures will affect the process, including inducing large changes in the control output value. Make sure you thoroughly consider the impact of any changes to minimize the risk of injury to personnel or damage to equipment. The auto tune in the DL05 is not intended to be used as a replacement for your process knowledge.

Open-Loop Test

Whether you use manual or auto tuning, it is very important to verify basic characteristics of a newly-installed process before attempting to tune it. With the loop in Manual Mode, verify the following items for each new loop.

- **Setpoint** – verify that the SP source can generate a setpoint. Put the PLC in Run Mode and leave the loop in Manual Mode, then monitor the loop table location V+02 to see the SP value(s). (If you are using the ramp/soak generator, test it now).
- **Process Variable** – verify that the PV value is an accurate measurement, and the PV data arriving in the loop table location V+03 is correct. If the PV signal is very noisy, consider filtering the input either through hardware (RC low-pass filter), or using the filter in this chapter.
- **Control Output** – if it is safe to do so, manually change the output a small amount (perhaps 10%) and observe its affect on the process variable. Verify the process is direct-acting or reverse acting, and check the setting for the control output (inverted or non-inverted). Make sure the control output upper and lower limits are not equal to each other.
- **Sample Rate** – while operating open-loop, this is a good time to find the ideal sample rate (procedure given earlier in this chapter). However, if you are going to use auto tuning, the auto tuning procedure will automatically calculate the sample rate in addition to the PID gains.

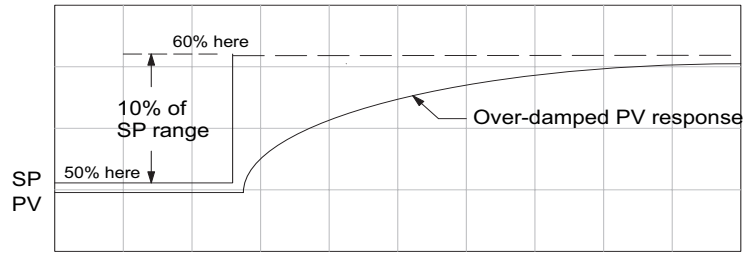
Manual Tuning Procedure

It is not necessary to try to obtain the best values for the P, I and D parameters in the PID loop by trial and error. Following is a typical procedure for tuning a temperature control loop which you may use to tune your loop.

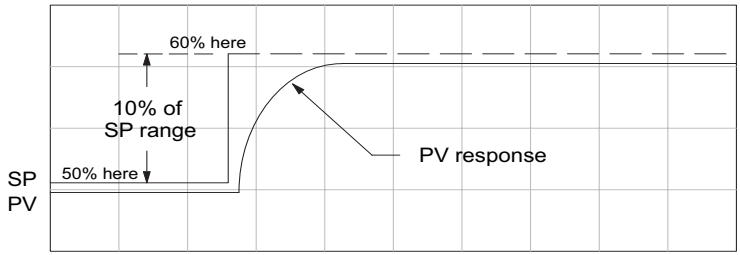
Monitor the values of SP, PV and CV with a loop trending instrument or use the PID View feature in *Direct*SOFT programming software (see page 8-48).

NOTE: *We recommend using the PID View Tuning and Trending window to select manual for the vertical scale feature, for both SP/PV area and Bias/Control Output areas. The auto scaling feature would otherwise change the vertical scale on the process parameters and add confusion to the loop tuning process.*

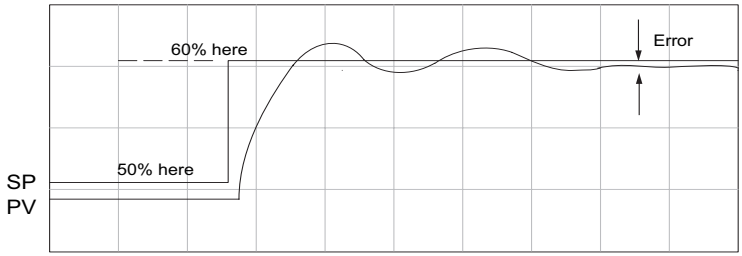
- Adjust the gains so the Proportional Gain = 0.5 or 1.0 (1.0 is a good value based on experience), Integral Gain = 9999 (this basically eliminates reset) and Derivative Gain =0000. This disables the integrator and derivative terms, and provides some proportional gain.
- Check the bias value in the PID View and set it to zero.
- Set the SP to a value equal to 50% of the full range.
- Now, select Auto Mode. If the loop will not stay in Auto Mode, check the troubleshooting tips at the end of this chapter. Allow the PV to stabilize around the 50% point of the range.
- Change the SP to the 60% point of the range.



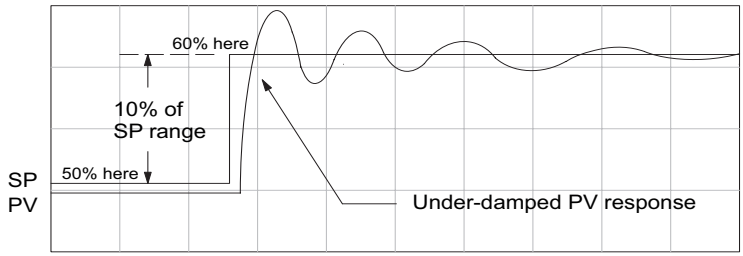
The response may take awhile, but you will see that there isn't any oscillation. This response is not desirable since it takes a long time to correct the error; also, there is a difference between the SP and the PV.



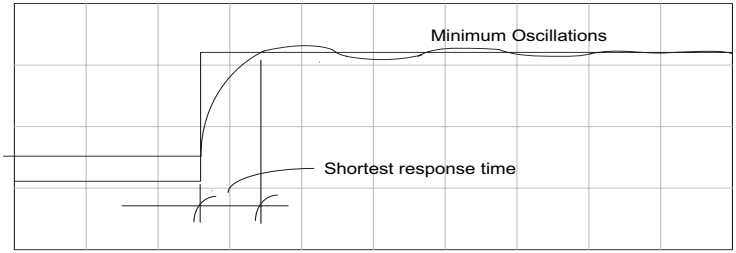
- Increase the Proportional gain, for example to 2.0. The control output will be greater and the response time will be quicker. The trend should resemble the figure below.



- Increase the Proportional gain in small increments, such as 4, 6, 7, etc. until the control output response begins to oscillate. This is the Proportional gain that should be recorded. Now, return the Proportional gain to the stable response, for example, 9.7. The error, SP-PV, should be small, but not at zero.



- Next, add a small amount of Integral gain (reset) in order for the error to reach zero. Begin by using 80 seconds (adjust in minutes if necessary). The error should get smaller.
- Set the Integral gain to a lower value, such as 50 for a different response. If there is no response, continue to decrease the reset value until the response becomes unstable. See the figure below.



For discussion, let us say that a reset value of 35 made the control output unstable. Return the reset value to the stable value, such as 38. Be careful with this adjustment since the oscillation can destroy the process.

The control output response should be optimal now, without a Derivative gain. The recorded values are:

Proportional gain = 9.7 and
Integral gain = 38 seconds.

Note that the error has been minimized.

The foregone method is the most common method used to tune a PID loop. Derivative gain is almost never used in a temperature control loop. This method can also be used for other control loops, but other parameters may need to be added for a stable control output.

Test your loop for a high PV of 80% and again for a low PV of 20%, and correct the values if necessary. Small adjustments of the parameters can make the control output more precise or more unstable. It is sometimes acceptable to have a small overshoot to make the control output react quicker.

The derivative gain can be helpful for those control loops which are not controlling temperature. For these loops, try adding a value of 0.5 for the derivative gain and see if this improves the control output. If there is little or no response, increase the derivative by increments of 0.5 until there is an improvement to the output trend. Recall that the derivative gain reacts with a rate of change of the error.

Alternative Manual Tuning Procedures by Others

The following tuning procedures have been extracted from various publications about PID process control. These procedures are for comparison to the procedure in this manual.

Tuning PID Controllers

Two-Mode Simple Method – for P-I controllers

1. Turn off reset and set the gain to a small value (0.5 - 1.0).
2. Increase gain until cycling starts, then decrease gain slightly.
3. Make setpoint changes to observe offset (error).
4. Increase reset to eliminate offset (error).
5. Repeat steps 2 through 4 until you obtain the largest gain and reset consistent with the criteria of the control desired, i.e., offset, overshoot, stability.

Zeigler-Nichols Method– "Quarter amplitude decay"

1. Turn off reset and rate; set the proportional gain to a fairly large value.
2. Make a small setpoint change and observe how the controlled variable cycles.
3. Adjust the gain until the cycle is self-sustaining, and of constant amplitude; this value is the ultimate gain (G_u).
4. Measure the period of cycling in minutes. This is the ultimate period (P_u).
5. Calculate the controller adjustments as follows:

   P only: $G = G_u/2$

   P & I : $G = G_u/2.2$

   $Ti = 1.2/P_u$ (repeats/minute)

   P-I-D: $G = G_u/1.6$

   $T_i = 2.0/P_u$ (repeats/minute)

   $T_d = P_u/8.0$ (minutes)

Pessen Method

1. Follow the procedure described above (Zeigler-Nichols) to determine the ultimate gain and ultimate period.
2. Apply the formulas below.

   For no overshoot during startup:

   $G = G_u/5.0$

   $T_i = 3/P_u$ (repeats/minute)

   $T_d = P_u/2$ (minutes)

   For some overshoot, but better response to disturbances:

   $G = G_u/3$

   $T_i = 3/P_u$ (repeats/minute)

   $T_d = P_u/3$ (minutes)

Auto Tuning Procedure

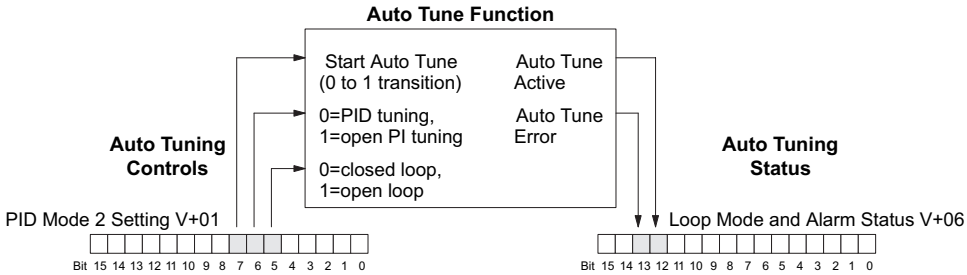
The auto tuning feature for the DL05 loop controller will only run once each time it is enabled in the PID table. Therefore, auto tuning does not run continuously during operation (this would be *adaptive* control). Whenever there is a substantial change in loop dynamics, such as mass of process, size of actuator, etc., the tuning process will need to be repeated in order to derive new gains required for optimal control.

WARNING: Only authorized personnel fully familiar with all aspects of the process should make changes that affect the loop tuning constants. Using the loop auto tuning procedures will affect the process, including inducing large changes in the control output value. Make sure you thoroughly consider the impact of any changes to minimize the risk of injury to personnel or damage to equipment. The auto tune in the DL05 is not intended to be used as a replacement for your process knowledge.

Once the physical loop components are connected to the PLC, auto tuning can be initiated within *Direct*SOFT (see the *Direct*SOFT Programming Software Manual), and it can be used to establish initial PID parameter values. Auto tuning is the best "guess" the CPU can do after some trial tests.

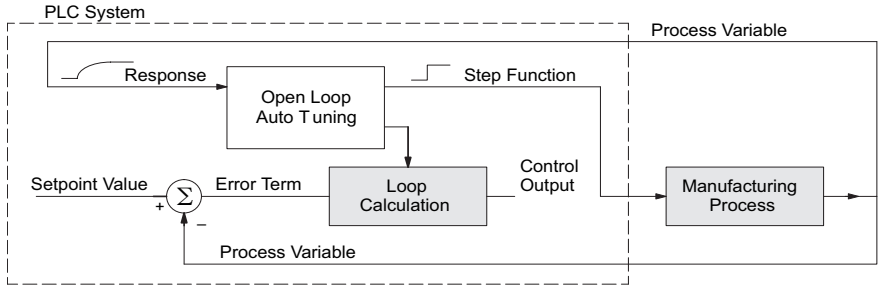
The loop controller offers both closed-loop and open-loop methods. The following sections describe how to use the auto tuning feature, and what occurs in open and closed-loop auto tuning.

The controls for the auto tuning function uses three bits in the PID Mode 2 word V+01, as shown below. *Direct*SOFT programming software will manipulate these bits automatically when you use the auto tune feature within *Direct*SOFT. Or, you may have your ladder logic access these bits directly for allowing control from another source such as a dedicated operator interface. The individual control bits allow you to start the auto tune procedure, select PID or PI tuning and select closed-loop or open-loop tuning. If you select PI tuning, the auto tune procedure leaves the derivative gain at 0. The Loop Mode and Alarm Status word V+06 reports the auto tune status as shown. Bit 12 will be on (1) during the auto tune cycle, automatically returning to off (0) when done.



Open-Loop Auto Tuning

During an open-loop auto tuning cycle, the loop controller operates as shown in the diagram below. Before starting this procedure, place the loop in Manual Mode and ensure the PV and control output values are in the middle of their ranges (away from the end points).

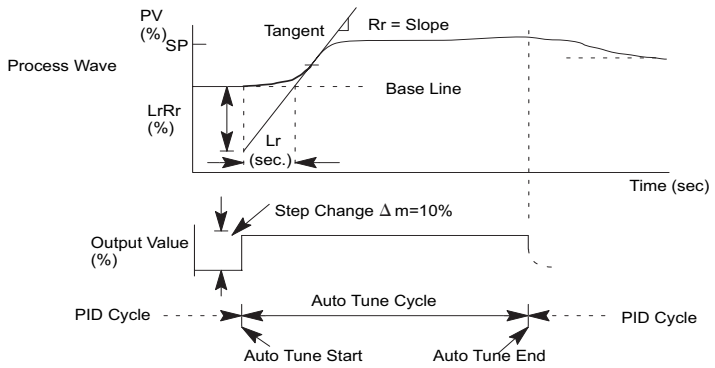


***NOTE**: In theory, the SP value does not matter in this case, because the loop is not closed. However, the requirement of the firmware is that the SP value must be more than 5% of the PV range from the actual PV before starting the auto tune cycle (for the DL05, 12 bit PV should be 205 counts or more below the SP for forward-acting loops, or 205 counts or more above the SP for reverse-acting loops).*

When auto tuning, the loop controller induces a step change on the output and simply observes the response of the PV. From the PV response, the auto tune function calculates the gains and the sample time. It automatically places the results in the corresponding registers in the loop table.

The following timing diagram shows the events which occur in the open-loop auto tuning cycle. The auto tune function takes control of the control output and induces a 10%-of-span step change. If the PV change which the loop controller observes is less than 2%, then the step change on the output is increased to 20%-of-span.

- When Auto Tune starts, step change output m=10%
- During Auto Tune, the controller output reached the full scale positive limit. Auto Tune stopped and the Auto Tune Error bit in the Alarm word bit turned on.
- When PV change is under 2%, output is changed at 20%. Open Loop Auto Tune Cycle Wave: Step Response Method



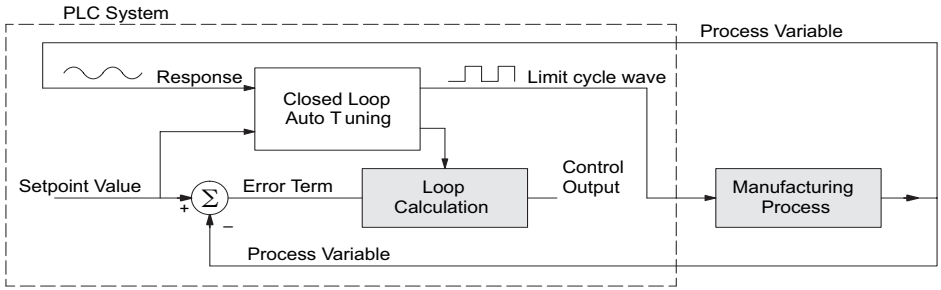
When the loop tuning observations are complete, the loop controller computes R_r (maximum slope in %/sec.) and L_r (dead time in sec). The auto tune function computes the gains according to the Zeigler-Nichols equations, shown below:

PID Tuning	PI Tuning
$P = 1.2*\Delta m/L_rR_r$	$P = 0.9*\Delta m/L_rR_r$
$I = 2.0* L_r$	$I = 3.33* L_r$
$D = 0.5* L_r$	D = 0
Sample Rate = $0.056* L_r$	Sample Rate = $0.12*L_r$
Δm = Output step change (10% = 0.1, 20% = 0.2)	

We highly recommend using *Direct*SOFT programming software for the auto tuning interface. The duration of each auto tuning cycle will depend on the mass of the process. A slowly-changing PV will result in a longer auto tune cycle time. When the auto tuning is complete, the proportional, integral, and derivative gain values are automatically updated in loop table locations V+10, V+11, and V+12 respectively. The sample time in V+07 is also updated automatically. You can test the validity of the values the auto tuning procedure yields by measuring the closed-loop response of the PV to a step change in the output. The instructions on how to do this are in the section on the manual tuning procedure (located prior to this auto tuning section).

Closed-Loop Auto Tuning

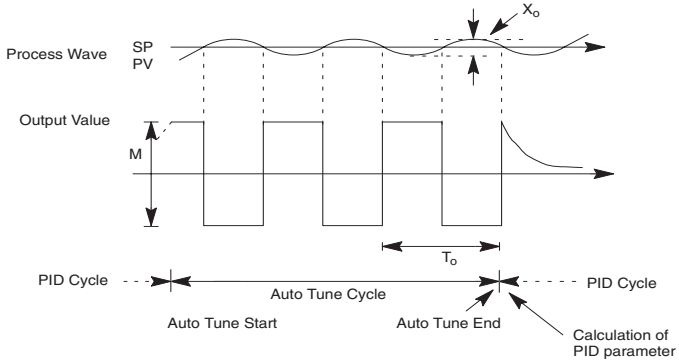
During a closed-loop auto tuning cycle the loop controller operates as shown in the diagram below.



When auto tuning, the loop controller imposes a square wave on the output. Each transition of the output occurs when the PV value crosses over/under the SP value. Therefore, the frequency of the limit cycle is roughly proportional to the mass of the process. From the PV response, the auto tune function calculates the gains and the sample time. It automatically places the results in the corresponding registers in the loop table.

The following timing diagram shows the events which occur in the closed-loop auto tuning cycle. The auto tune function examines the direction of the offset of the PV from the SP. The auto tune function then takes control of the control output and induces a full-span step change in the opposite direction. Each time the sign of the error (SP – PV) changes, the output changes full-span in the opposite direction. This proceeds through three full cycles.

Closed Loop Auto Tune Cycle Wave: Limit Cycle Method



- Mmax = Output Value upper limit setting.
- Mmin = Output Value lower limit setting.

This example is direct–acting.

When set to reverse–acting, the output will be inverted. When the loop tuning observations are complete, the loop controller computes To (bump period) and Xo (amplitude of the PV). Then it uses these values to compute Kpc (sensitive limit) and Tpc (period limit). From these values, the loop controller auto tune function computes the PID gains and the sample rate according to the Zeigler-Nichols equations shown below:

Kpc = 4M / (π *X0) Tpc = 0	
M = Amplitude of output	
PID Tuning	**PI Tuning**
P = 0.45*Kpc	P = 0.30*Kpc
I = 0.60*Tpc	I = 1.00*Tpc
D = 0.10*Tpc	D = 0
Sample Rate = 0.014*Tpc	Sample Rate = 0.03*Tpc

Auto tuning error

If the auto tune error bit (bit 13 of loop Mode/Alarm status word V+06) is on, please verify the PV and SP values are at least 5% of full scale difference, as required by the auto tune function.

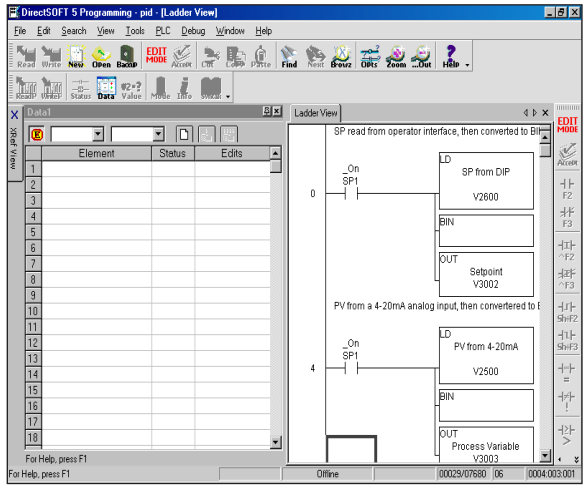
***NOTE**: If your PV fluctuates rapidly, you probably need to use the built-in analog filter (see page 8–53) or create a filter in ladder logic (see example on page 8–54).*

Use DirectSOFT Data View with PID View

The **Data View** window is a very useful tool which can be used to help tune your PID loop. You can compare the variables in the **PID View** with the actual values in the V-memory location with Data View.

Open a New Data View Window

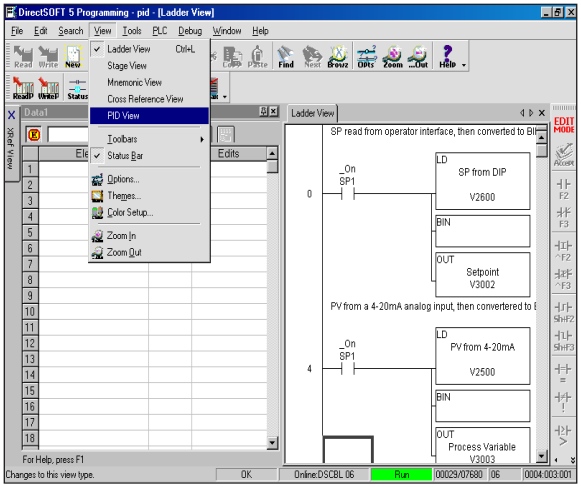
A new Data View window can be opened in any one of three ways; the menu bar **Debug > Data View > New**, the keyboard shortcut **Ctrl + Shift + F3** or the **Data** button on the Status toolbar. By default, the Data View window is assigned ***Data1*** as the default name. This name can be changed for the current view using the Options dialog. The following diagram is an example of a newly opened Data View. The window will open next to the Ladder View by default.



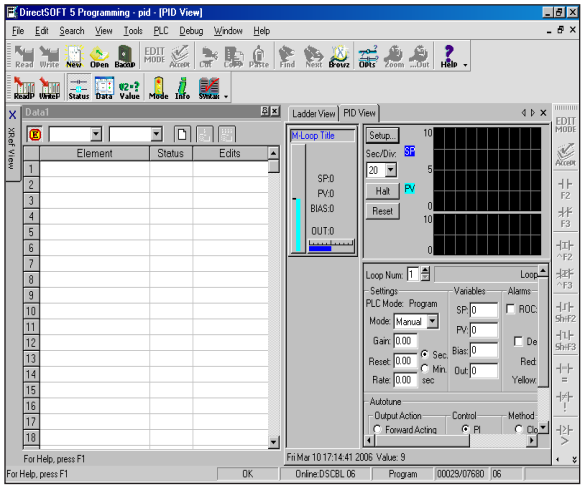
The Data View window can be used just as it is shown above for troubleshooting your PID logic, and it can be most useful when tuning the PID loop.

Open PID View

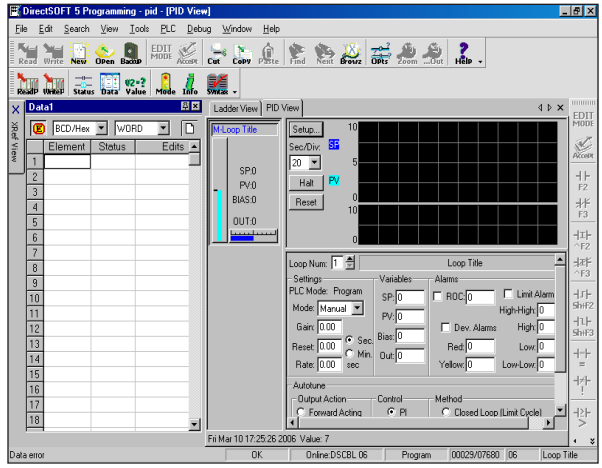
The PID View can only be opened after a loop has been setup in your ladder program and the programming computer is connected to the PLC (online). PID View is opened by selecting it from the View submenu on the Menu bar, **View > PID View**. The PID View can also be opened by clicking on the PID View button from the PLC Setup toolbar if it is in view



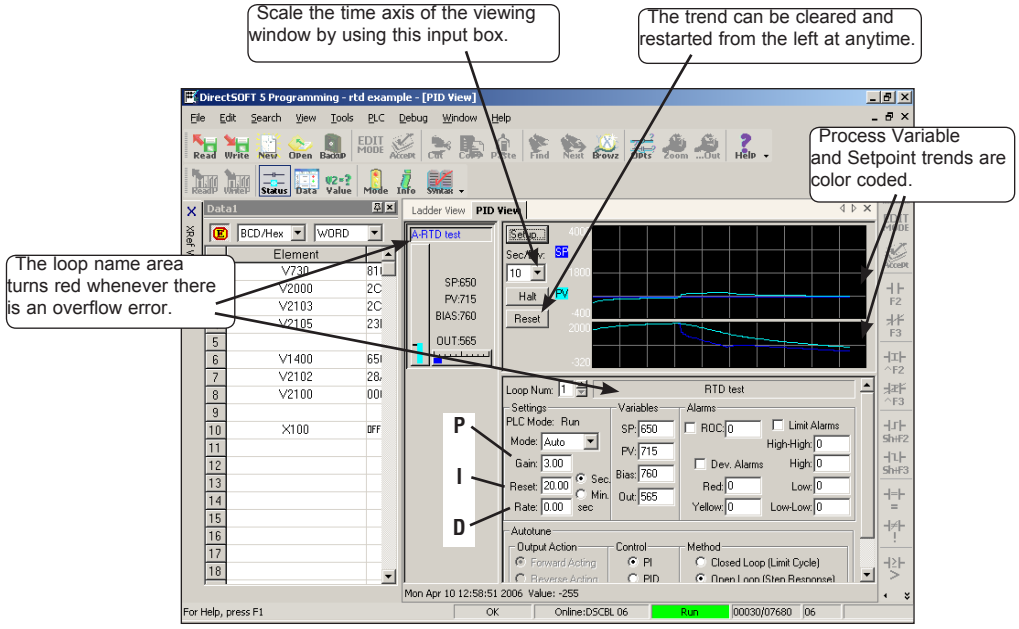
The PID View will open and appear over the Ladder View which can be brought into view by clicking on its tab. When using the Data View and the PID View together, each view can be sized for better use as shown on the facing page.



The two views are now ready to be used to tune your loop. You will be able to see where the PID values have been set and see the process that it is controlling.



The diagram below illustrates how to use the views to see the current SP, PV and Output values, along with the other PID addresses. Refer to the Loop Table Definitions for details of each word in the table. This is also good data type reference for each word in the table.



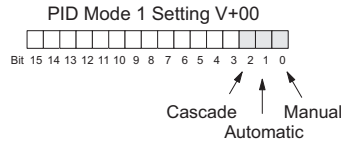
With both windows positioned in this manner, you are able to see where the PID values have been set and see the process that it is controlling. In the diagram below, you can see the current SP, PV and Output values, along with the other PID addresses. Refer to the Loop Table Word Definitions for details for each word in the table. This is also a good data type reference for each word in the table.

Using Other PID Features

It's a good idea to understand the special features of the DL05 and how to use them. You may want to incorporate some of these features for your PID.

How to Change Loop Modes

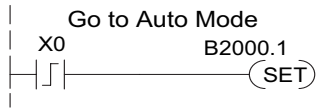
The first three bits of the PID Mode 1 word (V+00) request the operating mode of the corresponding loop. Note: These bits are mode change *requests*, not commands (certain conditions can prohibit a particular mode change – see next page).



The normal state of these mode request bits is "000". To request a mode change, you must SET the corresponding bit to a "1", using a one-shot. The PID loop controller automatically resets the bits back to "000" after it reads the mode change request. Methods of requesting mode changes are:

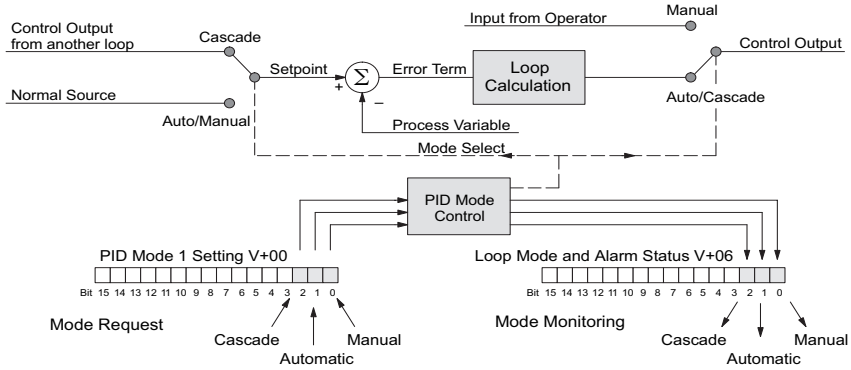
- ***Direct***SOFT's **PID View** – this is the easiest method. Use the drop-down menu, or click on one of the radio buttons, and the appropriate bit will get set.
- **Ladder program**– ladder logic can request any loop mode when the PLC is in Run Mode. This will be necessary after application startup if mode changes are part of the application.

Use the rung shown to the right to SET the mode bit (do not use an OUT coil). On a 0–1 transition of X0, the rung sets the Auto bit equal to 1. The loop controller resets it.



- **Operator panel** – interface the operator's panel to ladder logic using standard methods, then use the logic to the right to set the mode bit.

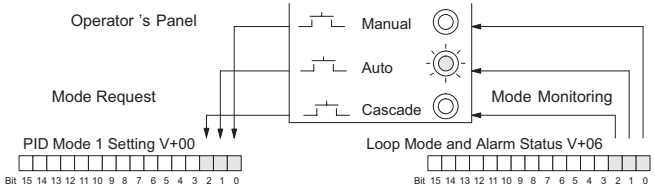
Since mode changes can only be *requested*, the PID loop controller will decide when to permit mode changes and provide the loop mode status. It reports the current mode on bits 0, 1, and 2 of the Loop Mode/Alarm Status word, location V+06 in the loop table. The parallel request/monitoring functions are shown in the figure below. The figure also shows the two possible mode-dependent SP sources, and the two possible Control Output sources.



Operator Panel Control of PID Modes

Since the modes Manual, Auto and Cascade are the most fundamental and important PID loop controls, you may want to "hard-wire" mode control switches to an operator's panel. Most applications will need only Manual and Auto selections (Cascade is used in special applications). Remember that mode controls are really *mode request bits*, and the actual loop mode is indicated elsewhere.

The following figure shows an operator's panel using momentary push-buttons to request PID mode changes. The panel's mode indicators do not connect to the switches, but interface to the corresponding data locations.



PLC Modes Effect on Loop Modes

If you have selected the option for the loops to follow the PLC mode, the PLC modes (Program, Run) interact with the loops as a group. The following summarizes this interaction:

- When the PLC is in Program Mode, all loops are placed in Manual Mode and no loop calculations occur. However, note that output modules (including analog outputs) turn off in PLC Program Mode. So, actual manual control is not possible when the PLC is in Program Mode.
- The only time the CPU will allow a loop mode change is during PLC Run Mode operation. As such, the CPU records the modes of all 4 loops as the desired mode of operation. If power failure and restoration occurs during PLC Run Mode, the CPU returns all loops to their prior mode (which could be Manual, Auto, or Cascade).
- On a Program-to-Run mode transition, the CPU forces each loop to return to its prior mode recorded during the last PLC Run Mode.
- You can add and configure new loops only when the PLC is in Program Mode. New loops automatically begin in Manual Mode.

Loop Mode Override

In normal conditions, the mode of a loop is determined by the request to V+00, bits 0, 1, and 2. However, some conditions exist which will prevent a requested mode change from occurring:

- A loop that is not set independent of PLC mode cannot change modes when the PLC is in Program mode.
- A major loop of a cascaded pair of loops cannot go from Manual to Auto until its minor loop is in Cascade mode.

In other situations, the PID loop controller will automatically change the mode of the loop to ensure safe operation:

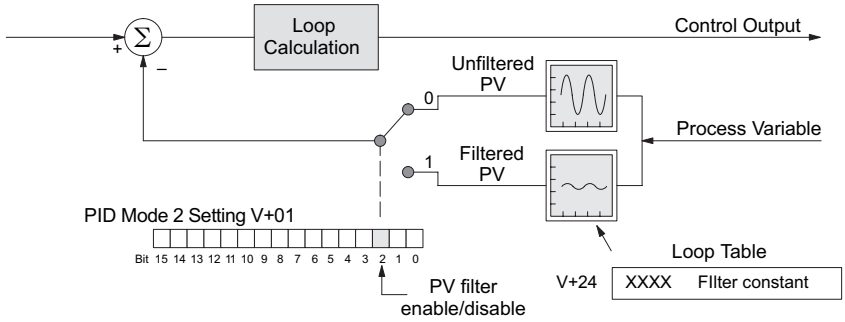
- A loop which develops an error condition automatically goes to Manual.
- If the minor loop of a cascaded pair of loops leaves Cascade Mode for any reason, its major loop automatically goes to Manual Mode.

PV Analog Filter

A noisy PV signal can make tuning difficult and can cause the control output to be more extreme than necessary, as the output tries to respond to the peaks and valleys of the PV. There are two equivalent methods of filtering the PV input to make the loop more stable. The first method is accomplished using the DL05's built-in filter. The second method achieves a similar result using ladder logic.

The DL05 Built-in Analog Filter

The DL05 provides a selectable first-order low-pass PV input filter. **We only recommend the use of a filter during auto tuning or PID control if there is noise on the input signal.** You may disable the filter after auto tuning is complete, or continue to use it if the PV input signal is noisy.



Bit 2 of PID Mode Setting 2 provides the enable/disable control for the low-pass PV filter (0=disable, 1=enable). The roll-off frequency of the single-pole low-pass filter is controlled by using register V+24 in the loop parameter table, the filter constant. The data format of the filter constant value is BCD, with an implied decimal point 00X.X, as follows:

- The filter constant has a valid range of 000.1 to 001.0.
- *Direct*SOFT converts values above the valid range to 001.0 and values below this range to 000.1
- Values close to 001.0 result in higher roll-off frequencies, while values closer to 000.1 result in lower roll-off frequencies.

We highly recommend using *Direct*SOFT programming software for the auto tuning interface. The duration of each auto tuning cycle will depend on the mass of your process. A slowly-changing PV will result in a longer auto tune cycle time.

When the auto tuning is complete, the proportional and integral gain values are automatically updated in loop table locations V+10 and V+11 respectively. The derivative is calculated if you auto tune for PID and updated in loop table location V+12. The sample time in V+07 is also updated automatically. You can test the validity of the values the auto tuning procedure yields by measuring the closed-loop response of the PV to a step change in the output. The instructions on how to do this are in the section on the manual tuning procedure.

The algorithm which the built-in filter follows is:

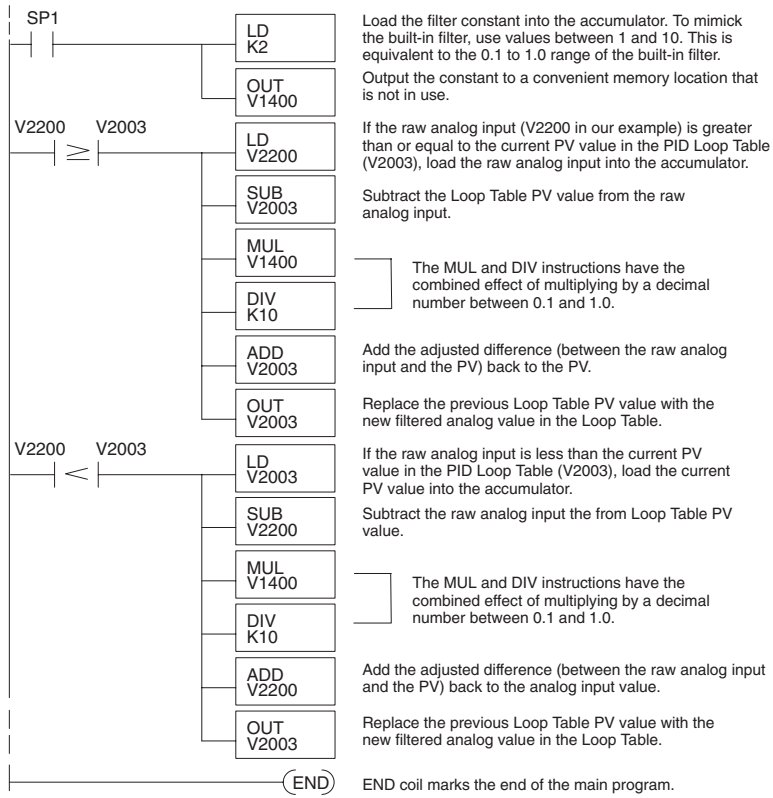
$$yi = k\,(xi - yi{-}1) + yi{-}1$$

yi is the current output of the filter
xi is the current input to the filter
yi–1 is the previous output of the filter
k is the PV Analog Input Filter Factor

Creating an Analog Filter in Ladder Logic

A similar algorithm can be built in your ladder program. Your analog inputs can be filtered effectively using either method. The following programming example describes the ladder logic you will need. Be sure to change the example memory locations to those that fit your application.

Filtering can induce a 1 part in 1000 error in your output because of "rounding." If your process cannot tolerate a 1 part in 1000 error, do not use filtering. Because of the rounding error, you should not use zero or full scale as alarm points. Additionally, the smaller the filter constant the greater the smoothing effect, but the slower the response time. Be sure a slower response is acceptable in controlling your process.

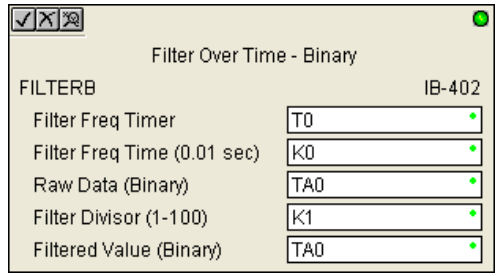


Use the DirectSOFT Filter Intelligent Box Instruction

For those who are using ***Direct***SOFT programming software, you have the opportunity to use Intelligent Box (IBox) instruction IB-402, Filter Over Time in Binary (decimal). This IBox will perform a first-order filter on the Raw Data on a defined time interval. The equation is,

New = Old + [(Raw - Old)/FDC] where

- New =New Filtered Value
- Old = Old Filtered Value
- FDC = Filter Divisor Constant
- Raw = Raw Data



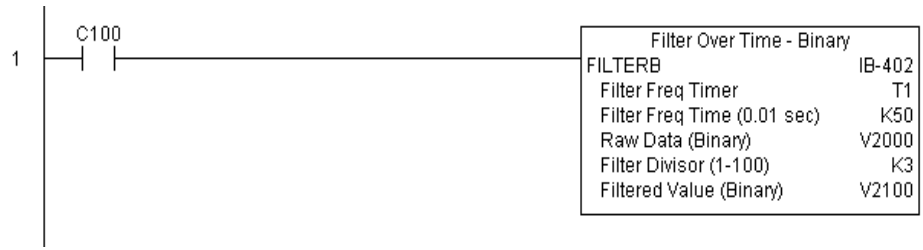
The Filter Divisor Constant is an integer in the range K1 to K100, such that if it equaled K1, then no filtering is performed.

The rate at which the calculation is performed is specified by time in hundredths of a second (0.01 seconds) as the Filter Freq Time parameter. Note that this Timer instruction is embedded in the IBox and must NOT be used any other place in your program. Power flow controls whether the calculation is enabled. If it is disabled, the Filter Value is not updated. On the first scan from Program to Run mode, the Filter Value is initialized to 0 to give the calculation a consistent starting point.

Since the following binary filter example does not write directly to the PID PV location, the BCD filter could be used with BCD values and then converted to BIN.

FilterB Example

Following is an example of how the FilterB IBox is used in a ladder program. The instruction is used to filter a binary value that is in V2000. Timer (T1) is set to 0.5 seconds, the rate at which the filter calculation will be performed. The filter constant is set to 3.0. A larger value will increase the smoothing effect of the filter. A value of 1 results with no filtering. The filtered value will be placed in V2100.

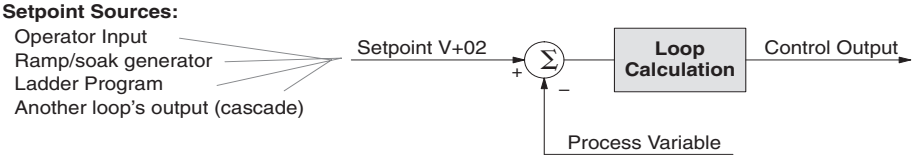


See Chapter 5, for more detailed information.

Ramp/Soak Generator

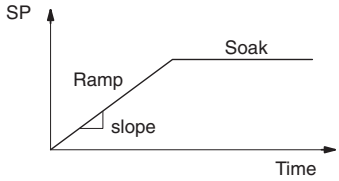
Introduction

Our discussion of basic loop operation noted the setpoint for a loop will be generated in various ways, depending on the loop operating mode and programming preferences. In the figure below, the ramp/soak generator is one of the ways the SP may be generated. It is the responsibility of your ladder program to ensure only one source attempts to write the SP value at V+02 at any particular time.



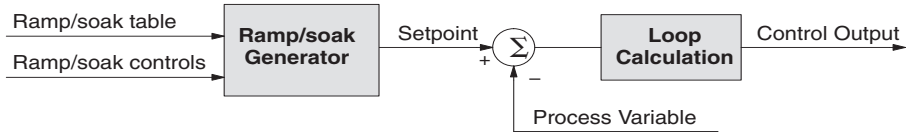
If the SP for your process rarely changes or can tolerate step changes, you probably will not need to use the ramp/soak generator. However, some processes require precisely-controlled SP value changes. *The ramp/soak generator can greatly reduce the amount of programming required for these applications.*

The terms "ramp" and "soak" have special meanings in the process control industry, and refer to desired setpoint (SP) values in temperature control applications. In the figure to the right, the setpoint increases during the ramp segment. It remains steady at one value during the soak segment.



Complex SP profiles can be generated by specifying a series of ramp/soak segments. The ramp segments are specified in SP units per second. The soak time is also programmable in minutes.

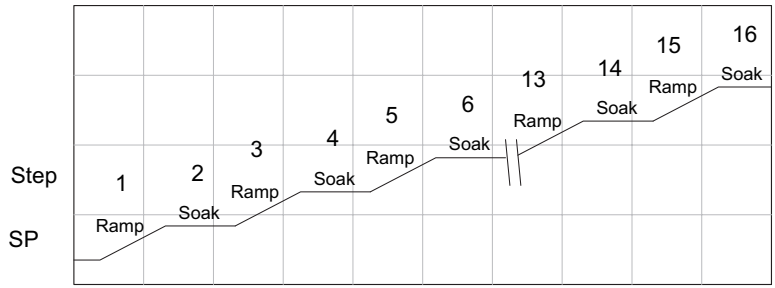
It is instructive to view the ramp/soak generator as a dedicated function to generate SP values, as shown below. It has two categories of inputs which determine the SP values generated. The ramp/soak table must be programmed in advance, containing the values that will define the ramp/soak profile. The loop reads from the table during each PID calculation as necessary. The ramp/soak controls are bits in a special loop table word that control the real-time start/stop functionality of the ramp/soak generator. The ladder program can monitor the status of the ramp soak profile (current ramp/segment number).



Now that we have described the general ramp/soak generator operation, we list its specific features:

- Each loop has its own ramp/soak generator (use is optional).
- You may specify up to eight ramp/soak steps (16 segments).
- The ramp soak generator can run anytime the PLC is in Run mode. Its operation is independent of the loop mode (Manual or Auto).
- Ramp/soak real-time controls include Start, Hold, Resume, and Jog.
- Ramp/soak monitoring includes Profile Complete, Soak Deviation (SP minus PV), and current ramp/soak step number.

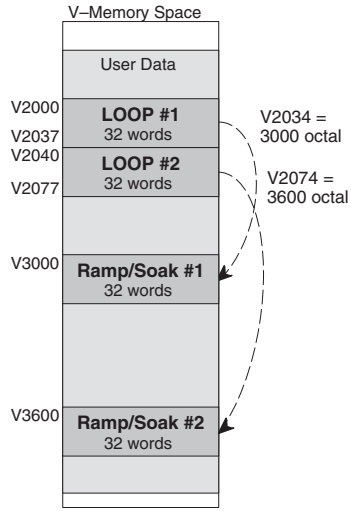
The following figure shows a SP profile consisting of ramp/soak segment pairs. The segments are individually numbered as steps from 1 to 16. The slope of each of the ramp may be either increasing or decreasing. The ramp/soak generator automatically knows whether to increase or decrease the SP based on the relative values of a ramp's end points. These values come from the ramp/soak table.



Ramp/Soak Table

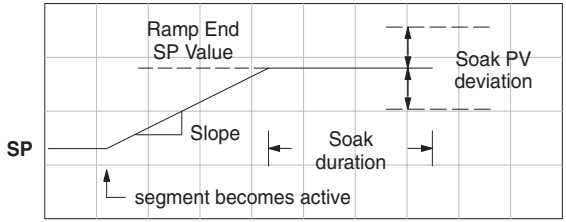
The parameters which define the ramp/soak profile for a loop are in a ramp/soak table. Each loop may have its own ramp/soak table, but it is optional. Recall the Loop Parameter table consists a 32-word block of memory for each loop, and together they occupy one contiguous memory area. However, the ramp/soak table for a loop is individually located, because it is optional for each loop. An address pointer in location V+34 in the loop table specifies the starting location of the ramp/soak table.

In the example to the right, the loop parameter tables for Loop #1 and #2 occupy contiguous 32-word blocks as shown. Each has a pointer to its ramp/soak table, independently located elsewhere in user V-memory. Of course, you may locate all the tables in one group, as long as they do not overlap.



The parameters in the ramp/soak table must be user-defined. The most convenient way is to use *Direct*SOFT programming software, which features a special editor for this table. Four parameters are required to define a ramp and soak segment pair, as pictured below.

- **Ramp End Value** – specifies the destination SP value for the end of the ramp. Use the same data format for this number as you use for the SP. It may be above or below the beginning SP value, so the slope could be up or down (we don't have to know the starting SP value for ramp #1).
- **Ramp Slope** – specifies the SP increase in counts (units) per second. It is a BCD number from 00.00 to 99.99 (uses implied decimal point).
- **Soak Duration** – specifies the time for the soak segment in minutes, ranging from 000.1 to 999.9 minutes in BCD (implied decimal point).
- **Soak PV Deviation** – (optional) specifies an allowable PV deviation above and below the SP value during the soak period. A PV deviation alarm status bit is generated by the ramp/soak generator.



Ramp/Soak Table

V+00	XXXX	Ramp End SP Value
V+01	XXXX	Ramp Slope
V+02	XXXX	Soak Duration
V+03	XXXX	Soak PV Deviation

The ramp segment becomes active when the previous soak segment ends. If the ramp is the first segment, it becomes active when the ramp/soak generator is started, and automatically assumes the present SP as the starting SP.

Offset	Step	Description	Offset	Step	Description
+ 00	1	Ramp End SP Value	+ 20	9	Ramp End SP Value
+ 01	1	Ramp Slope	+ 21	9	Ramp Slope
+ 02	2	Soak Duration	+ 22	10	Soak Duration
+ 03	2	Soak PV Deviation	+ 23	10	Soak PV Deviation
+ 04	3	Ramp End SP Value	+ 24	11	Ramp End SP Value
+ 05	3	Ramp Slope	+ 25	11	Ramp Slope
+ 06	4	Soak Duration	+ 26	12	Soak Duration
+ 07	4	Soak PV Deviation	+ 27	12	Soak PV Deviation
+ 10	5	Ramp End SP Value	+ 30	13	Ramp End SP Value
+ 11	5	Ramp Slope	+ 31	13	Ramp Slope
+ 12	6	Soak Duration	+ 32	14	Soak Duration
+ 13	6	Soak PV Deviation	+ 33	14	Soak PV Deviation
+ 14	7	Ramp End SP Value	+ 34	15	Ramp End SP Value
+ 15	7	Ramp Slope	+ 35	15	Ramp Slope
+ 16	8	Soak Duration	+ 36	16	Soak Duration
+ 17	8	Soak PV Deviation	+ 37	16	Soak PV Deviation

Many applications do not require all 16 R/S steps. Use all zeros in the table for unused steps. The R/S generator ends the profile when it finds ramp slope = 0.

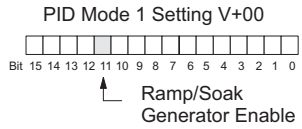
Ramp/Soak Table Flags

The individual bit definitions of the Ramp/Soak Table Flag (Addr+33) word is listed in the following table.

Bit	Ramp/Soak Flag Bit Description	Read/Write	Bit=0	Bit=1
0	Start Ramp / Soak Profile	Write	–	01 Start
1	Hold Ramp / Soak Profile	Write	–	01 Hold
2	Resume Ramp / soak Profile	Write	–	01 Resume
3	Jog Ramp / Soak Profile	Write	–	01 Jog
4	Ramp / Soak Profile Complete	Read	–	Complete
5	PV Input Ramp / Soak Deviation	Read	Off	On
6	Ramp / Soak Profile in Hold	Read	Off	On
7	Reserved	Read	Off	On
8–15	Current Step in R/S Profile	Read	decode as byte (hex)	

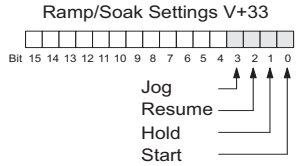
Ramp/Soak Generator Enable

The main enable control to permit ramp/soak generation of the SP value is accomplished with bit 11 in the PID Mode 1 Setting V+00 word, as shown to the right. The other ramp/soak controls in V+33 shown in the table above will not operate unless this bit=1 during the entire ramp/soak process.



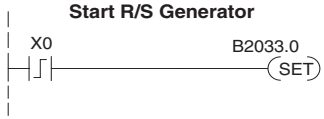
Ramp/Soak Controls

The four main controls for the ramp/soak generator are in bits 0 to 3 of the ramp/soak settings word in the loop parameter table. ***Direct***SOFT programming software controls these bits directly from the ramp/soak settings dialog. However, you must use ladder logic to control these bits during program execution. We recommend using the bit-of-word instructions.



Ladder logic must set a control bit to a "1" to command the corresponding function. When the loop controller reads the ramp/soak value, it automatically turns off the bit for you. Therefore, a reset of the bit is not required when the CPU is in Run Mode.

The example program rung to the right shows how an external switch X0 can turn on and the PD contact uses the leading edge to set the proper control bit to start the ramp soak profile. This uses the Set bit-of-word instruction.

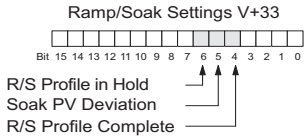


The normal state for the ramp/soak control bits is all zeros. Ladder logic must set only one control bit at a time.

- Start – A 0 to 1 transition will start the ramp soak profile. The CPU must be in Run Mode, and the loop can be in Manual or Auto Mode. If the profile is not interrupted by a Hold or Jog command, it finishes normally.
- Hold – A 0 to 1 transition will stop the ramp/soak profile in its current state, and the SP value will be frozen.
- Resume – A 0 to 1 transition cause the ramp/soak generator to resume operation if it is in the hold state. The SP values will resume from their previous value.
- Jog – A 0 to 1 transition will cause the ramp/soak generator to truncate the current segment (step), and go to the next segment.

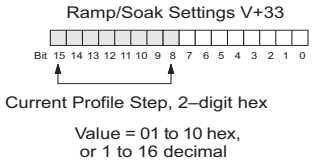
Ramp/Soak Profile Monitoring

You can monitor the Ramp/Soak profile status using other bits in the Ramp/Soak Settings V+33 word, shown to the right.



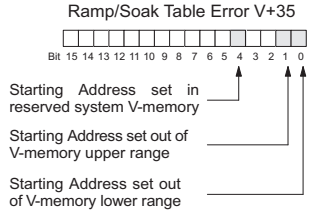
- R/S Profile Complete: =1 when the last programmed step is done.
- Soak PV Deviation: =1 when the error (SP–PV) exceeds the specified deviation in the R/S table.
- R/S Profile in Hold: =1 when the profile was active but is now in hold. Ramp/Soak Settings V+33

The number of the current step is available in the upper 8 bits of the Ramp/Soak Settings V+33 word. The bits represent a 2-digit hex number, ranging from 1 to 10. Ladder logic can monitor these to synchronize other parts of the program with the ramp/soak profile. Load this word to the accumulator and shift right 8 bits, and you have the step number.



Ramp/Soak Programming Errors

The starting address for the ramp/soak table must be a valid location. If the address points outside the range of user V-memory, one of the bits to the right will turn on when the ramp/soak generator is started. We recommend using *Direct*SOFT programming software to configure the ramp/soak table. It automatically range checks the addresses for you.



Testing Your Ramp/Soak Profile

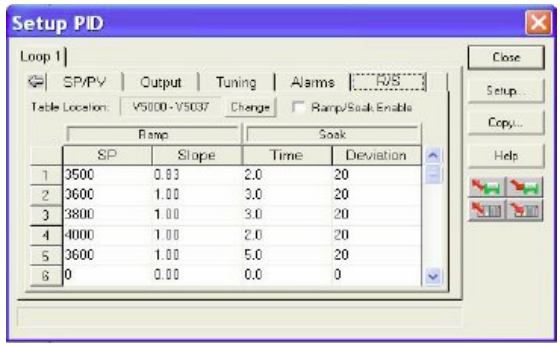
It's a good idea to test your ramp/soak profile before using it to control the process. This is easy to do, because the ramp/soak generator will run even when the loop is in Manual Mode. Using *Direct*SOFT's PID View will be a real time-saver, because it will draw the profile on-screen for you. Be sure to set the trending timebase slow enough to display completed ramp-soak segment pairs in the waveform window.

DirectSOFT Ramp/Soak Example

The following example will step you through the Ramp/Soak setup.

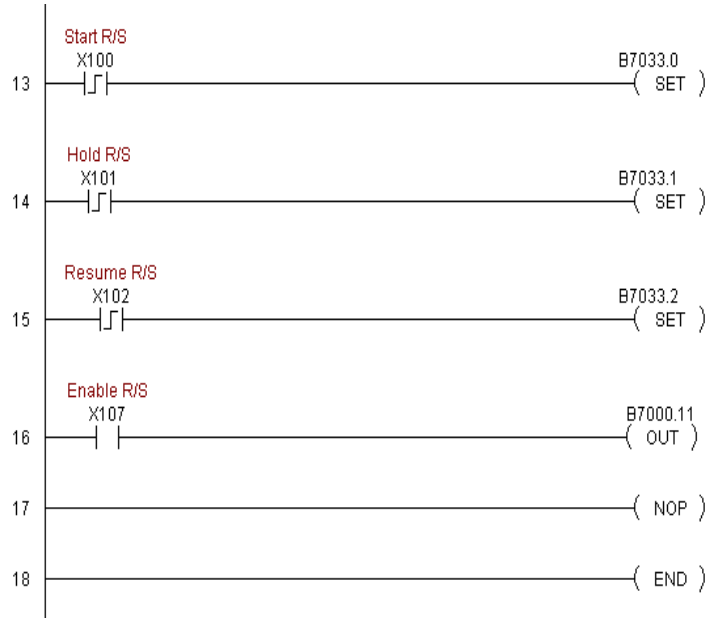
Setup the Profile in PID Setup

The first step is to use Setup PID in *Direct*SOFT programming software to set the profile of your process. Open the Setup PID window and select the R/S tab, and then enter the Ramp and Soak data.



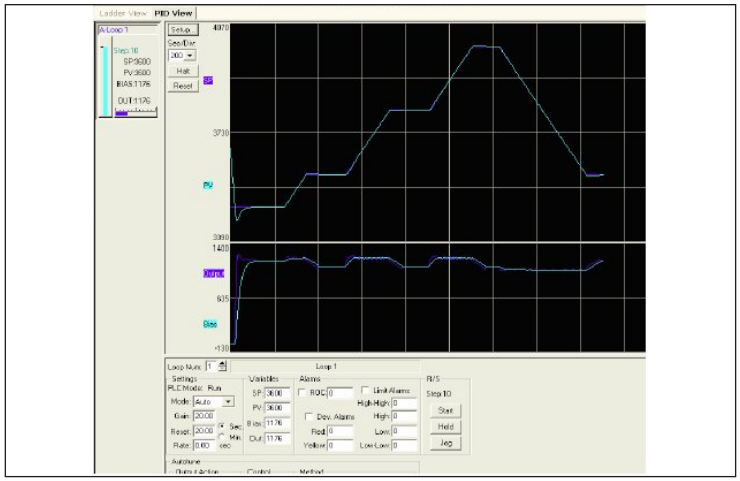
Program the Ramp/Soak Control in Relay Ladder

Refer to the Ramp/Soak Flag bit Description table on page 8-60 when adding the control rungs to your program similar to the ladder rungs below. For the example below, PID parameters begin at V7000. The Ramp/Soak Bit flags are located at V7033.



Test the Profile

After the Ramp/Soak program has been developed in RLL, test the program. Check your profile by using PID View. If there are any changes to be made in the profile, they are made in the PID Setup R/S profile. Make the changes in Program mode then start the Ramp/Soak process again.



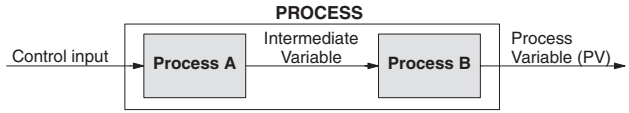
Cascade Control

Introduction

Cascaded loops are an advanced control technique that is superior to individual loop control in certain situations. As the name implies, cascade means that one loop is connected to another loop. In addition to Manual (open loop) and Auto (closed loop) Modes, the DL05 also provides Cascaded Mode.

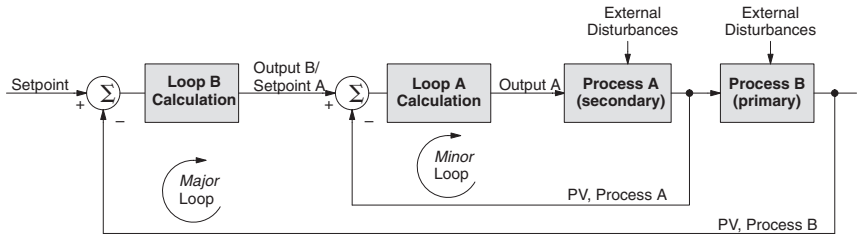
***NOTE**: Cascaded loops are an advanced process control technique. Therefore we recommend their use only for experienced process control engineers.*

When a manufacturing process is complex and contains a lag time from control input to process variable output, even the most perfectly tuned single loop around the process may yield slow and inaccurate control. It may be the actuator operates on one physical property, which eventually affects the process variable, measured by a different physical property. Identifying the intermediate variable allows us to divide the process into two parts as shown in the following figure.



The principle of cascaded loops is simply that we add another process loop to more precisely control the intermediate variable! This separates the source of the control lag into two parts, as well.

The diagram below shows a cascade control system, showing that it is simply one loop nested inside another. The inside loop is called the minor loop, and the outside loop is called the major loop. For overall stability, the minor loop must be the fastest responding loop of the two (try a factor of 10 for a better response time). We do have to add the additional sensor to measure the intermediate variable (PV for process A). Notice the setpoint for the minor loop is automatically generated for us, by using the output of the major loop. Once the cascaded control is programmed and debugged, we only need to deal with the original setpoint and process variable at the system level. The cascaded loops behave as one loop, but with improved performance over the previous single-loop solution.



One of the benefits to cascade control can be seen by examining its response to external disturbances. Remember the minor loop is faster acting than the major loop. Therefore, if a disturbance affects process A in the minor loop, the Loop A PID calculation can correct the resulting error before the major loop sees the effect.

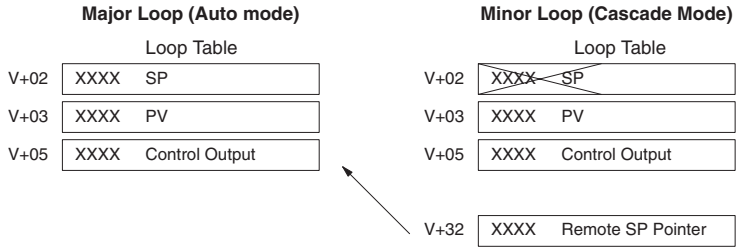
Cascaded Loops in the DL05 CPU

In the use of the term "cascaded loops", we must make an important distinction. Only the minor loop will actually be in the Cascade Mode. In normal operation, the major loop must be in Auto Mode. If you have more than two loops cascaded together, the outer-most (major) loop must be in Auto Mode during normal operation, and all inner loops in Cascade Mode.

***NOTE**: Technically, both major and minor loops are "cascaded" in strict process control terminology. Unfortunately, we are unable to retain this convention when controlling loop modes. Remember that all minor loops will be in Cascade Mode, and only the outer-most (major) loop will be in Auto Mode.*

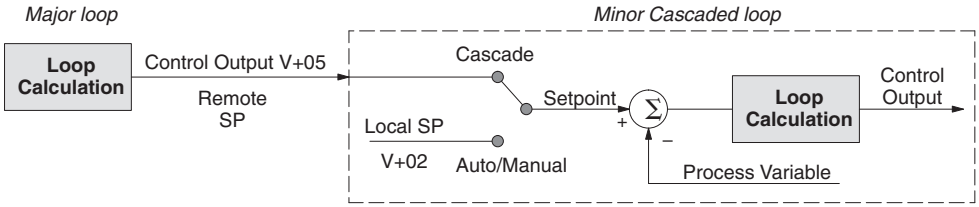
You can cascade together as many loops as necessary on the DL05, and you may have multiple groups of cascaded loops. For proper operation on cascaded loops you must use the same data range (12/15 bit) and unipolar/bipolar settings on the major and minor loop.

To prepare a loop for Cascade Mode operation as a minor loop, you must program its remote Setpoint Pointer in its loop parameter table location V+32, as shown below. The pointer must be the address of the V+05 location (control output) of the major loop. In Cascade Mode, the minor loop will ignore the its local SP register (V+02), and read the major loop's control output as its SP instead.



When using *Direct*SOFT's PID View to watch the SP value of the minor loop, *Direct*SOFT automatically reads the major loop's control output and displays it for the minor loop's SP. The minor loop's normal SP location, V+02, remains unchanged.

Now, we use the loop parameter arrangement above and draw its equivalent loop schematic, shown below.



Remember that a major loop goes to Manual Mode automatically if its minor loop is taken out of Cascade Mode.

Tuning Cascaded Loops

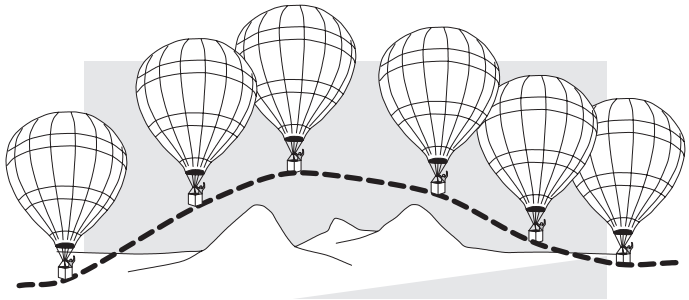
When tuning cascaded loops, you will need to de-couple the cascade relationship and tune the minor loop using one of the loop tuning procedures previously covered. Once this has been done, have the minor loop in cascade mode and auto tune the major loop (see Step 4).

1. If you are not using auto tuning, then find the loop sample rate for the minor loop, using the method discussed earlier in this chapter. Then set the sample rate of the major loop slower than the minor loop by a factor of 10. Use this as a starting point.
2. Tune the minor loop first. Leave the major loop in Manual Mode, and you will need to generate SP changes for the minor loop manually as described in the loop tuning procedure.
3. Verify the minor loop gives a critically-damped response to a 10% SP change while in Auto Mode. Then we are finished tuning the minor loop.
4. In this step, you will need to get the minor loop in Cascade Mode, and then the Major loop in Auto Mode. We will be tuning the major loop with the minor loop treated as a series component its overall process. Therefore, do not go back and tune the minor loop again while tuning the major loop.
5. Tune the major loop, following the standard loop tuning procedure in this section. The response of the major loop PV is actually the overall response of the cascaded loops together.

Time-Proportioning Control

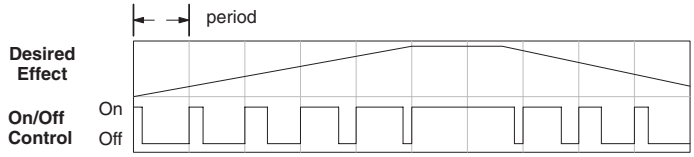
The PID loop controller in the DL05 CPU generates a smooth control output signal across a numerical range. The control output value is suitable to drive an analog output module, which connects to the process. In the process control field, this is called *continuous control*, because the output is on (at some level) continuously.

While continuous control can be smooth and robust, the cost of the loop components (such as actuator, heater amplifiers) can be expensive. A simpler form of control is called *time-proportioning control.* This method uses actuator which are either on or off (no in-between). Loop components for on/off-based control systems are lower cost than their continuous control counterparts.



In this section, we will show you how to convert the control output of a loop to time-proportioning control for the applications that need it. Let's take a moment to review how alternately turning a load on and off can control a process. The diagram below shows a hot-air balloon following a path across some mountains. The desired path is the *setpoint*. The balloon pilot turns the burner on and off alternately, which is his *control output*. The large mass of air in the balloon effectively averages the effect of the burner, converting the bursts of heat into a continuous effect, slowly changing balloon temperature and ultimately the altitude, which is the *process variable*.

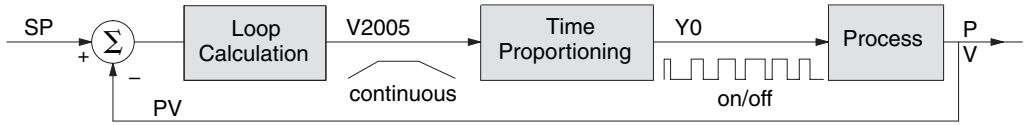
Time-proportioning control approximates continuous control by virtue of its duty-cycle – the ratio of ON time to OFF time. The following figure shows an example of how duty-cycle approximates a continuous level when it is averaged by a large process mass.



If we were to plot the on/off times of the burner in the hot-air balloon, we would probably see a very similar relationship to its effect on balloon temperature and altitude.

On/Off Control Program Example

The following ladder segment provides a time proportioned on/off control output. It converts the continuous output in V2005 to on/off control using the output coil, Y0.

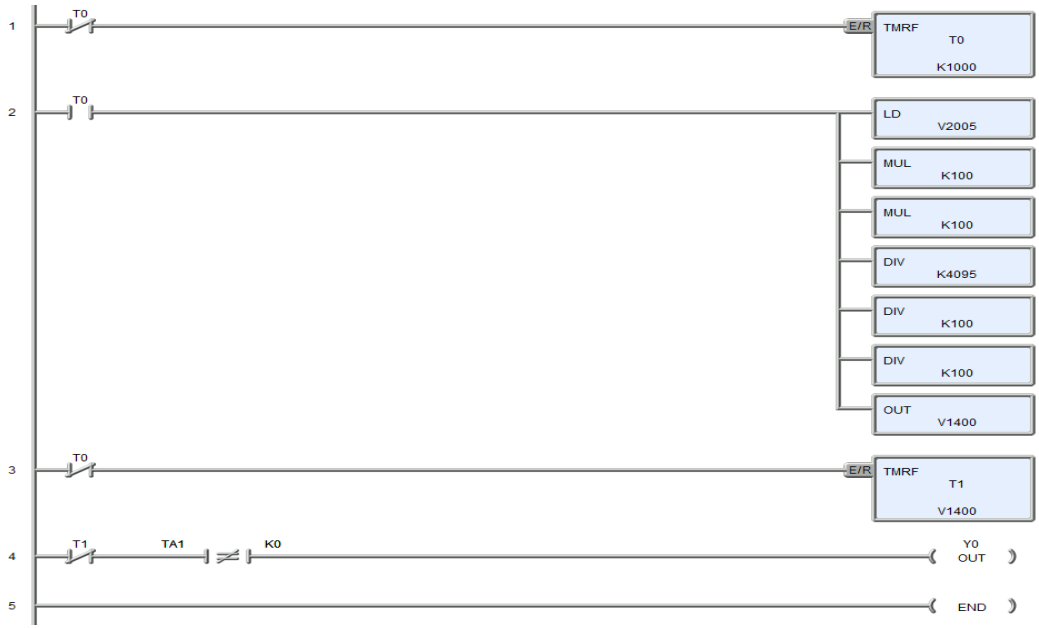


The example program uses two timers to generate On/Off control. It makes the following **assumptions**, which you can alter to fit your application:

- The loop table starts at V2000, so the control output is at V2005.
- The data format of the control output is 12-bit, unipolar (0 – FFF).
- The time base (one full cycle) for the On/Off waveform is 10 seconds. We use a fast timer (0.01 sec/tick), counting to 1000 ticks (10 seconds).
- The On/Off control output is Y0.

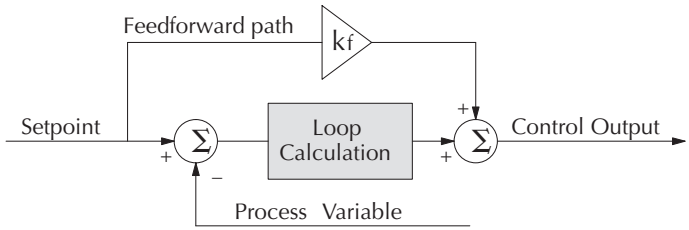
The time proportioning program must match the resolution of the PID output (1 part in 1000) to the resolution of the time base of T0 (also 1 part in 1000).

***NOTE**: Some processes change too fast for time proportioning control. Consider the speed of your process when you choose this control method. Use continuous control for processes that change too fast for time proportioning control. Also, consider using a solid state switch for a longer switch life instead of a relay.*



Feedforward Control

Feedforward control is an enhancement to standard closed-loop control. It is most useful for diminishing the effects of a *quantifiable and predictable* loop disturbance or sudden change in setpoint. Use of this feature is an option available to you on the DL05. However, it's best to implement and tune a loop without feedforward, and adding it only if better loop performance is still needed. The term "feed-forward" refers to the control technique involved, shown in the diagram below. The incoming setpoint value is fed forward around the PID equation, and summed with the output.

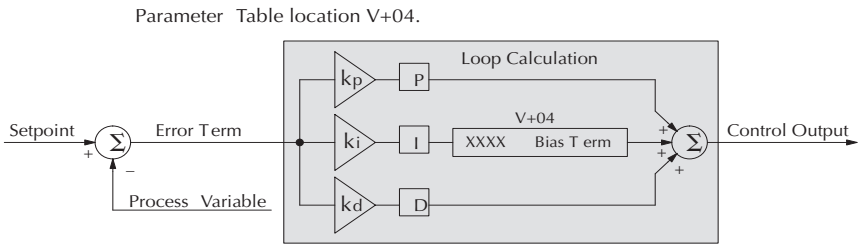


In the previous section on the bias term, we said that "the bias term value establishes a "working region" or operating point for the control output. *When the error fluctuates around its zero point, the output fluctuates around the bias value.*" Now, when there is a change in setpoint, an error is generated and the output must change to a new operating point. This also happens if a disturbance introduces a new offset in the loop. The loop does not really "know its way" to the new operating point... the integrator (bias) must increment/decrement until the error disappears, and then the bias has found the new operating point.

Suppose that we are able to know a sudden setpoint change is about to occur (common in some applications). We can avoid much of the resulting error in the first place, if we can quickly change the output to the new operating point. If we know (from previous testing) what the operating point (bias value) will be after the setpoint change, we can artificially change the output directly (which is feedforward). The benefits from using feedforward are:

- The SP–PV error is reduced during predictable setpoint changes or loop offset disturbances.
- Proper use of feedforward will allow us to reduce the integrator gain. Reducing integrator gain gives us an even more stable control system.

Feedforward is very easy to use in the DL05 loop controller, as shown below. The bias term has been made available to the user in a special read/write location, at PID Parameter Table location V+04.

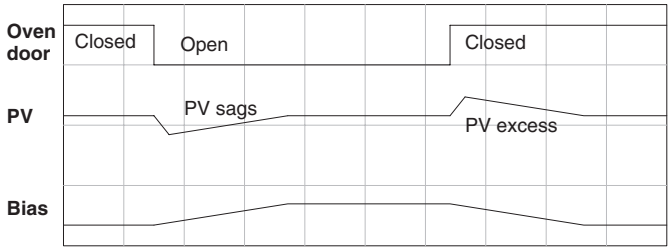


To change the bias (operating point), ladder logic only has to write the desired value to V+04. The PID loop calculation first reads the bias value from V+04 and modifies the value based on the current integrator calculation. Then it writes the result back to location V+04. This arrangement creates a sort of "transparent" bias term. All you have to do to implement feed forward control is write the correct value to the bias term at the right time (see the following example).

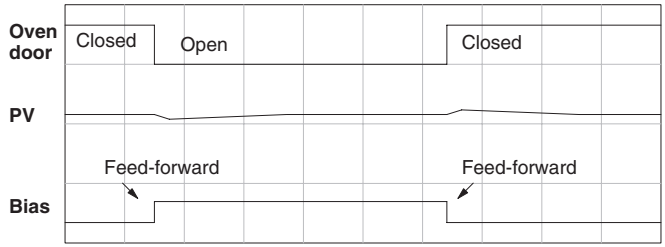
***NOTE**: When writing the bias term, one must be careful to design ladder logic to write the value only once, at the moment when the new bias operating point is to occur. If ladder logic writes the bias value on every scan, the loop's integrator is effectively disabled.*

Feedforward Example

How do we know when to write to the bias term, and what value to write? Suppose we have an oven temperature control loop, and we have already tuned the loop for optimal performance. Refer to the figure below. We notice that when the operator opens the oven door, the temperature sags a bit while the loop bias adjusts to the heat loss. Then when the door closes, the temperature rises above the SP until the loop adjusts again. Feedforward control can help diminish this effect.



First, we record the amount of bias change the loop controller generates when the door opens or closes. Then, we write a ladder program to monitor the position of an oven door limit switch. When the door opens, our ladder program reads the current bias value from V+04, adds the desired change amount, and writes it back to V+04. When the door closes, we duplicate the procedure, but subtracting desired change amount instead. The following figure shows the results.



The step changes in the bias are the result of our two feed-forward writes to the bias term. We can see the PV variations are greatly reduced. The same technique may be applied for changes in setpoint.

PID Example Program

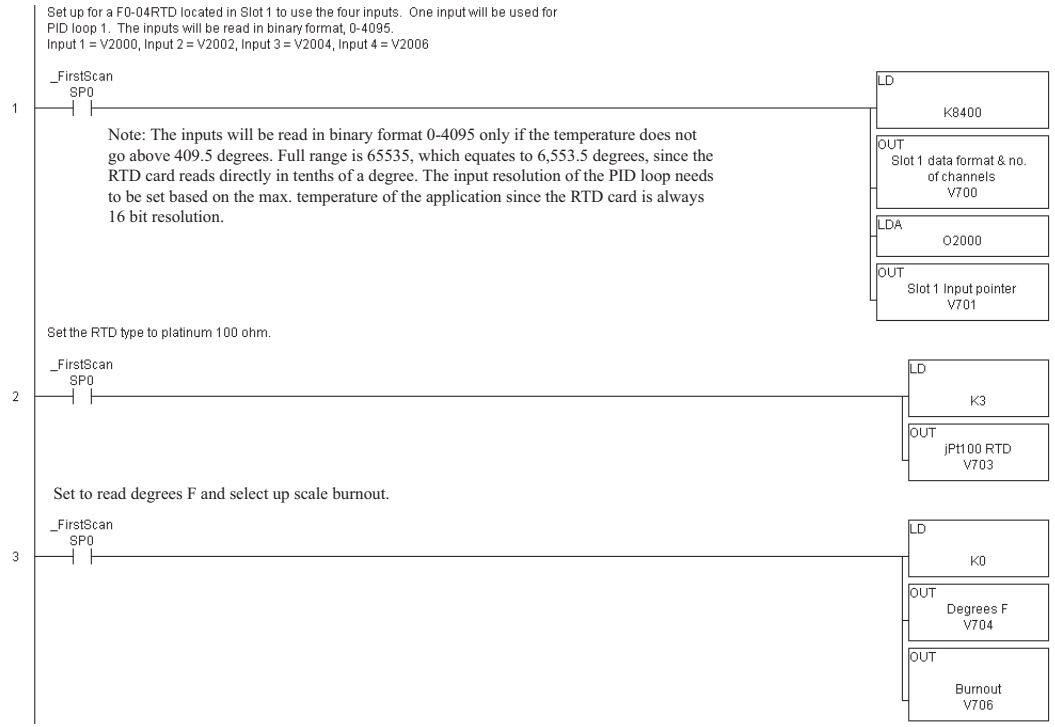
Program Setup for the PID Loop

After the PID loop, or loops, have been setup with *Direct*SOFT, you will need to edit your RLL program to include the rungs needed to setup the analog I/O module to be used by the PID loop(s).

The following example program shows how an RTD module, F0-04RTD, and an analog combination module, F0-2AD2DA-2, is setup. This example assumes that the PID table for loop 1 has a beginning address of V2100.

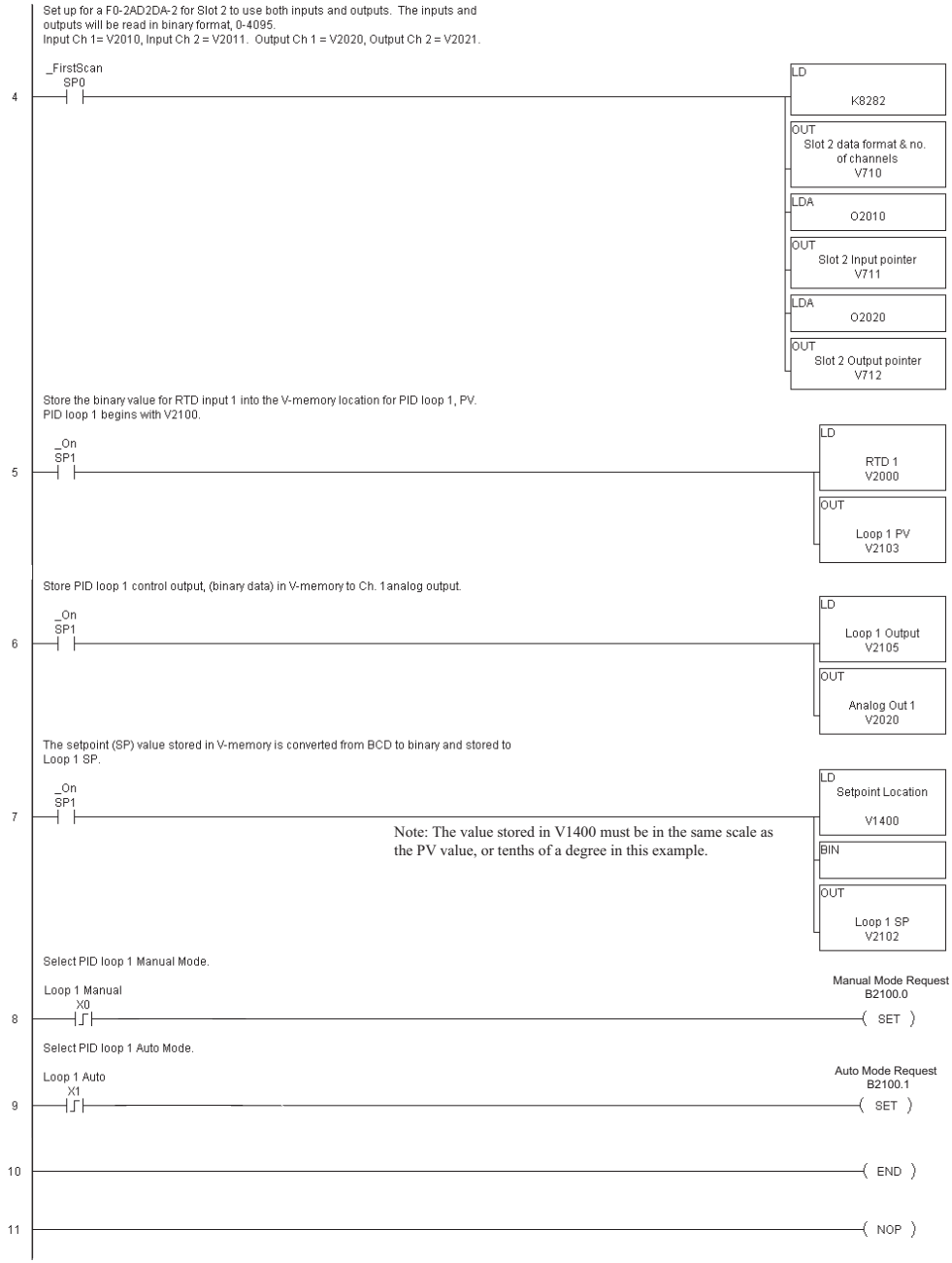
All of the analog I/O modules used with the DL05 are setup in a similar manner. Refer to the DL05/DL06 Options Manual for the setup information for the particular module that you will be using.

***Direct*SOFT**



Program continued on next page

Example Program (continued)



Note that the modules used in this PID example program were set up for binary format. They could have been setup for BCD format. In the latter case, the BCD data would have to be converted to binary format before being stored to the setpoint and process variable, and the control output would have to be converted from binary to BCD before being stored to the analog output.

By following the steps outlined in this chapter, you should be able to setup workable PID control loops. The ***Direct***SOFT Programming Software Manual provided more information for the use of PID View.

For a step-by-step tutorial, go to the Technical Support section located on our website, **www.automationdirect.com**. Once you are at the website, click on **Technical Support Home**. After this page opens, find and select **Guided Tutorials** located under the **Using Your Products** column. An **Animated Tutorial** page will open. Under **Available Tutorials**, find **PID Trainer** and select **View the Powerpoint slide show** and begin viewing the tutorial. The Powerpoint Viewer can be downloaded if your computer does not have Powerpoint installed.

Troubleshooting Tips

Q. The loop will not go into Automatic Mode.

A. Check the following for possible causes:

- A PV alarm exists, or a PV alarm programming error exists.
- The loop is the major loop of a cascaded pair, and the minor loop is not in Cascade Mode.

Q. The Control Output stays at zero constantly when the loop is in Automatic Mode.

A. Check the following for possible causes:

- The Control Output upper limit in loop table location V+31 is zero.
- The loop is driven into saturation, because the error never goes to zero value and changes (algebraic) sign.

Q. The Control Output value is not zero, but it is incorrect.

A. Check the following for possible causes:

- The gain values are entered improperly. Remember, gains are entered in the loop table in BCD, while the SP and PV are in binary. If you are using *Direct*SOFT, PID View will display the SP, PV, Bias and Control output in decimal (BCD), converting it to binary before updating the loop table.

Q. The Ramp/Soak Generator does not operate when I activate the Start bit.

A. Check the following for possible causes:

- The Ramp/Soak enable bit is off. Check the status of bit 11 of loop parameter table location V+00. It must be set =1.
- The hold bit or other bits in the Ramp/Soak control are on.
- The beginning SP value and the first ramp ending SP value are the same, so first ramp segment has no slope and consequently has no duration. The ramp/soak generator moves quickly to the soak segment, giving the illusion the first ramp is not working.
- The loop is in Cascade Mode, and is trying to get the SP remotely.
- The SP upper limit value in the loop table location V+27 is too low.
- Check your ladder program to verify it is not writing to the SP location (V+02 in the loop table). A quick way to do this is to temporarily place an end coil at the beginning of your program, then go to PLC Run Mode, and manually start the ramp/soak generator.

Q. The PV value in the table is constant, even though the analog module receives the PV signal.

A. Your ladder program must read the analog value from the module successfully and write it into the loop table V+03 location. Verify the analog module is generating the value, and the ladder is working.

Q. The Derivative gain doesn't seem to have any affect on the output.

A. The derivative limit is probably enabled (see section on derivative gain limiting).

Q. The loop Setpoint appears to be changing by itself.

A. Check the following for possible causes:

- The Ramp/Soak generator is enabled, and is generating setpoints.
- If this symptom occurs on loop Manual-to-Auto Mode changes, the loop automatically sets the SP=PV if set to bumless transfer mode 1 (bumpless transfer feature).
- Check your ladder program to verify it is not writing to the SP location (V+02 in the loop table). A quick way to do this is to temporarily place an end coil at the beginning of your program, then go to PLC Run Mode.

Q. The SP and PV values I enter with DirectSOFT work okay, but these values do not work properly when the ladder program writes the data.

A. The PID View in *Direct*SOFT lets you enter SP, PV, and Bias values in decimal, and displays them in decimal for your convenience. For example, when the data format is 12 bit unipolar, the values range from 0 to 4095. However, the loop table actually requires these to be in hex, so *Direct*SOFT converts them for you. Your ladder program must convert constant values from their BCD format (when entered as Kxxxx) to binary with the BIN instruction or you must enter them in the constant field (Kxxxx) as the hex equivalent of the decimal value.

Q. The loop seems unstable and impossible to tune, no matter what gains I use.

A. Check the following for possible causes:

- The loop sample time is set too long. Refer to the section near the front of this chapter on selecting the loop update time.
- The gains are too high. Start out by reducing the derivative gain to zero. Then reduce the integral gain by increasing the integral time value and the proportional gain if necessary.
- There is too much transfer lag in your process. This means the PV reacts sluggishly to control output changes. There may be too much "distance" between actuator and PV sensor, or the actuator may be weak in its ability to transfer energy into the process.
- There may be a process disturbance that is over-powering the loop. Make sure the PV is relatively steady when the SP is not changing.

Glossary of PID Loop Terminology

Automatic Mode – An operational mode of a loop, in which it makes PID calculations and updates the loop's control output.

Bias Freeze – A method of preserving the bias value (operating point) for a control output, by inhibiting the integrator when the output goes out-of-range. The benefit is a faster loop recovery.

Bias Term – In the position form of the PID equation, it is the sum of the integrator and the initial control output value.

Bumpless Transfer – A method of changing the operation mode of a loop while avoiding the usual sudden change in control output level. This consequence is avoided by artificially making the SP and PV equal, or the bias term and control output equal at the moment of mode change.

Cascaded Loops – A cascaded loop receives its setpoint from the output of another loop. Cascaded loops have a major/minor relationship, and work together to ultimately control one PV.

Cascade Mode – An operational mode of a loop, in which it receives its SP from another loop's output.

Continuous Control – Control of a process done by delivering a smooth (analog) signal as the control output.

Control Output – The numerical result of a PID equation which is sent by the loop with the intention of nulling out the current error.

Derivative Gain – A constant that determines the magnitude of the PID derivative term in response to the current error.

Direct-Acting Loop – A loop in which the PV increases in response to a control output increase. In other words, the process has a positive gain.

Error – The difference in value between the SP and PV, Error = SP – PV.

Error Deadband – An optional feature which makes the loop insensitive to errors when they are small. You can specify the size of the deadband.

Error Squared – An optional feature which multiplies the error by itself, but retains the original algebraic sign. It reduces the effect of small errors, while magnifying the effect of large errors.

Feedforward – A method of optimizing the control response of a loop when a change in setpoint or disturbance offset is known and has a quantifiable effect on the bias term.

Integral Gain – A constant that determines the magnitude of the PID integral term in response to the current error.

Major Loop – In cascade control, it is the loop that generates a setpoint for the cascaded loop.

Manual Mode – An operational mode of a loop, it which the PID calculations are stopped. The operator must manually control the loop by writing to the control output value directly.

Minor Loop – In cascade control, the minor loop is the subordinate loop that receives its SP from the major loop.

On/Off Control – A simple method of controlling a process, through on/off application of energy into the system. The mass of the process averages the on/off effect for a relatively smooth PV. A simple ladder program can convert the DL05's continuous loop output to on/off control.

PID Loop – A mathematical method of closed-loop control involving the sum of three terms based on proportional, integral, and derivative error values. The three terms have independent gain constants, allowing one to optimize (tune) the loop for a particular physical system.

Position Algorithm – The control output is calculated so it responds to the displacement (position) of the PV from the SP (error term).

Process – A manufacturing procedure which adds value to raw materials. Process control particularly refers to inducing chemical changes to the material in process.

Process Variable (PV) – A quantitative measurement of a physical property of the material in process, which affects final product quality and is important to monitor and control.

Proportional Gain – A constant that determines the magnitude of the PID proportional term in response to the current error.

PV Absolute Alarm – A programmable alarm that compares the PV value to alarm threshold values.

PV Deviation Alarm – A programmable alarm that compares the difference between the SP and PV values to a deviation threshold value.

Ramp/Soak Profile – A set of SP values called a profile, which is generated in real time upon each loop calculation. The profile consists of a series of ramp and soak segment pairs, greatly simplifying the task of programming the PLC to generate such SP sequences.

Rate – Also called differentiator, the rate term responds to the changes in the error term.

Remote Setpoint – The location where a loop reads its setpoint when it is configured as the minor loop in a cascaded loop topology.

Reset – Also called integrator, the reset term adds each sampled error to the previous, maintaining a running total called the bias.

Reset Windup A condition created when the loop is unable to find equilibrium, and the persistent error causes the integrator (reset) sum to grow excessively (windup). Reset windup causes an extra recovery delay when the original loop fault is remedied.

Reverse-Acting Loop – A loop in which the PV increases in response to a control output decrease. In other words, the process has a negative gain.

Sampling Time – The time between PID calculations. The CPU method of process control is called a sampling controller, because it samples the SP and PV only periodically.

Setpoint (SP) – The desired value for the process variable. The setpoint (SP) is the input command to the loop controller during closed loop operation.

Soak Deviation – The soak deviation is a measure of the difference between the SP and PV during a soak segment of the Ramp/Soak profile, when the Ramp/Soak generator is active.

Step Response – The behavior of the process variable in response to a step change in the SP (in closed loop operation), or a step change in the control output (in open loop operation).

Transfer – To change from one loop operational mode to another (between Manual, Auto, or Cascade). The word "transfer" probably refers to the transfer of control of the control output or the SP, depending on the particular mode change.

Velocity Algorithm – The control output is calculated to represent the rate of change (velocity) for the PV to become equal to the SP.

Bibliography

Fundamentals of Process Control Theory, Second Edition Author: Paul W. Murrill Publisher: Instrument Society of America ISBN 1–55617–297–4	Application Concepts of Process Control Author: Paul W. Murrill Publisher: Instrument Society of America ISBN 1–55617–080–7
PID Controllers: Theory, Design, and Tuning, 2nd Edition Author: K. Astrom and T Hagglund Publisher: Instrument Society of America ISBN 1–55617–516–7	Fundamentals of Temperature, Pressure, and Flow Measurements, Third edition Author: Robert P. Benedict Publisher: John Wiley and Sons ISBN 0–471–89383–8
Process / Industrial Instruments & Controls Handbook, Fourth Edition Author (Editor-in-Chief): Douglas M. Considine Publisher: McGraw-Hill, Inc ISBN 0-07-012445-0	pH Measurement and Control, Second Edition Author: Gregory K. McMillan Publisher: Instrument Society of America ISBN 1–55617–483–7
Programmable Controllers Concepts and Applications, First Edition Authors: C.T. Jones and L.A. Bryan Publisher: International Programmable Controls ISBN 0-915425-00-9	Fundamentals of Programmable Logic Controllers, Sensors, and Communications Author: Jon Stenerson Publisher: Prentice Hall ISBN 0-13-726860-2
Process Control, Third Edition Instrument Engineer's Handbook Author (Editor-in-Chief): Bela G. Liptak Publisher: Chilton ISBN 0–8019–8242–1	Process Measurement and Analysis, Third Edition Instrument Engineer's Handbook Author (Editor-in-Chief): Bela G. Liptak Publisher: Chilton ISBN 0–8019–8197–2