PID Loop Operation

In This Chapter...

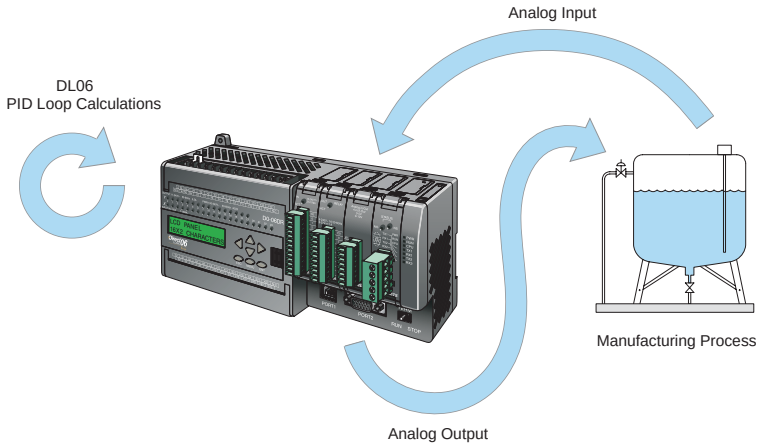
DL06 PID Loop Features

Main Features

The DL06 process loop control offers a sophisticated set of features to address many application needs. The main features are:

- Up to 8 loops, individual programmable sample rates
- Manual/Automatic/Cascaded loop capability available
- Two types of bumpless transfer available
- Full-featured alarms
- Ramp/soak generator with up to 16 segments
- Auto Tuning

The DL06 CPU has process control loop capability in addition to ladder program execution. You can select and configure up to eight loops. All sensor and actuator wiring connects directly to DL06 analog modules. All process variables, gain values, alarm levels, etc., associated with each loop reside in a Loop Variable Table in the CPU. The DL06 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 what you must do to configure and tune the



loops.

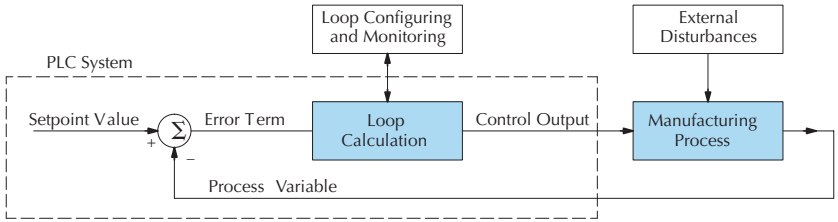
The best tool for configuring loops in the DL06 is the ***Direct***SOFT32 programming software, release 4.0, or later. ***Direct***SOFT32 uses dialog boxes to help you set up the individual loops. After completing the setup, you can use ***Direct***SOFT32's PID Trend View to tune each loop. The configuration and tuning selections you make are stored in the DL06's FLASH memory, which is retentive. The loop parameters also may be saved to disk for recall later.

PID Loop Feature	Specifications
Number of loops	Selectable, 8 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, 0.1 seconds for 5 to 8 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
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
Deadband	Specify 0.1% to 5% alarm deadband on all alarms
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

The Basics of PID Loops

The key parts of a PID control loop are shown in the block diagram below. The path from the PLC to the Manufacturing Process and back to the PLC is the "loop" in "closed loop control."



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 – a measurement of some physical property of the raw materials. Measurements are made using some type of sensor. For example, if the manufacturing process uses an oven, you will most likely want to control temperature. Temperature is a process variable.

Setpoint Value – the theoretically perfect quantity of the process variable, or the desired amount which yields the best product. The machine operator knows this value, and either sets it manually or programs it into the PLC for later automated use.

External Disturbances – the unpredictable sources of error which the control system attempts to cancel by offsetting their effects. For example, if the fuel input is constant an oven will run hotter during warm weather than it does during cold weather. An oven control system must counter-act this effect to maintain a constant oven temperature during any season. Thus, the weather (which is not very predictable), is one source of disturbance to this process.

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.

Loop Calculation – the real-time application of a mathematical algorithm to the error term, generating a control output command appropriate for minimizing the error magnitude. Various control algorithms are available, and the DL06 uses the Proportional-Integral-Derivative (PID) algorithm (more on this later).

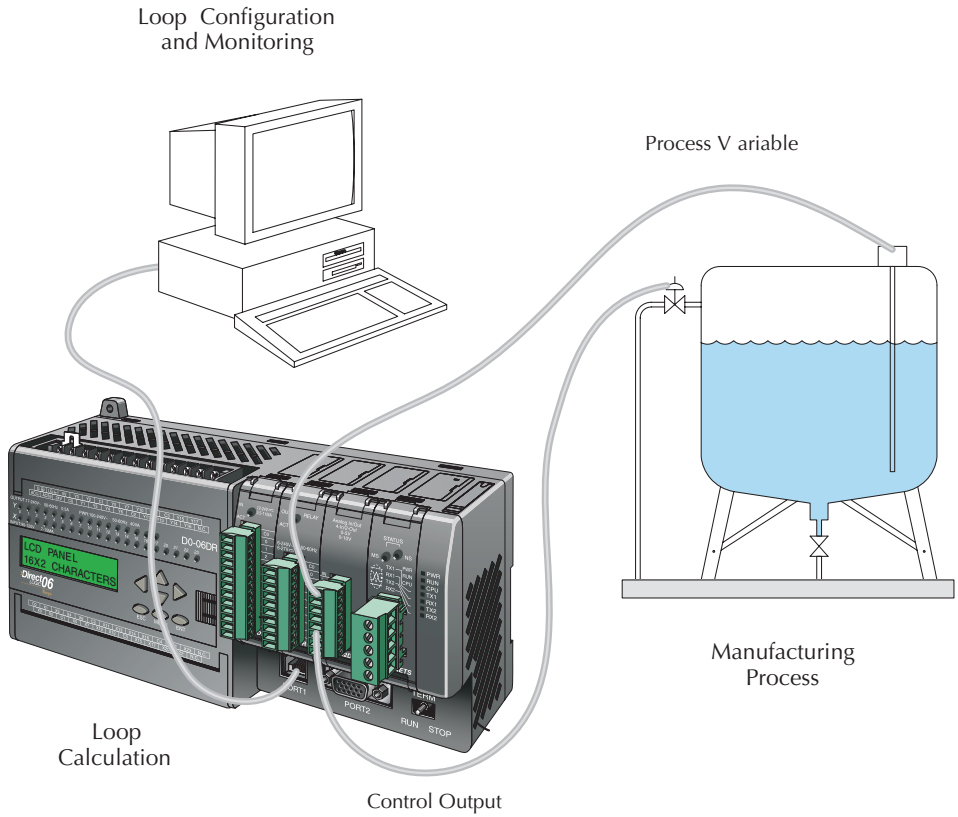
Control Output – the result of the loop calculation, which becomes a command for the process (such as the heater level in an oven).

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).

The diagram below shows each loop element in the form of its real-world physical component. The example manufacturing process involves a liquid in a reactor vessel. A sensor probe measures a process variable which may be pressure, temperature, or another parameter. The sensor signal is amplified through a transducer, and is sent through the wire in analog form to the PLC input module.

The PLC reads the PV from its analog input. The CPU executes the loop calculation, and writes to the analog output. This signal goes to a device in the manufacturing process, such as a heater, valve, pump, etc. Over time, the liquid begins to change enough to be measured on the sensor probe. The process variable changes accordingly. The next loop calculation occurs, and the loop cycle repeats in this manner continuously.



The personal computer shown is used to run *Direct*SOFT32, the PLC programming software for *Direct*LOGIC programmable controllers. *Direct*SOFT32, release 4.0 or later, can program the DL06 PLC (including the PID feature). The software features a forms-based editor to configure loop parameters. It also features a PID loop trending screen which will be helpful during the loop tuning process. Details on how to use that software are in the *Direct*SOFT32 Manual.

Loop Setup Parameters

Loop Table and Number of Loops

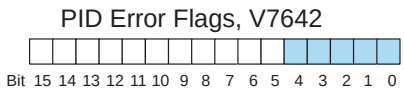
The DL06 PLC gets its PID loop processing instructions only from tables in V-memory. A "PID instruction" type in RLL does not exist for the *Direct*Logic PLCs. Instead, the CPU reads setup parameters from reserved V-memory locations. Shown in the table below, you must program a value in V7640 to point to the main loop table. Then you will need to program V7641 with the number of loops you want the CPU to calculate. V7642 contains error flags which will be set if V7640 or V7641 are programmed improperly.

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

If the number of loops is "0", the loop controller task is turned off during the ladder program scan. The loop controller will allow use of loops in ascending order, beginning with 1. For example, you cannot use loop 1 and 4 while skipping 2 and 3. The loop controller attempts to control the full number of loops specified in V7641.

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.



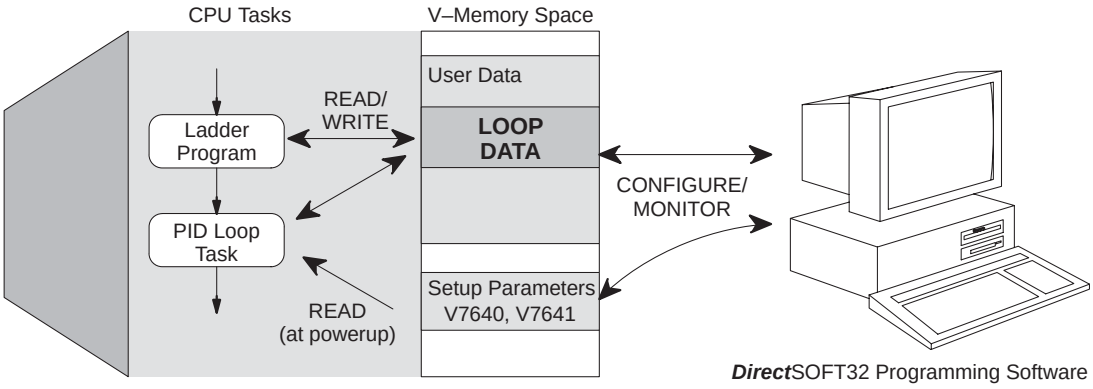
If you use the *Direct*SOFT32 loop setup dialog box, its automatic range checking prohibits possible setup errors. However, the setup parameters may be written using other methods such as RLL, so the error flag register may be helpful in those cases. The following table lists the errors reported in V7642.

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 V7577. Use an address closer to V1200.

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

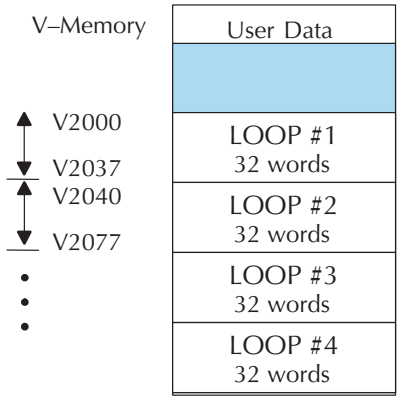
Establishing the Loop Table Size and Location

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 Parameter table contains data for only as many loops as you selected in V7641. Each loop configuration occupies 32 words (0 to 37 octal) in the loop table.

For example, suppose you have an application with 4 loops, and you choose V2000 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.



Loop Table Word Definitions

The parameters associated with each loop are listed in the following table. The address offset is in octal, to help you locate specific parameters in a loop table. For example, if a table begins at V2000, then the location of 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	-
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 auto transfer, base/slot/channel number/pointer	word/hex	
32	Addr + 37	Control output auto transfer, channel number	word/hex	

*Read data only when alarm enable bit transitions 0 to 1
**Read data only on PLC Mode change

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. Additional information about the use of this word is available later in this chapter.

Bit	PID Mode Setting 1 Description	Read/Write	Bit=0	Bit=1
0	Manual Mode Loop Operation request	write	–	0-1 request
1	Automatic Mode Loop Operation request	write	–	0-1 request
2	Cascade Mode Loop Operation request	write	–	0-1 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. Additional information about the use of this word is available later in this chapter.

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

Note 1: 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 2: 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 3: 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. More details are in the PID Mode section and Alarms section.

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. Further details are given in the Ramp / Soak Operation section.

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)	

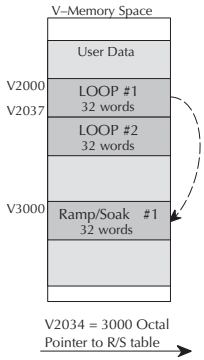
Bits 8–15 must be read as a byte to indicate the current segment number of the Ramp/Soak generator in the profile. This byte will have the values 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, and 10, which represent segments 1 to 16 respectively. If the byte=0, then the Ramp/Soak table is not active.

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*SOFT32 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 this chapter.

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	–	–	–

Loop Sample Rate and Scheduling

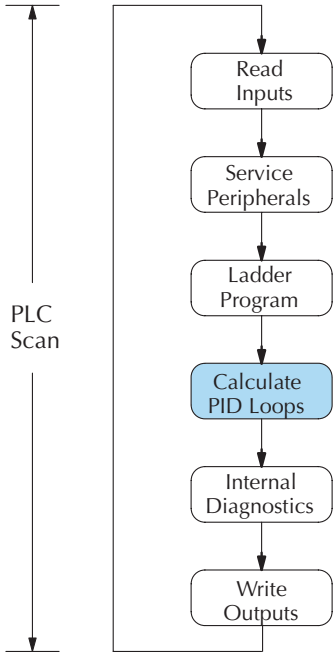
Loop Sample Rates Addr + 07

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.

Note: It is possible to keep the PID loops running even when the ladder is not. This is done by selecting direct access in Addr + 36 and placing a 1 in bit 15 of Addr + 00.

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 DL06 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.

You select the desired sample rate for each loop, and the CPU automatically schedules and executes PID calculations on the appropriate scans.



Choosing the Best Sample Rate

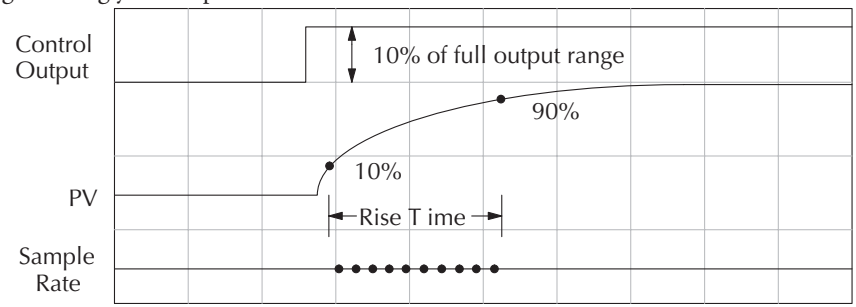
For any particular control loop, there is no single perfect sample rate to use. A good sample rate is a compromise that simultaneously satisfies various guidelines:

- The desired sample rate is proportional to the response time of the PV to a change in control output. Usually, a process with a large mass will have a slow sample rate, but a small mass needs a faster sample rate.
- Faster sample rates provide a smoother control output and accurate PV performance, but use more CPU processing time. Sample rates much faster than necessary serve only to waste CPU processing power.
- Slower sample rates provide a rougher control output and less accurate PV performance, but use less CPU processing time.
- A sample rate which is too slow will cause system instability, particularly when a change in the setpoint or a disturbance occurs.

As a starting point, determine a sample rate for your loop which will be fast enough to avoid control instability (which is extremely important). Follow the procedure on the next page to find a starting sample rate:

Determining a suitable sample rate (Addr+07):

1. Operate the process open-loop (the loop does not even need to be configured yet). Place the CPU in run mode (and the loop in Manual mode, if you have already configured it). Manually set the control output value so the PV is stable and in the middle of a safe range.
2. Try to choose a time when the process will have negligible external disturbances. Then induce a sudden 10% step change in the control value.
3. Record the rise or fall time of the PV (time between 10% to 90% points).
4. Divide the recorded rise or fall time by 10. This is the initial sample rate you can use to begin tuning your loop.

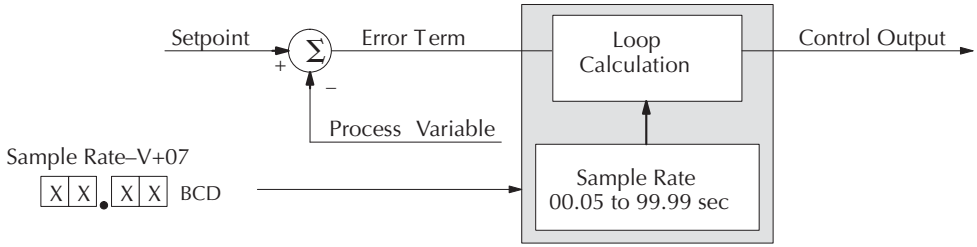


In the figure above, suppose the measured rise time response of the PV was 25 seconds. The suggested sample rate from this measurement will be 2.5 seconds. For illustration, the sample rate time line shows ten samples within the rise time period. These show the frequency of PID calculations as the PV changes values. Of course, the sample rate and PID calculations are continuous during operation.

NOTE: *An excessively fast sample rate will diminish the available resolution in the PV Rate-of-Change Alarm, because the alarm rate value is specified in terms of PV change per sample period. For example, a 50 mS sample rate means the smallest PV rate-of-change we can detect is 20 PV counts (least significant bit counts) per second, or 1200 LSB counts per minute.*

Programming the Sample Rate

The Loop Parameter table for each loop has a data location for the sample rate. Referring to the figure below, location addr+07 contains a BCD number from 00.05 to 99.99 (with an implied decimal point). This represents 50 mS to 99.99 seconds. This number may be programmed using ***Direct***SOFT32's PID Setup screen, or any other method of writing to V-memory. It must be programmed before the loop will operate properly.



PID Loop Effect on CPU Scan Time

Since PID loop calculations are a task within the CPU scan activities, the use of PID loops will increase the *average* scan time. The amount of scan time increase is proportional to the number of loops used and the sample rate of each loop.

The execution time for a single loop calculation depends on the number of options selected, such as alarms, error squared, etc. The chart to the right gives the range of times you can expect.

PID Calculation Time	
Minimum	150 µS
Typical	250 µS
Maximum	350 µS

To calculate scan time increase, we also must know (or estimate) the scan time of the ladder (without loops). A fast scan time will increase by a smaller percentage than a slow scan time will, when adding the same PID loop calculation load in each case. The formula for average scan time calculation is:

$$\text{Avg. Scan Time with PID loop} = \left[\frac{\text{Scan time without loop}}{\text{Sample rate of loop}} \times \text{PID calculation time} \right] + \text{Scan time without loop}$$

$$\text{Average Scan time with PID loop} = \left[\frac{50\ \text{mS}}{3\ \text{sec.}} \times 250\ \mu\text{S} \right] + 50\ \text{mS} = 50.004\ \text{mS}$$

As the calculation shows, the addition of only one loop with a slow sample rate has a very small effect on scan time. Next, expand the equation above to show the effect of adding any number of loops:

$$\text{Avg. Scan Time with PID loops} = \left[\sum_{n=1}^{n=L} \frac{\text{Scan time without loop}}{\text{Sample rate of nth loop}} \times \text{PID calculation time} \right] + \text{Scan time without loops}$$

In the new equation above, you calculate the summation term (inside the brackets) for each loop from 1 to L (last loop), and add the right-most term "scan time without loops" only once at the end. Suppose you have a DL06 PLC controlling four loops. The table below shows the data and summation term values for each loop.

Loop Number	Description	Sample Rate	Summation Term
1	Steam Flow, Inlet valve	0.25 sec	50 µS
2	Water bath temperature	30 sec	0.42 µS
3	Dye level, main tank	10 sec	1.25 µS
4	Steam Pressure, Autoclave	1.5 sec	8.3 µS

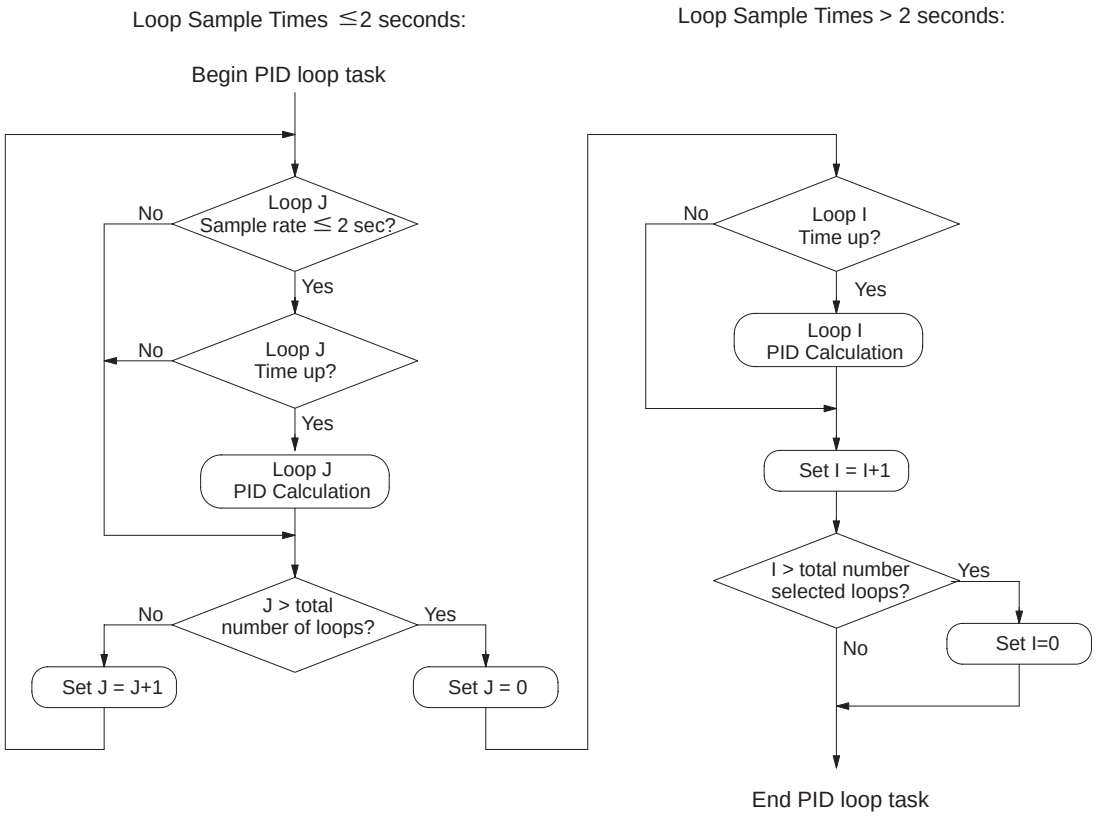
Now adding the summation terms, plus the original scan time value, we have:

$$\text{Avg. Scan Time with PID loops} = \left[50\ \mu\text{S} + 0.42\ \mu\text{S} + 1.25\ \mu\text{S} + 8.3\mu\text{S} \right] + 50\ \text{mS} = 50.06\ \text{mS}$$

The DL06 CPU only does PID calculation on a particular scan for the loop(s) which have sample time periods that are due for an update (calculation). The built-in loop scheduler applies the following rules:

- Loops with sample rates less than or equal to 2 seconds are processed at the rate of as many loops per scan as is required to maintain each loop's sample rate. Specifying loops with fast sample rates will increase the PLC scan time. *So, use this capability only if you need it!*
- Loops with sample rates more than 2 seconds are processed at the rate of one or fewer loops per scan, at the minimum rate required to maintain each loop's sample rate.

The implementation of loop calculation scheduling is shown in the flow chart below. This is a more detailed look at the contents of the "Calculate PID Loops" task in the CPU scan activities flow chart. The pointers "I" and "J" correspond to the slow (more than 2 sec) and fast (less than or equal to 2 sec) loops, respectively. The flow chart allows the J pointer to increment from loop 1 to the last loop, if there are any fast loops specified. The I pointer increments only once per scan, and then only when the next slow loop is due for an update. In this way, both I and J pointers cycle from 1 to the highest loop number used, except at different rates. Their combined activity keeps all loops properly updated.



Ten Steps to Successful Process Control

Modern electronic controllers such as the DL06 CPU 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 make 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

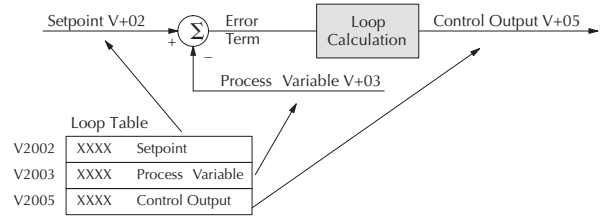
Assuming the control strategy is sound, it is still crucial to *properly size the actuators and properly scale the sensors.*

- 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 deg. 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 DL06 provides 12–bit, 15–bit and 16- 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 modules. 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 DL06 analog input modules with 4 channels per module that accept 0 – 20mA or 4 – 20mA signals. Also, analog input and output combination modules are now available. 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***SOFT32 (4.0 or later). This software provides PID Setup dialog boxes which simplify the task. **Note**: It is important to understand the meaning of all loop parameters mentioned in this chapter before choosing values to enter.

8

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 (use 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 (see Loop Tuning on page 8–38) 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 Automatic Mode on page 8–39). In this step, you tune the loop 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.

Step 10: Save Parameters

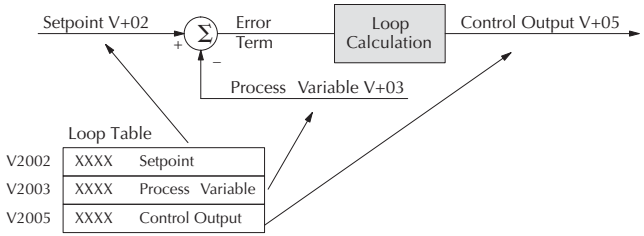
When the loop tests and tuning sessions are complete, be sure to save all loop setup parameters to disk.

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.

Basic Loop Operation

Data Locations

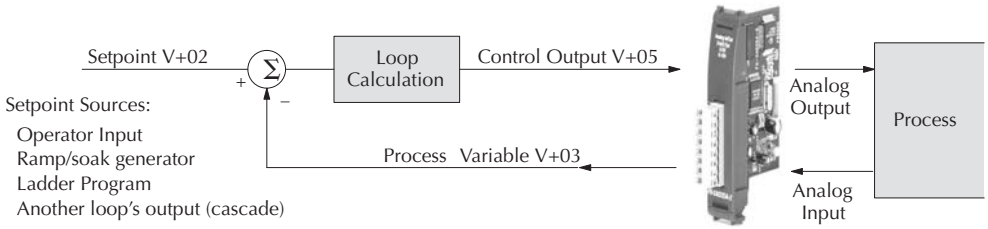
Each PID loop is dependent on the instructions and data values in its respective loop table. The following diagram shows an example of the loop table locations corresponding to the main three loop variables: SP, PV, and Control Output. The example below begins at V2000 (you can use any memory location compatible with Loop Table requirements). The SP, PV and Control Output are located at the addresses shown.



Data Sources

The data for the SP, PV, and Control Output must interface with real-world devices. In the figure below, the sources or destinations are shown for each loop variable. The Control Output and Process Variable values move through the analog input/output combination module to interface with the process itself.

A few rungs of ladder logic are required to copy data from the analog module to the loop table, or vice versa. Refer to the analog module chapter of this manual for an example of the required ladder logic.



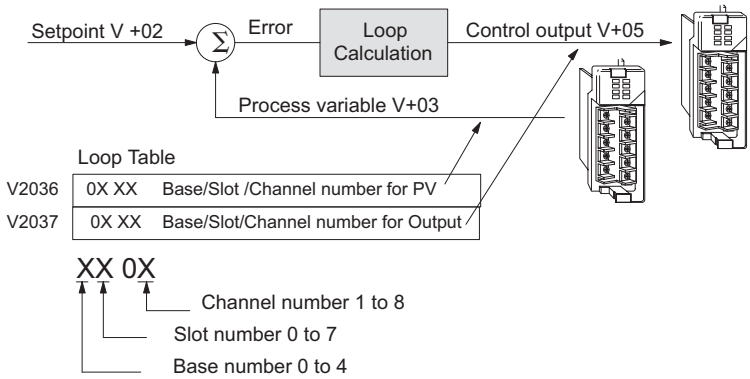
The Setpoint has several possible sources, as listed above. Many applications will use two or more of the sources at different times, depending on the loop mode. In addition, the loop control strategy and programming method also determine how the setpoint is generated.

When using the built-in Ramp/Soak generator or when cascading a loop, the PID controller automatically writes the setpoint data in location addr+02 for you. **If you want to use a setpoint from any other source, the ladder program must write that setpoint to the loop table location addr+02.**

Each of the three main loop parameters can have only one source or destination at any given time. During the application development, it is a good idea to draw loop schematic diagrams showing data sources, etc., to help avoid mistakes.

Auto Transfer to Analog I/O

The loop controller in the DL06 CPUs has the ability to directly access (referred to as auto transfer) analog I/O values or V–memory registers apart from the ladder logic scan. In particular, these parameters are the process variable (PV) and the control output. This feature is helpful if you must perform closed-loop PID control while the CPU is in Program Mode or if you wish to use the pointer method for the analog I/O or calculations in ladder logic to provide the PV values when in RUN mode. The loop controller can read the analog PV value in the selected data format from the desired analog module, and write the control output value to the desired output module. This auto transfer feature, when enabled, accesses the analog values only once per PID calculation for each respective loop. 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. The following figure shows the loop table parameters at addr+36 and addr+37 and their role in direct access to the analog values.



You may program these loop table parameters directly, or use the PID Setup feature in DirectSOFT32 for easy configuring. For example, a value of "0102" in register V2036 directs the loop controller to read the PV data from slot number 1, and the second channel. Note that slot 1 is the second slot to the right of the CPU, because slot 0 is adjacent to the CPU. 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 transfer the value between the loop table and the physical I/O module. If the PV or control output values require some math manipulation by ladder logic, then it will not be possible to use the auto transfer to/from I/O function of the loop controller. In this case, ladder logic will need to be used to perform the math and transfer the data to or from the analog modules as required.

NOTE: If the auto transfer to/from I/O function is used, the analog data for all of the channels on the analog modules being used with this feature cannot be accessed by any other method, i.e., pointer or multiplex.

PV Auto Transfer Functions with Filtering Options

The built-in filter uses the following algorithm **:**

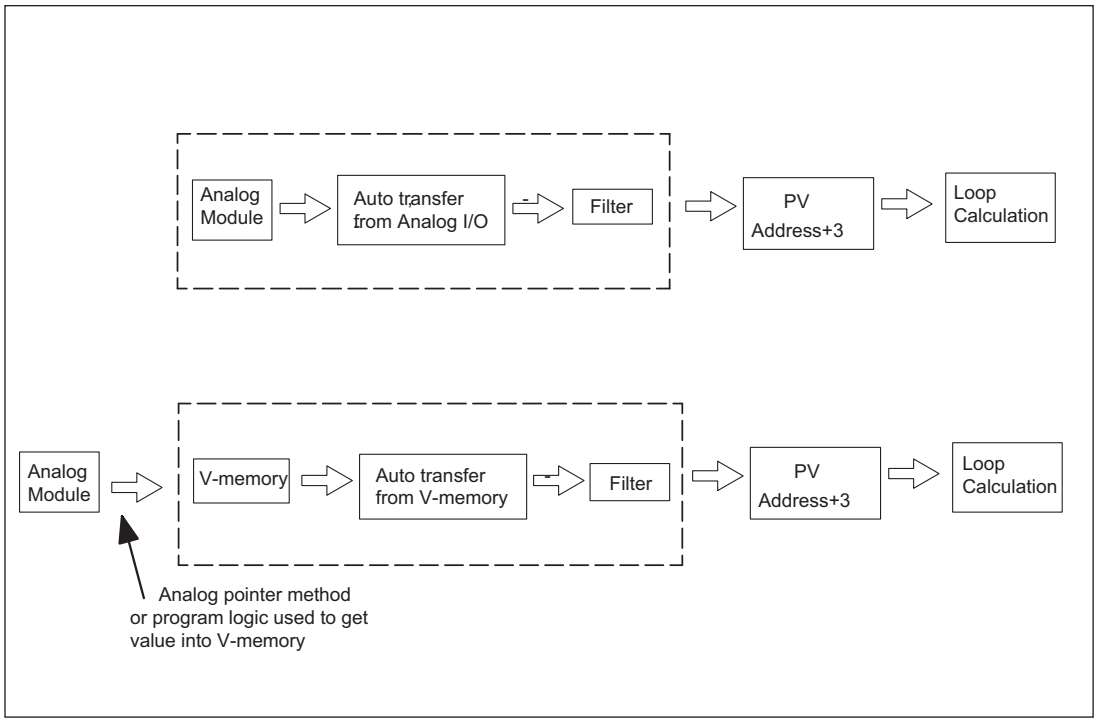
$$y_i = k\,(x_i - y_{i-1}) + y_{i-1}$$

- y_i is the current output of the filter
- x_i is the current input to the filter
- y_{i-1} is the previous output of the filter
- k is the PV Analog Input Filter Factor

The diagrams below show how the auto transfer function (address + 36) and PV filtering (address + 01, bit 2) interact. The options are:

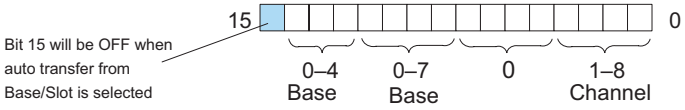
• Auto transfer directly from an analog I/O module channel with the filter enabled or disabled. When this function is used, the analog pointer method cannot be used to read the module's channel values.

• Auto–transfer directly from a V–memory location with the filter enabled or disabled. When this function is used, either the analog pointer method or program logic must be used to write a value to the V–memory location specified.



PV Auto Transfer (Addr + 36) from I/O Module Base/Slot/Channel Option

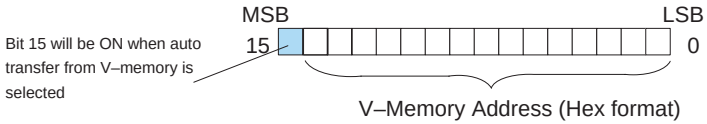
The nibble definitions for PV Auto Transfer word (Addr + 36) are listed in the table below for the Transfer from Base/Slot option. When this option is used for any channel on an analog input module, the ladder logic pointer method cannot be used for this module. (Refer to the DL06 Analog I/O Modules (D0–OPTIONS–M) for pointer method information).



Type	Base number	Slot number	Channel number
DL06	0 (Option card slot)	1-4	1-16

PV Auto Transfer (Addr + 36) from V–memory Option

The definitions for PV Auto Transfer word (Addr + 36) are listed in the table below for the Transfer from V–memory option. The ladder logic pointer method can be used with this option to get the analog module's channel values into V–memory. (Refer to the DL06 Analog I/O Modules (D0–OPTIONS–M) for pointer method information).

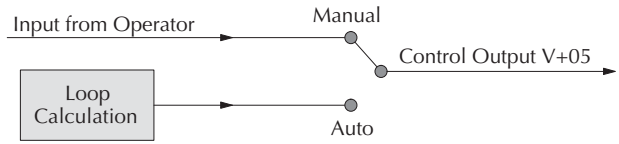


Type	V-Memory Range (Data Area)
DL06	V400-V677 / V1200-V7377 / V10000-V17777

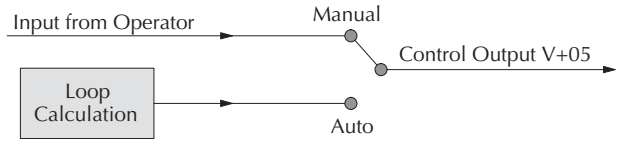
Loop Modes

The DL06 gives you 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. An introduction to the three control modes and their signal sources follows.

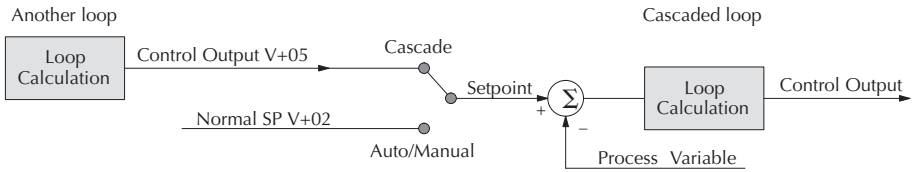
In **Manual Mode**, the loop is not executing PID calculations (however, loop alarms are still active). With regard to the loop table, the CPU stops writing values to location addr+05 for that loop. It is expected that an operator or other intelligent source is manually controlling the output, by observing the PV and writing data to addr+05 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 addr+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 addr+02, using the control output value from another loop. So in Auto or Manual modes, the loop calculation uses the data at addr+02. In Cascade Mode, the loop calculation reads the control output from another loop's parameter table.



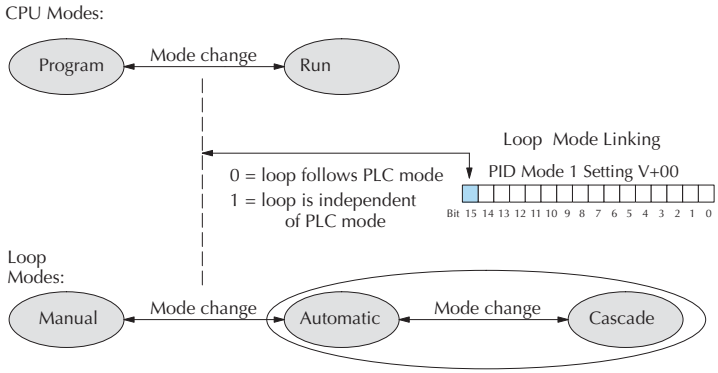
As pictured below, A loop can be changed from one mode to another, but *cannot go from Manual Mode directly to Cascade*, *or vice versa*. This mode change is prohibited because a loop would be changing two data sources at the same time, and could cause a loss of control.

CPU Modes and Loop Modes

The DL06 PLC has the ability to run PID calculations while the CPU is in Program Mode. Usually, a CPU in Program Mode has halted all operations. However, a DL06 PLC in Program Mode may or may not be running PID calculations (and providing PID control output), depending on your configuration settings. Having the ability to run loops independent of the ladder logic 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.

Loops that run independent of the ladder scan must have the ability to directly access the analog module channels for the PV and control output values. The loop controller does have this capability, which is covered in the section on direct access of analog I/O (located prior to this section in this chapter).

The relationship between CPU modes and loop modes is depicted in the figure below. The vertical dashed line shows the optional relationship between the mode changes. Bit 15 of PID Mode 1 setting word (addr+00) determines the selection. If set to zero so the loop follows the CPU mode, then placing the CPU in Program Mode will force all loops into Manual Mode. Similarly, placing the CPU in Run mode will allow each loop to return to the mode it was in previously (which includes Manual, Automatic, and Cascade). With this selection you

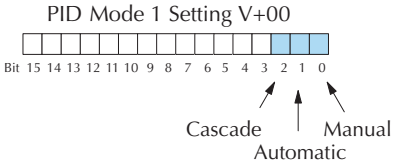


automatically affect the modes of the loops by changing the CPU mode.

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 ladders and the other is running the process loops.

***NOTE**: To make changes to any **loop table parameter values**, the PID loop must be in Manual Mode and the PLC must be stopped. If you have selected (as shown above) 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 your programming device (such as **Direct**SOFT32) will be able to place the loop into Manual Mode. After you change the loop's parameter setting, just restore bit 15 to a value of 1 to re-establish PID operation independent of the CPU.*

How to Change Loop Modes

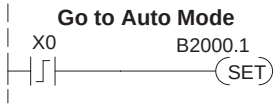


The first three bits of the PID Mode 1 word (addr+00) request the operating mode of the corresponding loop. Note: these bits are mode change *requests*, not commands (certain conditions can prohibit a particular mode change – see next page).

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

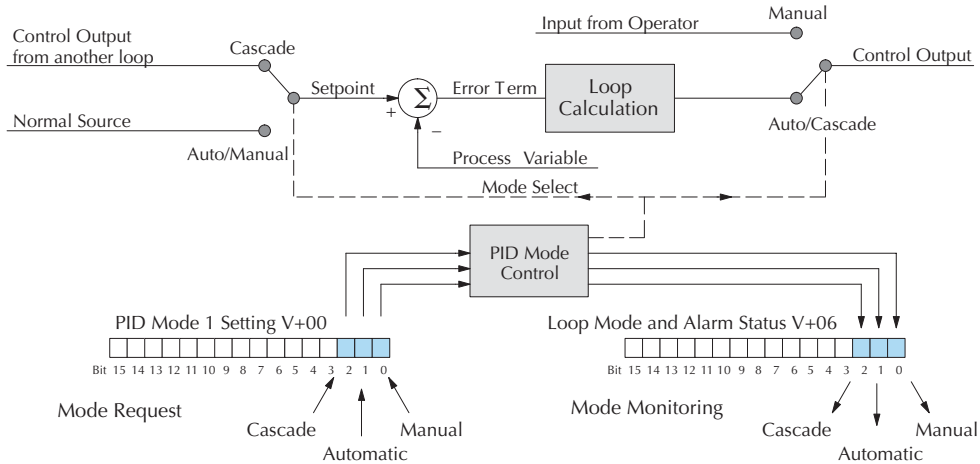
- ***Direct***SOFT32's **PID View** – this is the easiest method. Click on one of the radio buttons, and ***Direct***SOFT32 sets the appropriate bit.
- **HPP** – Use Word Status (WD ST) to monitor the contents of addr+00, which will be a 4-digit BCD/hex value. You must calculate and enter a new value for addr+00 that ORs the correct mode bit with its current value.
- **Ladder program**– ladder logic can request any loop mode when the PLC is in Run Mode. This will be necessary after application startup.

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



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

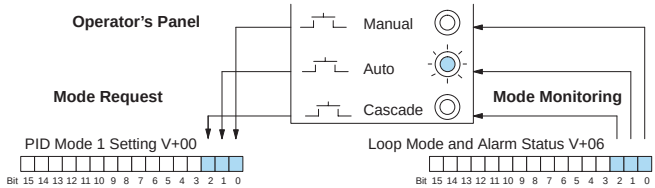
Since we can only *request* mode changes, the PID loop controller decides when to permit mode changes and provides the loop mode status. It reports the current mode on bits 0, 1, and 2 of the Loop Mode and Alarm Status word, location addr+06 in the loop table. The parallel request / monitoring functions 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 a few advanced 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 8 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 addr+00, bits 0, 1, and 2. However, some conditions exist which will prevent a requested mode change from occurring:

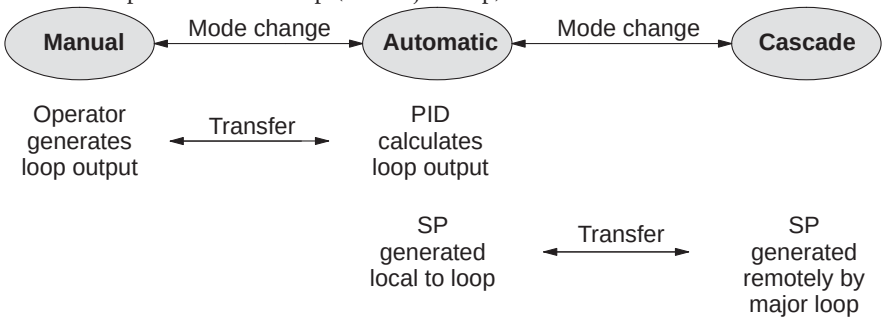
- A loop that is not set independent of PLC mode cannot change modes when the PLC is in Program mode.
- A major loop of a cascaded pair of loops cannot go from Manual to Auto until its minor loop is in Cascade mode.

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.

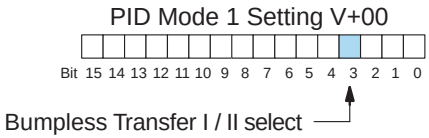
Bumpless Transfers

In process control, the word "transfer" has a particular meaning. A loop transfer occurs when we change its mode of operation, as shown below. When we change loop modes, what we are really doing is causing a transfer of control of some loop parameter from one source to another. For example, when a loop changes from Manual Mode to Automatic Mode, control of the output changes from the operator to the loop controller. When a loop changes from Automatic Mode to Cascade Mode, control of the SP changes from its original source in Auto Mode to the output of another loop (the major loop).



The basic problem of loop transfers is the two different sources of the loop parameter being transferred will have different numerical values. This causes the PID calculation to generate an undesirable step change, or "bump" on the control output, thereby upsetting the loop to some degree. The "bumpless transfer" feature arbitrarily forces one parameter equal to another at the moment of loop mode change, so the transfer is smooth (no bump on the control output).

The bumpless transfer feature of the DL06 loop controller is available in two types: Bumpless I, and Bumpless II. Use ***Direct***SOFT32's PID Setup dialog box to select transfer type. Or, you can use bit 3 of PID Mode 1 addr+00 setting as shown.



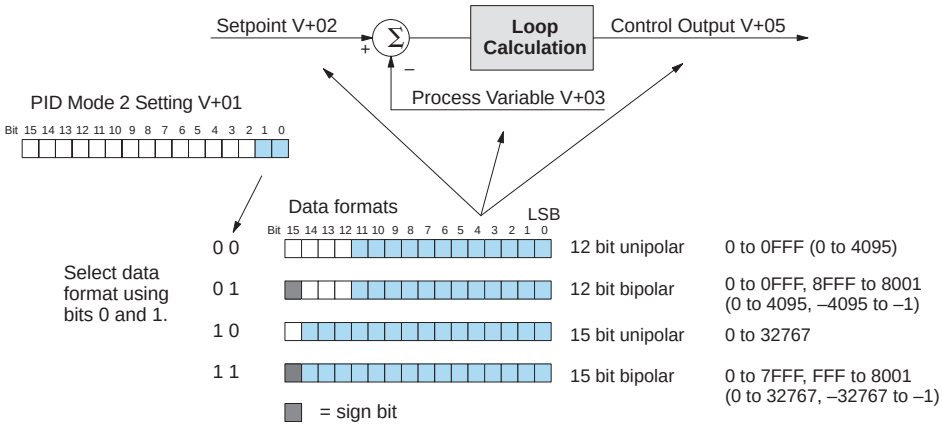
The characteristics of Bumpless I and II transfer types are listed in the chart below. Note that their operation also depends on which PID algorithm you are using, the position or velocity form of the PID equation. Note that you must use Bumpless Transfer type I when using the velocity form of the PID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

TransferType	Transfer Select Bit	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

PID Loop Data Configuration

Loop Parameter Data Formats

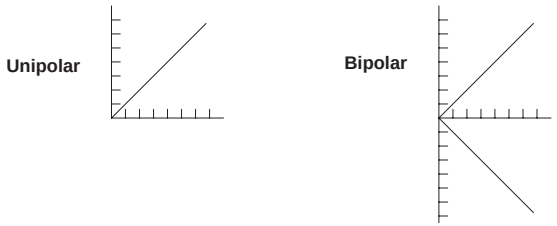
In choosing the Process Variable range and resolution, a related choice to make is the data format of the three main loop variables: SP, PV, and Control Output (the Integrator sum in addr+04 also uses this data format). The four data formats available are 12 or 15 bit (right justified), signed or unsigned (MSB is sign bit in bipolar formats). The four binary combinations of bits 0 and 1 of PID Mode 2 word addr+01 choose the format. The *Direct*SOFT32 PID Setup dialog sets these bits automatically when you select the data format from the menu.



The data format is a very powerful setting, because it determines the numerical interface between the PID loop and the PV sensor, and the Control Output device. The Setpoint must also be in the same data format. Normally, the data format is chosen during the initial loop configuration and is not changed again.

Choosing Unipolar or Bipolar Format

Choosing the data format involves deciding whether to use unipolar or bipolar numbers. Most applications such as temperature control will use only positive numbers, and therefore need unipolar format. Usually it is the Control Output which determines bipolar/unipolar selection. For example, velocity control may include control of forward and reverse directions. At a zero velocity setpoint the desired control output is also zero. In that case, bipolar format must be used.



Handling Data Offsets

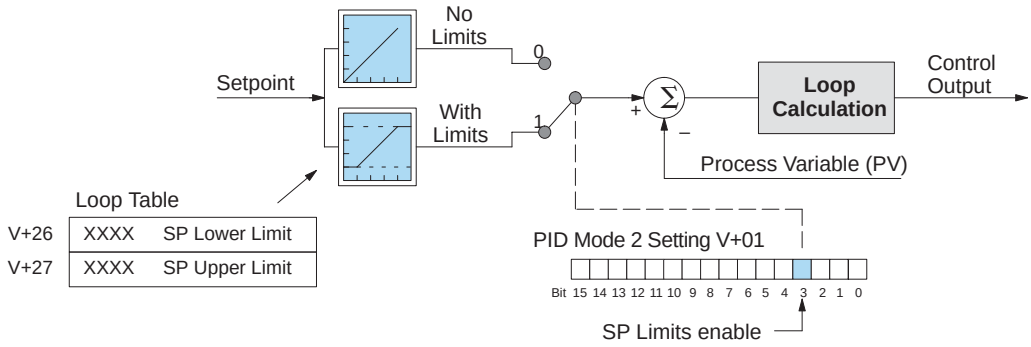
In many batch process applications, sensors or actuators interface to DL06 analog modules using 4–20 mA signals. This signal type has a built-in 20% offset, because the zero-point is a 4 mA instead of 0 mA. However, remember the analog modules convert the signals into data *and remove the offset at the same time*. For example, a 4–20 mA signal is often converted to 0000 – 0FFF hex, or 0 to 4095 decimal. In this case, all you need to do is choose 12-bit unipolar data format, and make sure the ladder program copies the data appropriately between the loop table and the analog modules.

- **PV Offset** – In the event you have a PV value with a 20% offset, convert it to zero–offset by subtracting 20% of the top of its range, and multiply by1.25.
- **Control Output** – In the event the Control Output is going to a device with 20% offset, all you need to do is have the ladder program write a value equivalent to the offset to the integrator register (addr+04), before transitioning from Manual to Auto mode. The loop will then see this offset as a part of the process, taking care of it for you automatically.

Setpoint (SP) Limits

The Setpoint in loop table location addr+02 represents the desired value of the process variable. After selecting the data format for these variables, you can set limits on the range of SP values which the loop calculation will use. Many loops have two or more possible sources writing the Setpoint at various times, and the limits you set will help safeguard the process from the effects of a bad SP value.

In the figure below, the SP has a selectable limit function, enabled by PID Mode 2 Setting addr+01 word, bit 3. If enabled, then locations addr+26 and addr+27 determine the lower and upper SP limits, respectively. The loop calculation applies this limit internally, so it is always possible to write any value to addr+02.

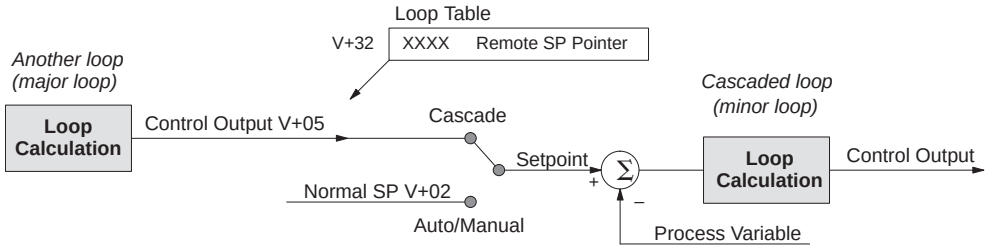


The loop calculation checks these SP upper and lower limits before each calculation. This means ladder logic can change the limit settings while a process is in progress, allowing you to keep a tighter guard band on the SP input value.

Remote Setpoint (SP) Location

You may recall there are generally several possible data sources for the SP value. The PID loop controller has the built-in ability to select between two sources according to the current loop mode. Refer to the figure below. A loop reads its setpoint from table location addr+02 in Auto or Manual modes. If you plan to use Cascade Mode for the loop at any time, then you must program its loop parameter table with a *remote setpoint pointer*.

The Remote SP pointer resides in location addr+32 in the loop table. For loops that will be cascaded (made a minor loop), you will need to program this location with the address of the major loop's Control Output address. Find the starting location of the major loop's parameter table and add offset +05 to it.

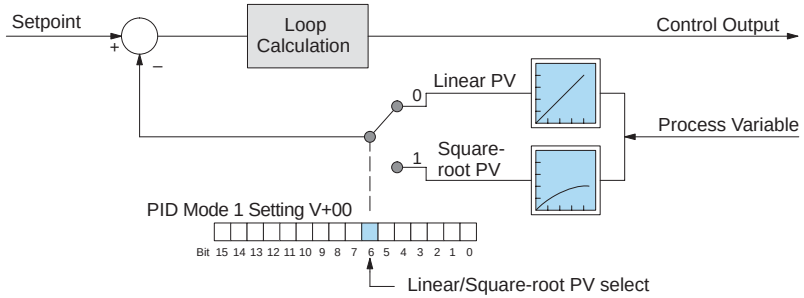


A *Direct*SOFT32 Loop Setup dialog box will allow you to enter the Remote SP pointer if you know the address. Otherwise, you can enter it with a HPP or program it through ladder logic using the LDA instruction.

Process Variable (PV) Configuration

The process variable input to each loop is the value the loop is ultimately trying to control, to make it equal to the setpoint and follow setpoint changes as quickly as possible. Most sensors for process variables have a primarily linear response curve. Most temperature sensors are mostly linear across their sensing range. However, flow sensing using an orifice plate technique gives a signal representing (approximately) the square of the flow. Therefore, a square-root extract function is necessary before using the signal in a linear control system (such as PID).

Some flow transducers are available which will do the square-root extract, but they add cost to the sensor package. The PID loop PV input has a selectable square-root extract function, pictured below. You can select between normal (linear) PV data, and data needing a square-root extract by using PID Mode setting addr+00 word, bit 6.



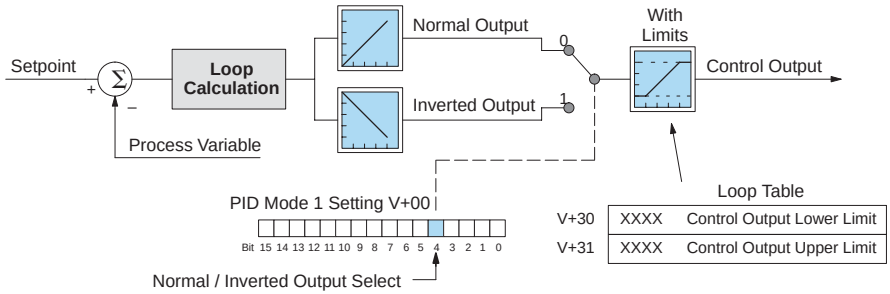
IMPORTANT: The scaling of the SP must be adjusted if you use PV square-root extract, because the loop drives the output so the square root of the PV is equal to the PV input. Divide the desired SP value by the square root of the analog span, and use the result in the addr+02 location for the SP. This does reduce the resolution of the SP, but most flow control loops do not require a lot of precision (the recipient of the flow is integrating the errors). Use one of the following formulas for the SP according to the data format you are using. It's a good idea to set the SP upper limit to the top of the allowed range.

Data Format	SP Scaling	SP Range	PV range
12-bit	SP = PV input / 64	0 – 64	0 – 4095
15-bit	SP = PV input / 181.02	0 – 181	0 – 32767
16-bit	SP = PV input / 256	0 – 256	0 – 65535

Control Output Configuration

The Control Output is the numerical result of the PID calculation. All of the other parameter choices ultimately influence the value of a loop's Control Output for each calculation. Some final processing selections dedicated to the Control Output are available, shown below. At the far right of the figure, the final output may be restricted by lower and upper limits that you program. The values for addr+30 and addr+31 may be set once using *Direct*SOFT32's PID Setup dialog box.

The Control Output lower and upper limits can help guard against commanding an excessive correction to an error when a loop fault occurs (such as PV sensor signal loss). However, do not use these limits to restrict mechanical motion that might otherwise damage a machine (use hard-wired limit switches instead).



The other available selection is the normal/inverted output selection (called "forward/reverse" in *Direct*SOFT32). Use bit 4 of the PID Mode 1 Setting addr+00 word to configure the output. Independently of unipolar or bipolar format, a normal output goes upward on positive errors and downward on negative errors (where Error=(SP–PV)). The inverted output reverses the direction of the output change.

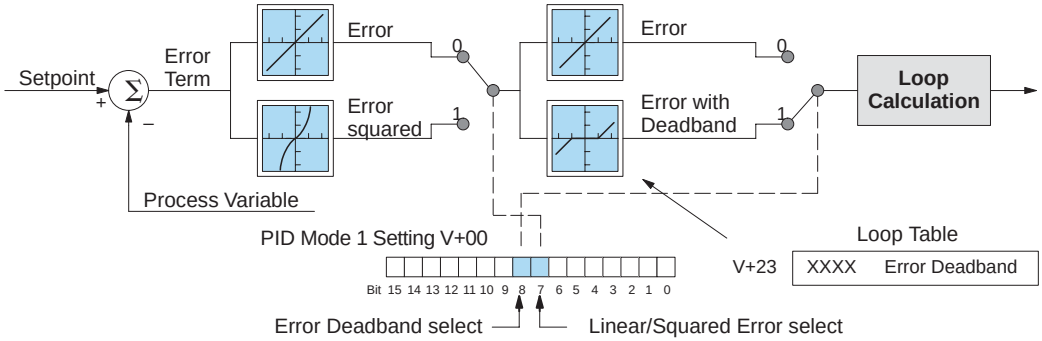
The normal/inverted output selection is used to configure direct-acting/reverse-acting loops. This selection is ultimately determined by the direction of the response of the process variable to a change in the control output in a particular direction. Refer to the PID Algorithms section for more on direct-acting and reverse-acting loops.

Error Term Configuration

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). If the PV square-root extract is enabled, then Error = (SP – (sqrt(PV)). In any case, the size of the error and algebraic sign determine the next change of the control output for each PID calculation.

Now we will superimpose some "special effects" on to the error term as described. Refer to the diagram below. Bit 7 of the PID Mode Setting 1 addr+00 word lets you select a linear or squared error term, and bit 8 enables or disables the error deadband.

***NOTE**: When first configuring a loop, it's best to use the standard error term. After the loop is tuned, then you will be able to tell if these functions will enhance control.*



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.

Error Deadband – When selected, the error 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.

Loop parameter location addr+23 must be programmed with a desired deadband amount. Units are the same as the SP and PV units (0 to FFF in 12-bit mode, and 0 to 7FFF in 15-bit mode). The PID loop controller automatically applies the deadband symmetrically about the zero-error point.

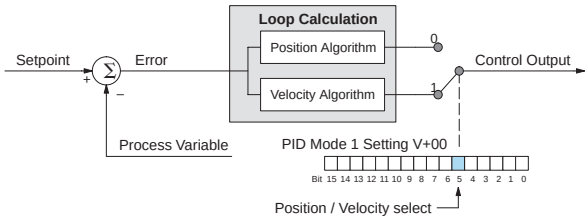
PID Algorithms

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 DL06 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 vise-versa).

The DL06 features 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.

The majority of applications will use the position form of the PID equation. If you are not sure of which algorithm to use, try the Position Algorithm first. Use ***Direct***SOFT32's PID View Setup dialog box to select the desired algorithm. Or, use bit 5 of PID Mode 1 Setting addr+00 word as shown below to select the algorithm.



***NOTE**: The selection of a PID algorithm is very fundamental to control loop operation, and is normally never changed after the initial configuration of a loop.*

Position Algorithm

The Position Algorithm causes the PID equation to calculate the Control Output Mn:

$$M_n = K_c * e_n + K_i * \sum_{i=1}^{n} e_i + K_r * (e_n - e_{n-1}) + M_o$$

In the formula above, the sum of the integral terms and the initial output are combined into the "Bias" term, Mx. Using the bias term, we define formulas for the Bias and Control Output as a function of sampling time:

$$Mx_o = M_o$$

$$Mx_n = K_i * e_n + Mx_{n-1}$$

$$M_n = K_i * \sum_{i=1}^{n} e_i + M_o$$

$$M_n = K_c * e_n + K_r * (e_n - e_{n-1}) + Mx_n \text{.....Output for sampling time "n"}$$

The position algorithm variables and related variables are:

Ts = Sample rate

Kc = Proportional gain

Ki = Kc * (Ts/Ti) coefficient of integral term

Kr = Kc * (Td/Ts) coefficient of derivative term

Ti = Reset time (integral time)

Td = Rate time (derivative time)

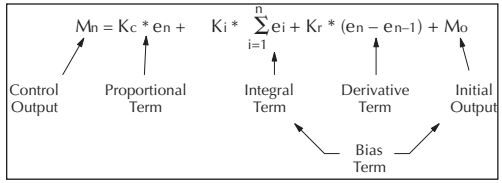
SPn = Set Point for sampling time "n" (SP value)

PVn = Process variable for sampling time "n" (PV)

en = SPn – PVn = Error term for sampling time "n"

M0 = Control Output for sampling time "0"

Mn = Control Output for sampling time "n"



Analysis of these equations will be found in most good text books on process control. At a glance, we can isolate the parts of the PID Position Algorithm which correspond to the P, I, and D terms, and the Bias as shown above.

The initial output is the output value assumed from Manual mode control when the loop transitioned to Auto Mode. The sum of the initial output and the integral term is the bias term, which holds the "position" of the output. Accordingly, the Velocity Algorithm discussed next does not have a bias component.

Velocity Algorithm

The Velocity Algorithm form of the PID equation can be obtained by transforming Position Algorithm formula with subtraction of the equation of (n–1)th degree from the equation of nth degree.

The velocity algorithm variables and related variables are:

Ts = Sample rate

Kc = Proportional gain

Ki = Kc * (Ts/Ti) = coefficient of integral term

Kr = Kc * (Td/Ts) = coefficient of derivative term

Ti = Reset time (integral time)

Td = Rate time (derivative time)

SPn = Set Point for sampling time "n" (SP value)

PVn = Process variable for sampling time "n" (PV)

en = SPn – PVn = Error term for sampling time "n"

Mn = Control Output for sampling time "n"

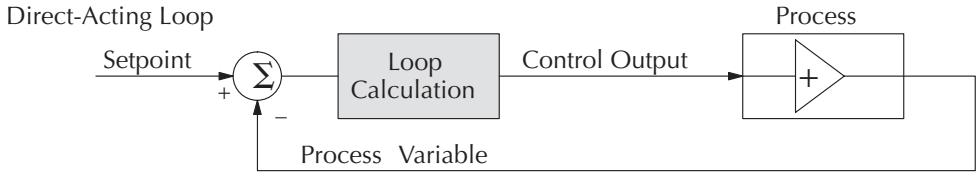
The resulting equations for the Velocity Algorithm form of the PID equation are:

$$\Delta M_n = M_n - M_{n-1}$$
$$\Delta M_n = K_c * (e_n - e_{n-1}) + K_i * e_n + K_r * (e_n - 2*e_{n-1} + e_{n-2})$$

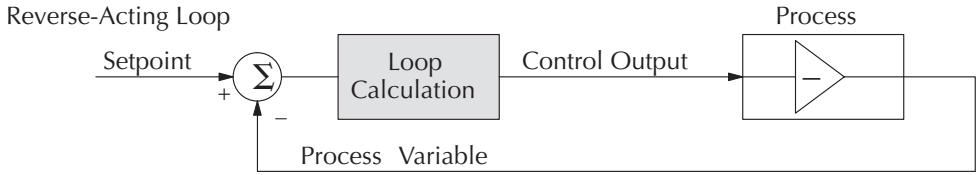
Direct-Acting and Reverse-Acting Loops

The gain of a process determines, in part, how it must be controlled. The process shown in the diagram below has a positive gain, which we call "direct-acting". This means that when the control output increases, the process variable also eventually increases. Of course, a true process is usually a complex transfer function that includes time delays. Here, we are only interested in the direction of change of the process variable in response to a control output change.

Most process loops will be direct-acting, such as a temperature loop. An increase in the heat applied increases the PV (temperature). Accordingly, direct-acting loops are sometimes called *heating loops*.



A "reverse-acting" loop is one in which the process has a negative gain, as shown below. An increase in the control output results in a decrease in the PV. This is commonly found in refrigeration controls, where an increase in the cooling input causes a decrease in the PV (temperature). Accordingly, reverse-acting loops are sometimes called *cooling loops*.

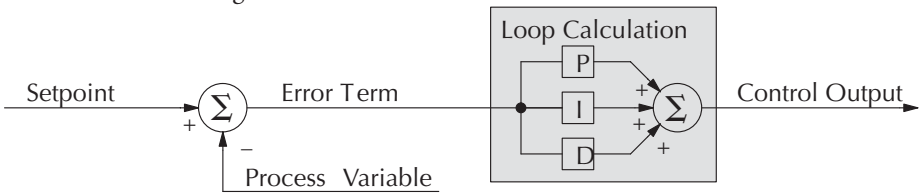


It is crucial to know whether a particular loop is direct or reverse-acting! Unless you are controlling temperature, there is no obvious answer. In a flow control loop, a valve positioning circuit can be configured and wired reverse-acting as easily as direct-acting. One easy way to find out is to run the loop in Manual Mode, where you must manually generate control output values. Observe whether the PV goes up or down in response to a step increase in the control output.

To run a loop in Auto or Cascade Mode, the control output must be correctly programmed (refer to the previous section on Control Output Configuration). Use "normal output" for direct-acting loops, and "inverted output" for reverse-acting loops. To compensate for a reverse-acting loop, the PID controller must know to invert the control output. If you have a choice, configure and wire the loop to be direct-acting. This will make it easier to view and interpret loop data during the loop tuning process.

P-I-D Loop Terms

You may recall the introduction of the position and velocity forms of the PID loop equations. The equations basically show the three components of the PID calculation. The following figure shows a schematic form of the PID calculation, in which the control output is the sum of the proportional, integral and derivative terms. On each calculation of the loop, each term receives the same error signal value.

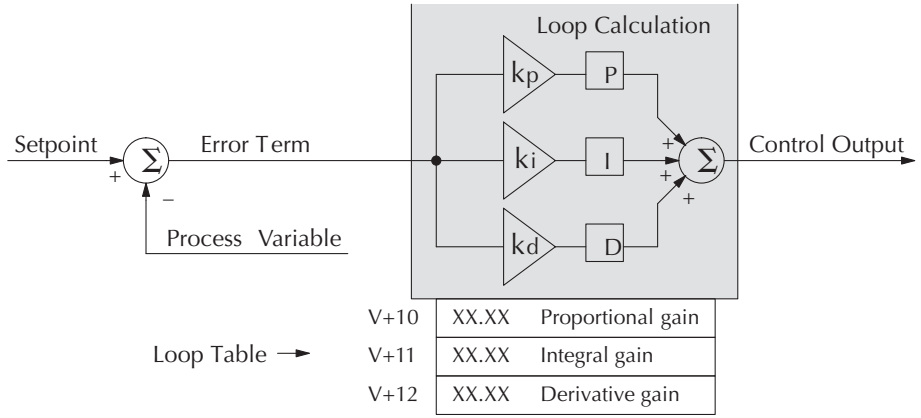


The role of the P, I, and D terms in the control task are as follows:

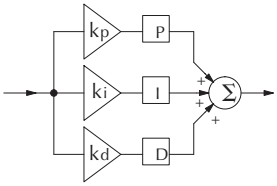
- **Proportional** – the proportional term simply responds proportionally to the current size of the error. This loop controller calculates a proportional term value for each PID calculation. When the error is zero, the proportional term is also zero.
- **Integral** – the integrator (or reset) term integrates (sums) the error values. Starting from the first PID calculation after entering Auto Mode, the integrator keeps a running total of the error values. For the position form of the PID equation, when the loop reaches equilibrium and there is no error, the running total represents the constant output required to hold the current position of the PV.
- **Derivative** – the derivative (or rate) term responds to change in the current error value from the error used in the previous PID calculation. Its job is to anticipate the probable growth of the error and generate a contribution to the output in advance.

The P, I, and D terms work together as a team. To do that effectively, they will need some additional instructions from us. The figure below shows the P, I, and D terms contain programmable **gain** values kp, ki, and kd respectively. The values reside in the loop table in the locations shown. The goal of the loop tuning process (covered later) is to derive gain values that result in good overall loop performance.

NOTE: The proportional gain is also simply called "gain", in PID loop terminology.



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 – Integral gain may be in units of seconds or minutes, by programming the bit shown. Derivative gain is in seconds.



V+10	XX.XX P gain	–
V+11	XX.XX I gain	0=sec, 1=min.
V+12	XX.XX D gain	sec.

PID Mode 2 Setting V+01

Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Units select

In *Direct*SOFT32's trend view, you can program the gain values and units in realtime while the loop is running. This is typically done only during the loop tuning process.

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.

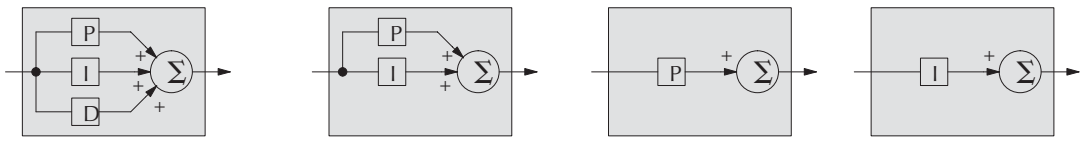
Integral Gain – Values range from 0001 to 9998, but they are used internally as xx.xx. An entry of "0000" or "9999"causes the integral gain to be "∞", effectively removing the integrator term from the PID equation. This accommodates applications which need proportional-only loops. The units of integral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

Derivative Gain – Values 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 need proportional and/or integral-only loops. The derivative term has an optional gain limiting feature, discussed in the next section.

***NOTE**: It is very important to know how to increase and decrease the gains. The proportional and derivative gains are as one might expect... smaller numbers produce less gains and larger numbers produce more gain. However, the integral term has a reciprocal gain(1/Ts), so smaller numbers produce more gain and larger numbers produce less gain. This is very important to know during loop tuning.*

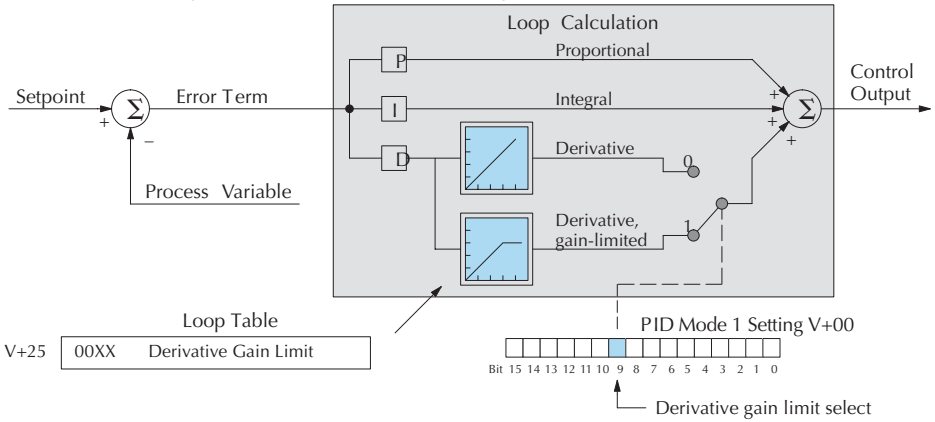
Using a Subset of PID Control

Each of the P, I, and D gains allows a setting to eliminate that term from the PID equation. Many applications actually work best by using a subset of PID control. The figure below shows the various combinations of PID control offered on the DL06. We do not recommend using any other combination of control, because most of them are inherently unstable.



Derivative Gain Limiting

The derivative term is unique in that it has an optional gain-limiting feature. This is provided because the derivative term 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. Use bit 9 of PID Mode 1 Setting addr+00 word to enable the gain limit.



The derivative gain limit in location addr+25 must have a value between 0 and 20, in BCD format. This setting is operational only when the enable bit = 1.

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

Bias Term

In the widely-used position form of the PID equation, an important component of the control output value is the bias term shown below. Its location in the loop table is in addr+04. the loop controller writes a new bias term after each loop calculation.

$$M_n = K_c * e_n + K_i * \sum_{i=1}^{n} e_i + K_r * (e_n - e_{n-1}) + M_o$$

Control Output | Proportional Term | Integral Term | Derivative Term | Initial Output

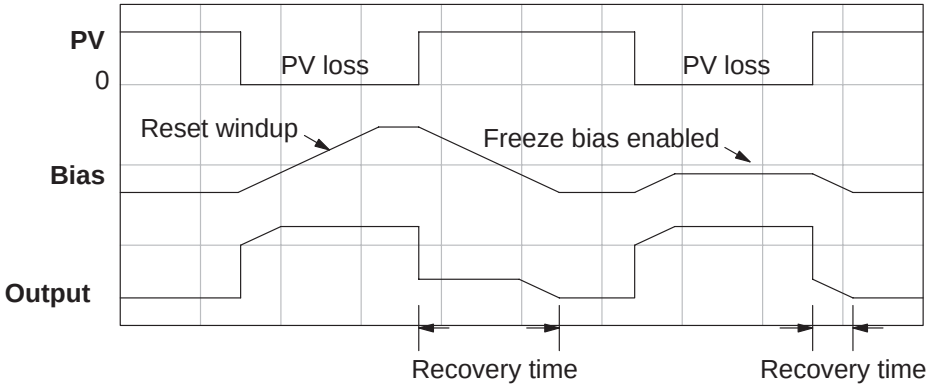
V+04 XXXX Bias term — Bias Term

If we cause the error (en) to go to zero for two or more sample periods, the proportional and derivative terms cancel. The bias term is the sum of the integral term and the initial output (Mo). It represents the steady, constant part of the control output value, and is similar to the DC component of a complex signal waveform.

The bias term value establishes a "working region" for the control output. *When the error fluctuates around its zero point, the output fluctuates around the bias value*. This concept is very important, because it shows us why the integrator term must respond more slowly to errors than either the proportional or derivative terms.

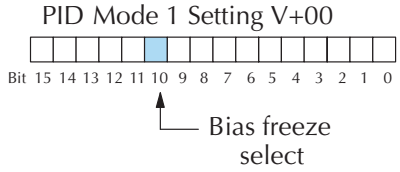
Bias Freeze

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 out of bounds. 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. You may enable the feature using the ***Direct***SOFT32 PID View setup dialog, or set bit 10 of PID Mode 1 Setting word as shown to the right.



NOTE: The bias freeze 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/12bit loop), the bias term still uses the end of range for the stopping point and bias freeze will not work.

In the feedforward method discussed later in this chapter, ladder logic writes directly to the bias term value. However, there is no conflict with the freeze bias feature, because bias term writes due to feedforward are relatively infrequent when in use.

Loop Tuning Procedure

This is perhaps the most important step in closed-loop process control. The goal of a loop tuning procedure 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.

Auto Tuning versus Manual Tuning – you may change the PID gain values directly (manual tuning), or you can have the PID processing engine in the CPU automatically calculate the gains (auto tuning). Most experienced process engineers will have a favorite method, and the DL06 will accommodate either preference. The use of the 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, note that performing the auto tuning procedure will get the gains close to optimal values, but additional manual tuning changes can take 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 DL06 is not intended to perform 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 the source which is to generate the setpoint can do so. You can put the PLC in Run Mode, but leave the loop in Manual Mode. Then monitor the loop table location addr+02 to see the SP value(s). The ramp/soak generator (if you are using it) should be tested now.
- **Process Variable** – verify the PV value is an accurate measurement, and the PV data arriving in the loop table location addr+03 is correct. If the PV signal is very noisy, consider filtering the input either through hardware (RC low-pass filter), or using a digital S/W filter.
- **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 give earlier in this chapter). However, if you are going to use auto tuning, note the auto tuning procedure will automatically calculate the sample rate in addition to the PID gains.

The discussion beginning on the following page covers the manual closed loop tuning procedure. If you want to perform only auto tuning, please skip the next section and proceed directly to the section on auto tuning.

Manual Closed Loop Tuning Procedure

Now comes the exciting moment when we actually close the loop (go to Auto Mode) for the first time. Use the following checklist **before** switching to Auto mode:

- **Monitor** the loop parameters with a loop trending instrument. We recommend using the PID view feature of ***Direct***SOFT32.

NOTE: We recommend using the PID trend view setup menu to select the vertical scale feature to manual, for both SP/PV area and Bias/Control Output areas. The auto scaling feature will otherwise change the vertical scale on the process parameters and add confusion to the loop tuning process.

- **Adjust** the gains so the Proportional Gain = 10, Integrator Gain = 9999, and Derivative Gain =0000. This disables the integrator and derivative terms, and provides a little proportional gain.
- **Check** the bias term value in the loop parameter table (addr+04). If it is not zero, then write it to zero using ***Direct***SOFT32 or HPP, etc.

Now we can transition the loop to Auto Mode. Check the mode monitoring bits to verify its true mode. If the loop will not stay in Auto Mode, check the troubleshooting tips at the end of this chapter.

CAUTION: If the PV and Control Output values begin to oscillate, reduce the gain values immediately. If the loop does not stabilize immediately, then transfer the loop back to Manual Mode and manually write a safe value to the control output. During the loop tuning procedure, always be near the Emergency Stop switch which controls power to the loop actuator in case a shutdown is necessary.

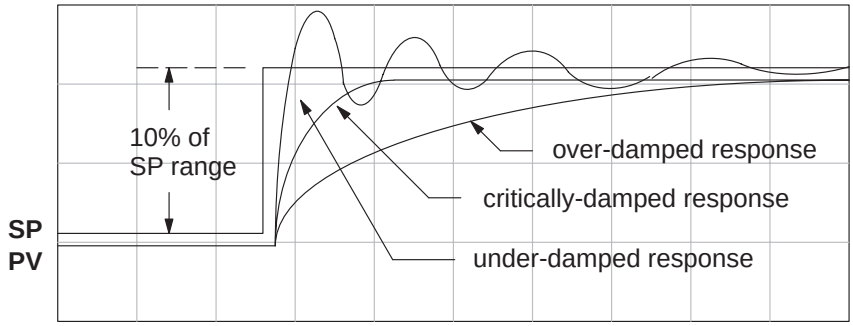
- At this point, the SP should = PV because of the bumpless transfer feature. Increase the SP a little, in order to develop an error value. With only the proportional gain active and the bias term=0, we can easily check the control output value:

  Control Output = (SP – PV) x proportional gain

- If the control output value changed, the loop should be getting more energy from the actuator, heater, or other device. Soon the PV should move in the direction of the SP. If the PV does not change, then increase the proportional gain until it moves slightly.
- Now, add a small amount of integral gain. ***Remember that large numbers are small integrator gains and small numbers are large integrator gains!*** After this step, the PV should = SP, or be very close.

Until this point we have only used proportional and integrator gains. Now we can "bump the process" (change the SP by 10%), and adjust the gains so the PV has an optimal response. Refer to the figure below. Adjust the gains according to what you see on the PID trend view. The critically- damped response shown gives the fastest PV response without oscillating.

- Over-damped response – the gains are too small, so gradually increase them, concentrating on the proportional gain first.
- Under-damped response – the gains are too large. Reduce the integral gain first, and then the proportional gain if necessary.
- Critically-damped response – this is the the optimal gain setting. You can verify that this is the best response by increasing the proportional gain slightly. the loop then should make one or two small oscillations.



Now you may want to add a little derivative gain to further improve the critically-damped response above. Note the proportional and integral gains will be very close to their final values at this point. Adding some derivative action will allow you to increase the proportional gain slightly without causing loop oscillations. The derivative action tends to tame the proportional response slightly, so adjust these gains together.

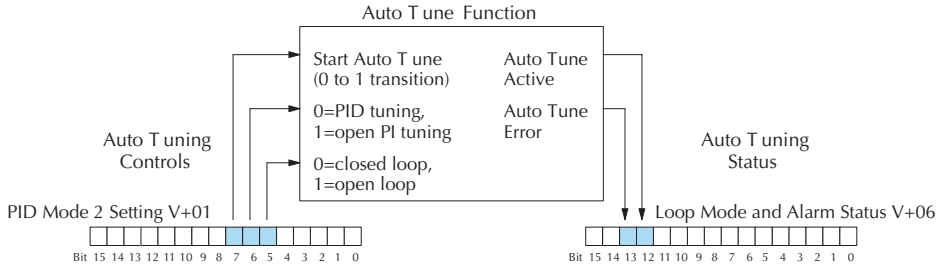
Auto Tuning Procedure

Autotuning is initiated within ***Direct***SOFT32. You can use autotuning to establish initial PID parameter values (autotuning is not run continuously during operation). Whenever a substantial change in loop dynamics occurs (mass of process, size of actuator, etc.), you will need to repeat the tuning procedure to derive the new gains that are required for optimal control.

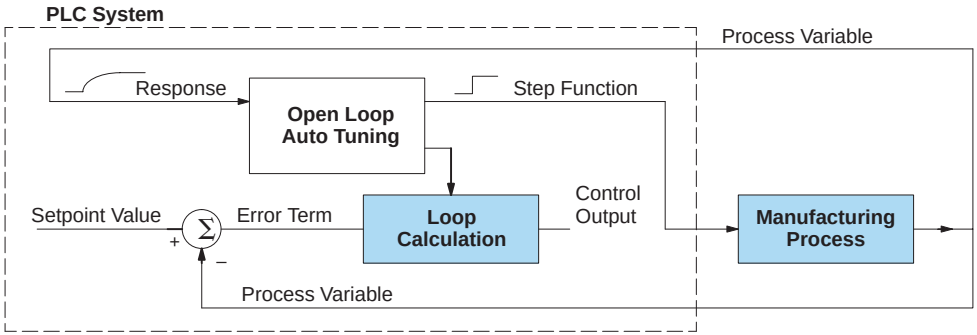
WARNING: Only authorized personnel fully familiar with all aspects of the process should make changes that affect the loop tuning constants. Using the loop auto tuning procedures will affect the process, including inducing large changes in the control output value. Make sure you thoroughly consider the impact of any changes to minimize the risk of injury to personnel or damage to equipment. The auto tune in the DL06 is not intended to perform as a replacement for your process knowledge.

The loop controller offers both closed-loop and open-loop methods. If you intend to use the auto tune feature, we recommend you use the open-loop method first. This will permit you to use the closed-loop method of auto tuning when the loop is operational (Auto Mode) and cannot be shut down (Manual Mode). 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 use three bits in the PID Mode 2 word addr+01, as shown below. *Direct*SOFT32 will manipulate these bits automatically when you use the auto tune feature within *Direct*SOFT32. Or, you may have 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 addr+06 reports the auto tune status as shown. Bit 12 will be on (1) when during the auto tuning cycle, automatically returning to off (0) when done.



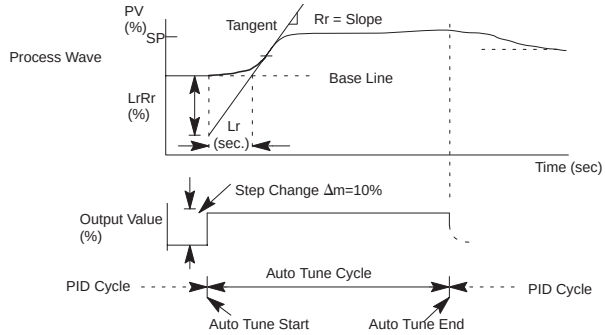
Open-Loop Auto Tuning – During an open-loop auto tuning cycle, the loop controller operates as shown in the diagram below. Before starting this procedure, place the loop in Manual mode and ensure the PV and control output values are in the middle of their ranges (away from the end points).



***NOTE**: In theory, the SP value does not matter in this case, because the loop is not closed. However, the firmware requires that the SP value be more than 205 counts away from the PV value before starting the auto tune cycle (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 Δm=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 Rr (maximum slope in %/sec.) and Lr (dead time in sec). The auto tune function computes the gains according to the Ziegler-Nichols equations, shown below:

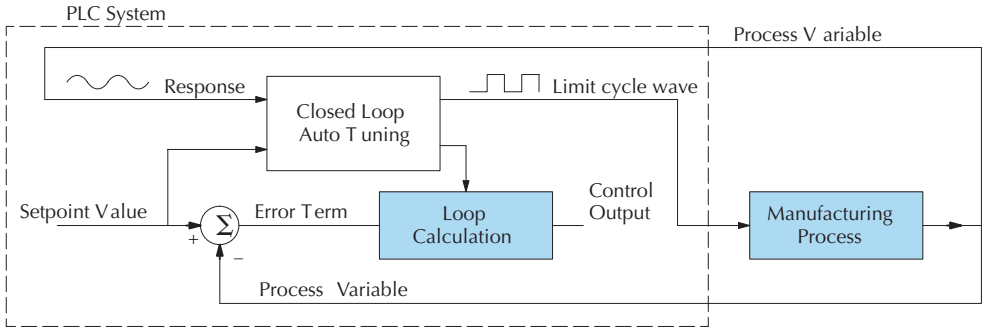
We highly recommend using ***Direct***SOFT32 for the auto tuning interface. The duration of each auto tuning cycle will depend on the mass of our process. A slowly-changing PV will result in a longer auto tune cycle time. When the auto tuning is complete, the proportional,

PID Tuning	PI Tuning
P=1.2*Δm/LrRr	P=0.9*Δm/LrRr
I=2.0* Lr	I=3.33* Lr
D=0.5* Lr	D=0
Sample Rate = 0.056* Lr	Sample Rate = 0.12*Lr
Δm = Output step change (10% = 0.1, 20% = 0.2)	

integral, and derivative gain values are automatically updated in loop table locations addr+10, addr+11, and addr+12 respectively. The sample time in addr+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 section on auto tuning).

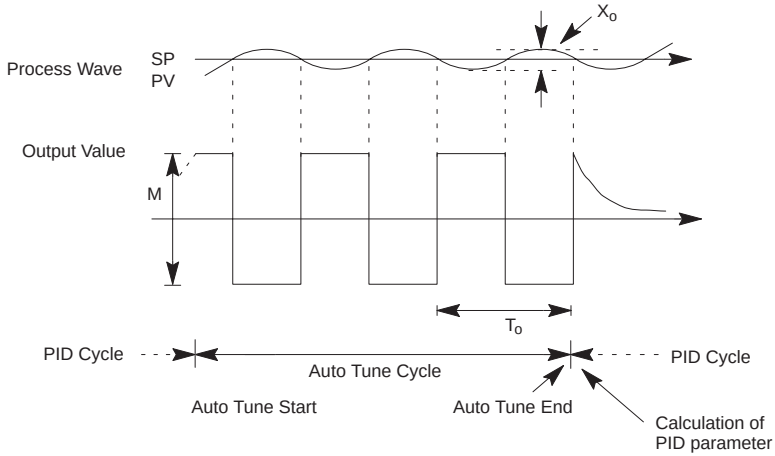
Auto Tuning error: If the auto tune error bit (bit 13 of Loop Mode and Alarm status word addr+06) is on, please verify the PV and SP values are within 5% of full scale difference, as required by the auto tune function. The bit will also turn on if the closed-loop method is in use, and the output goes to the limits of the range.

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 (or 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.



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

* This example is direct–acting. When set at reverse–acting, output is 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 Ziegler-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 and Alarm status word addr+06) is on, please verify the PV and SP values are within 5% of full scale difference, as required by the auto tune function. The bit will also turn on if the closed-loop method is in use, and the output goes to the limits of the range.

***NOTE**: If your PV fluctuates rapidly, you probably need to use the built-in analog filter (see page 8–45) or create a filter in ladder logic (see example on page 8–46).*

Tuning Cascaded Loops

In tuning cascaded loops, we will need to de-couple the cascade relationship and tune the loops individually, using one of the loop tuning procedures previously covered.

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.

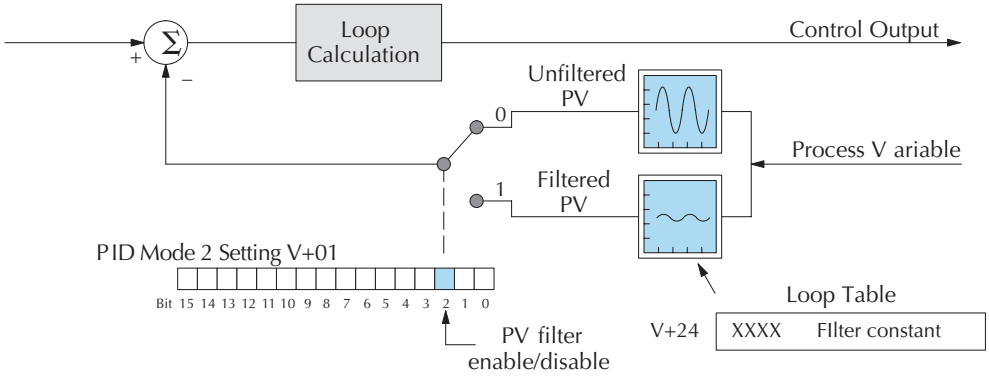
5. Tune the major loop, following the standard loop tuning procedure in this section. The response of the major loop PV is actually the overall response of the cascaded loops together.

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 DL06's built-in filter. The second method achieves a similar result using ladder logic.

The DL06 Built-in Analog Filter

The DL06 provides a selectable first-order low-pass PV input filter which can be particularly helpful during auto tuning, using the closed-loop method. Shown in the figure below, **we strongly recommend the use of a filter during auto tuning.** 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 addr+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***SOFT32 converts values above the valid range to 001.0 and values below this range to 000.1
- A setting of 000.0 or 001.1 to 999.9 essentially disables the filter.
- 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***SOFT32 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, integral, and derivative gain values are automatically updated in loop table locations addr+10, addr+11, and addr+12 respectively. The sample time in addr+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:

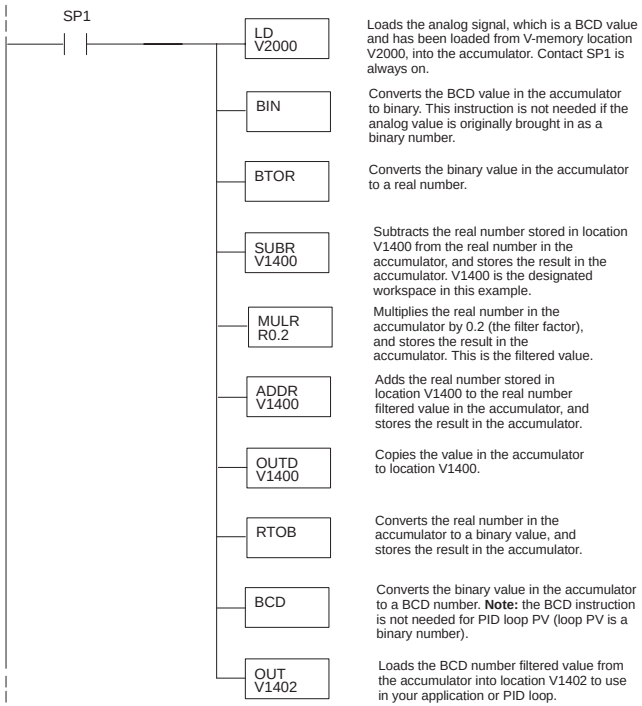
$$yi = k\,(xi - yi{-}1) + yi{-}1$$

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

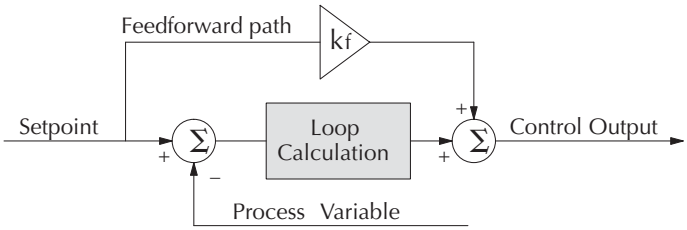
A similar algorithm can be built in your ladder program. Your analog inputs can be filtered effectively using either method. The following programming example describes the ladder logic you will need. Be sure to change the example memory locations to those that fit your application.

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.



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 DL06. 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.

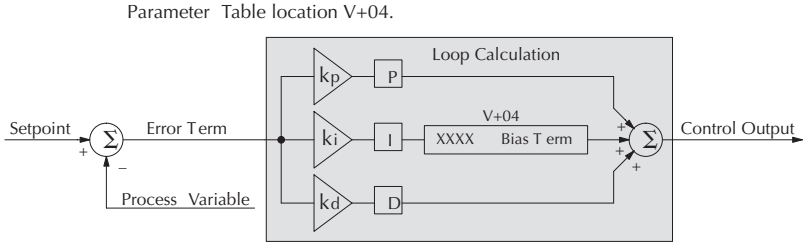


In the previous section on the bias term, we said that "the bias term value establishes a "working region" or operating point for the control output. *When the error fluctuates around its zero point, the output fluctuates around the bias value.*" Now, when there is a change in setpoint, an error is generated 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 DL06 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 addr+04.

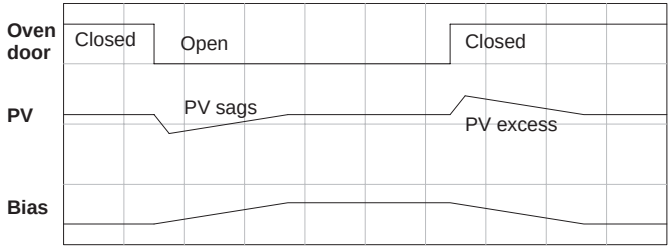


To change the bias (operating point), ladder logic only has to write the desired value to addr+04. The PID loop calculation first reads the bias value from addr+04 and modifies the value based on the current integrator calculation. Then it writes the result back to location addr+04. This arrangement 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 (the example below shows you how).

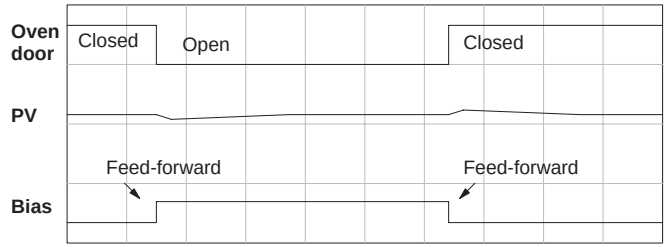
***NOTE**: When writing the bias term, one must be careful to design ladder logic to write the value only once, at the moment when the new bias operating point is to occur. If ladder logic writes the bias value on every scan, the loop's integrator is effectively disabled.*

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 addr+04, adds the desired change amount, and writes it back to addr+04. When the door closes, we duplicate the procedure, but subtracting desired change amount instead. The following figure shows the results.



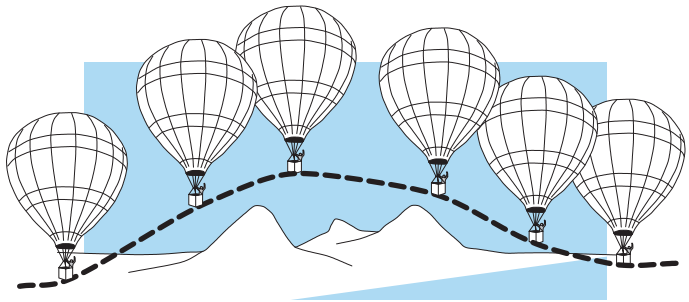
The step changes in the bias are the result of our two feed-forward writes to the bias term. We can see the PV variations are greatly reduced. The same technique may be applied for changes in setpoint.

Time-Proportioning Control

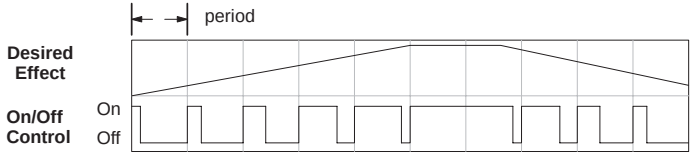
The PID loop controller in the DL06 CPU generates a smooth control output signal across a numerical range. The control output value is suitable to drive an analog output module, which connects to the process. In the process control field, this is called *continuous control*, because the output is on (at some level) continuously.

While continuous control can be smooth and robust, the cost of the loop components (such as actuators, heater amplifiers) can be expensive. A simpler form of control is called *time-proportioning control.* This method uses actuators 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*.



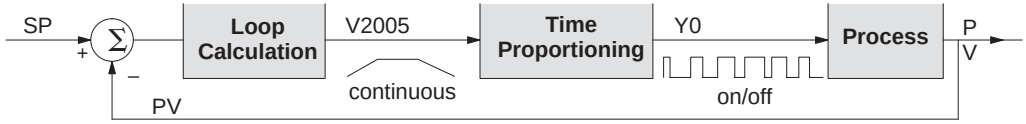
Time-proportioning control approximates continuous control by virtue of its duty-cycle – the ratio of ON time to OFF time. The following figure shows an example of how duty cycle approximates a continuous level when it is averaged by a large process mass.



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.



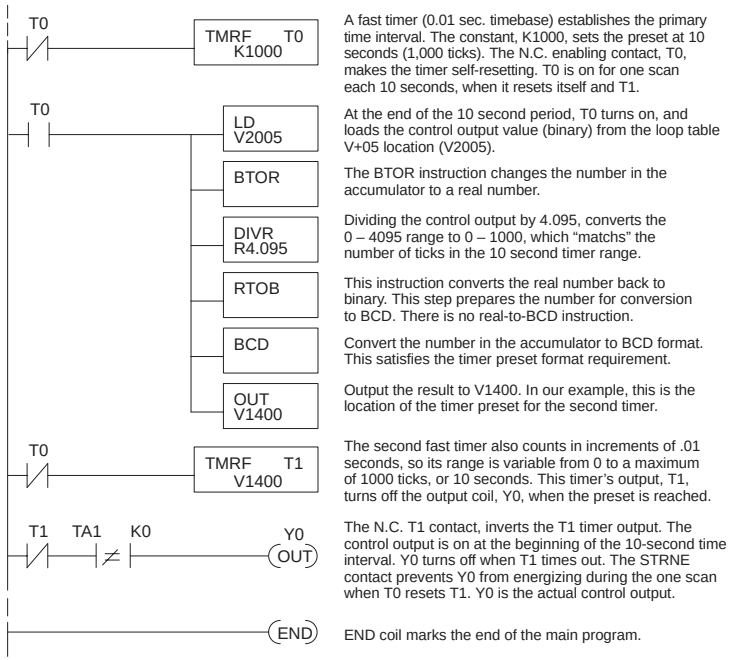
The example program uses two timers to generate On/Off control. It makes the following **assumptions**, which you can alter to fit your application:

- 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) or 0-4,095).
- The On/Off control output is Y0.

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

8

***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.*



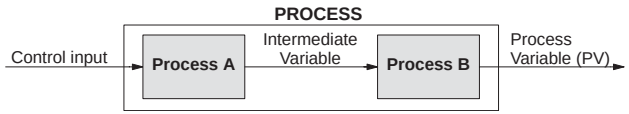
Cascade Control

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 DL06 also provides Cascaded Mode.

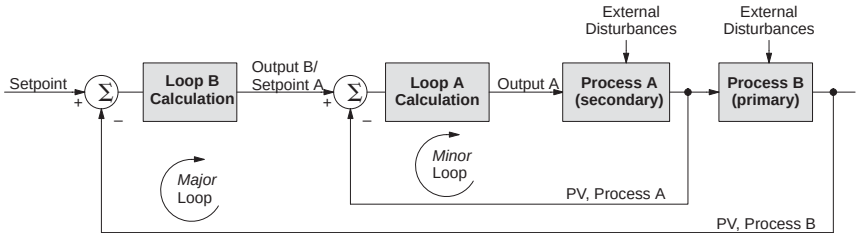
***NOTE**: Cascaded loops are an advanced process control technique. Therefore we recommend their use only for experienced process control engineers.*

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. 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.

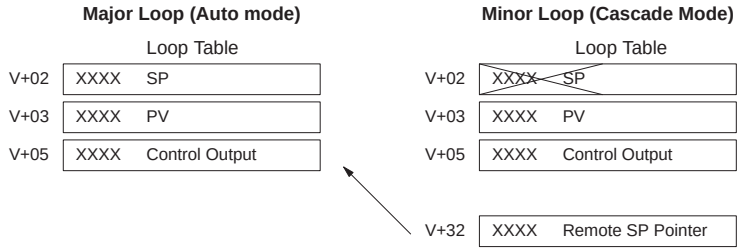
Cascaded Loops in the DL06 CPU

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.*

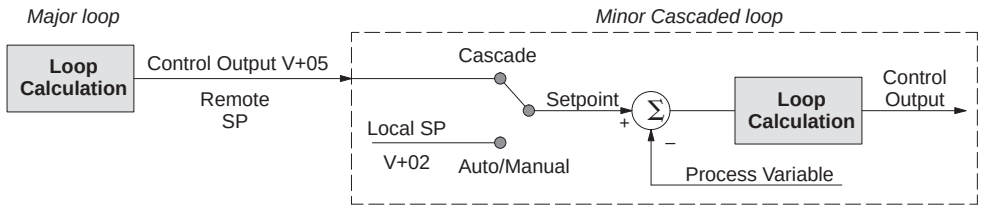
You can cascade together as many loops as necessary on the DL06, and you may have multiple groups of cascaded loops. For proper operation on cascaded loops you must use the same data range (12/15 bit) and unipolar/bipolar settings on the major and minor loop.

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 addr+32, as shown below. The pointer must be the address of the addr+05 location (control output) of the major loop. In Cascade Mode, the minor loop will ignore the its local SP register (addr+02), and read the major loop's control output as its SP instead.



When using *Direct*SOFT32's PID View to watch the SP value of the minor loop, *Direct*SOFT32 automatically reads the major loop's control output and displays it for the minor loop's SP. The minor loop's normal SP location, addr+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.

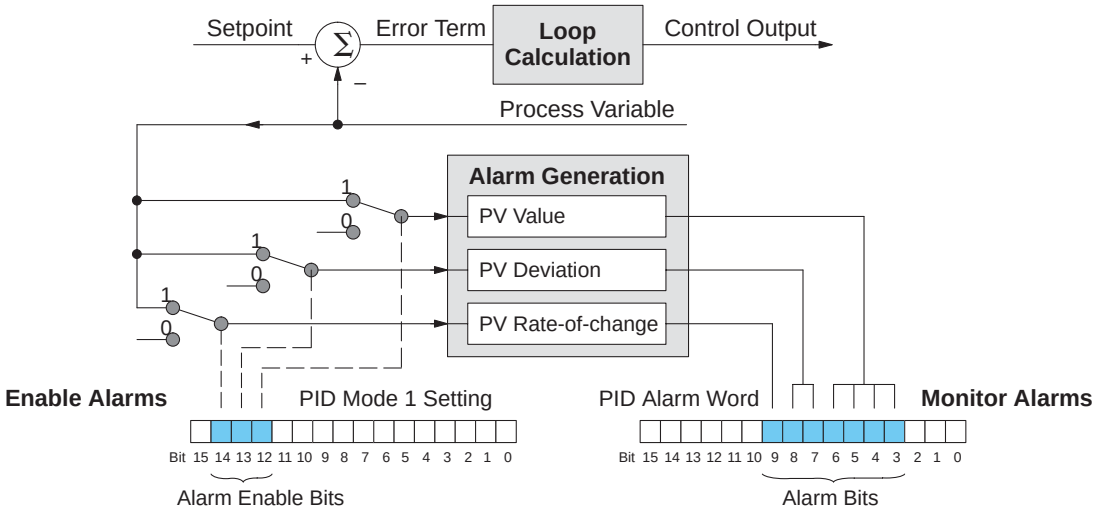
Process Alarms

The performance of a process control loop may be generally 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 DL06 CPU has a sophisticated set of alarm features for each loop:

- **PV Absolute Value Alarms** – monitors the PV with respect to two lower limit values and two upper limit values. It generates alarms whenever the PV goes outside these programmed limits.
- **PV Deviation Alarm** – monitors the PV value as compared to the SP. It alarms when the difference between the PV and SP exceed the programmed alarm value.
- **PV Rate-of-change Alarm** – computes the rate-of-change of the PV, and alarms if it exceeds the programmed alarm amount
- **Alarm Hysteresis** – works in conjunction with the absolute value and deviation alarms to eliminate alarm "chatter" near alarm thresholds.

The alarm thresholds are fully programmable, and each type of alarm may be independently enabled and monitored. The following diagram shows the PV monitoring function. Bits 12, 13, and 14 of PID Mode 1 Setting addr+00 word in the loop parameter table to enable/disable the alarms. ***Direct***SOFT32's PID View setup dialog screens allow easy programming, enabling, and monitoring of the alarms. Ladder logic may monitor the alarm status by examining bits 3 through 9 of PID Mode and alarm Status word addr+06 in the

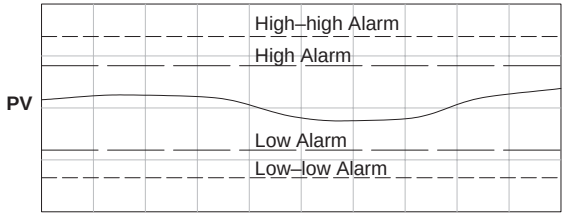


loop table.

Unlike the PID calculations, the alarms are always functioning any time the CPU is in Run Mode. The loop may be in Manual, Auto, or Cascade, and the alarms will be functioning if

PV Absolute Value Alarms

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.



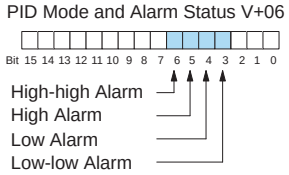
Loop Table

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

8

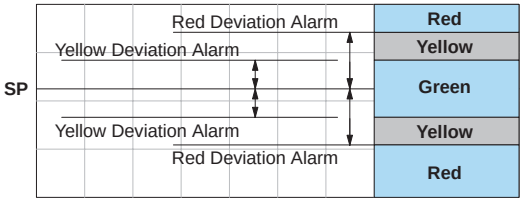
If the process remains out of control for some time, the PV will eventually cross one of the outer alarm thresholds, named High-high alarm and Low-low alarm. Their threshold values are programmed using the loop table registers listed above. A High-high or Low-low alarm indicates a serious condition exists, 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*SOFT32.



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 addr+17 and addr+20 as shown.

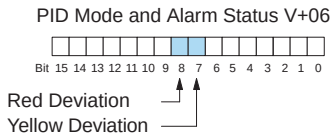


Loop Table

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*SOFT32.



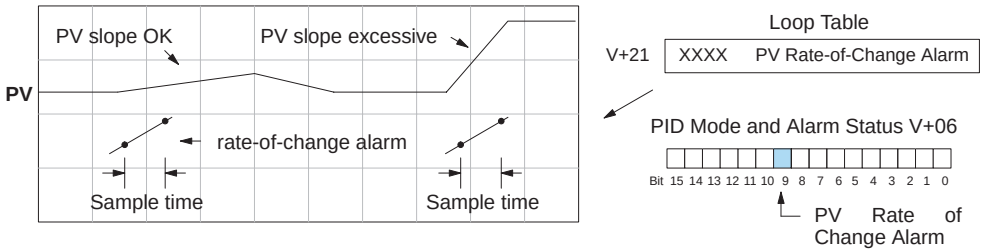
The PV Deviation Alarm can be independently enabled and disabled from the other PV alarms, using bit 13 of the PID Mode 1 Setting addr+00 word.

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.

PV Rate-of-Change Alarm

One powerful 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 DL06 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 addr+21.



As an example, suppose the PV is temperature for our process, and we want an alarm when the temperature changes faster than 15 degrees / minute. We must know PV counts per degree and the loop sample rate. Then, suppose the PV value (in addr+03 location) represents 10 counts per degree, and the loop sample rate is 2 seconds. We will use the formula below to convert our engineering units to counts / sample period:

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

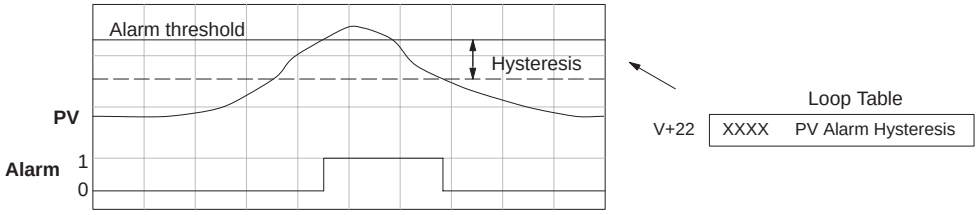
From the calculation result, we would program the value 5 in the loop table for the rate-of-change. The PV Rate-of-Change Alarm can be independently enabled and disabled from the other PV alarms, using bit 14 of the PID Mode 1 Setting addr+00 word.

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.

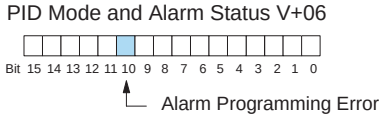
The PV Alarm Hysteresis amount is programmable from 1 to 200 (hex). When using the PV Deviation Alarm, the programmed hysteresis amount must be less than the programmed deviation amount. The figure below shows how the hysteresis is applied when the PV value goes past a threshold and descends back through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

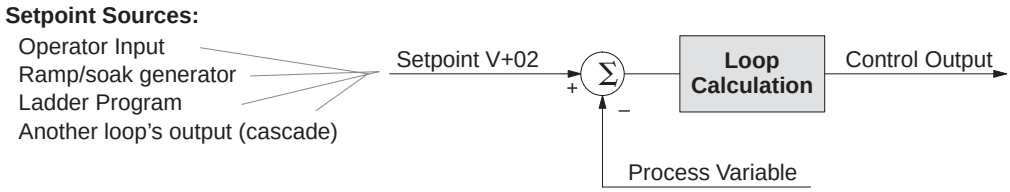


- PV Absolute Alarm value requirements:
  Low-low < Low < High < High-high
- PV Deviation Alarm requirements:
  Yellow < Red

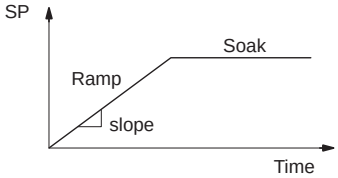
Ramp/Soak Generator

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 addr+02 at any particular time.



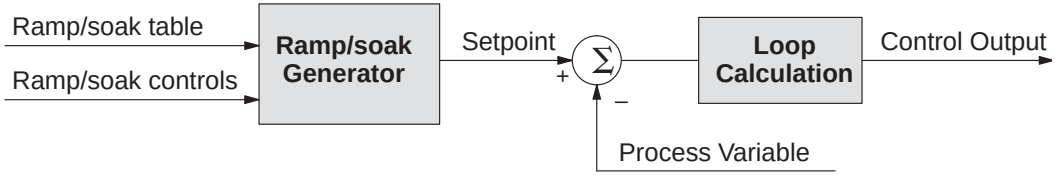
If the SP for your process rarely changes or can tolerate step changes, you probably will not need to use the ramp/soak generator. However, some processes require precisely-controlled SP value changes. *The ramp / soak generator can greatly reduce the amount of programming required for these applications.*



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 time. The soak time is also programmable in minutes.

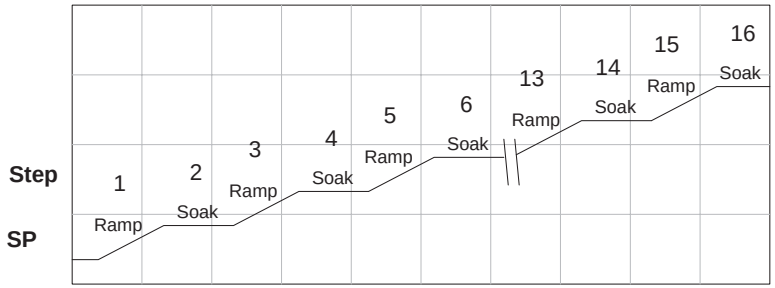
It is instructive to view the ramp/soak generator as a dedicated function to generate SP values, as shown below. It has two categories of inputs which determine the SP values generated. The ramp/soak table must be programmed in advance, containing the values that will define the ramp/soak profile. The 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.

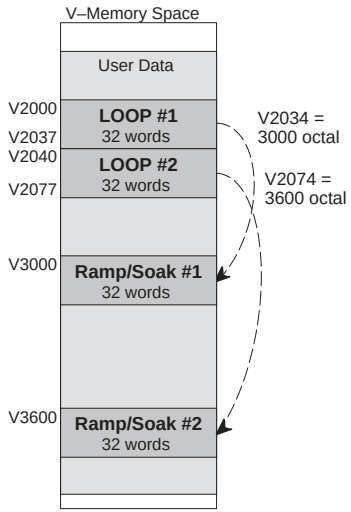
The following figure shows a SP profile consisting of ramp/soak segment pairs. The segments are individually numbered as steps from 1 to 16. The slope of each of the ramp may be either increasing or decreasing. The ramp/soak generator automatically knows whether to increase or decrease the SP based on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



Ramp/Soak Table

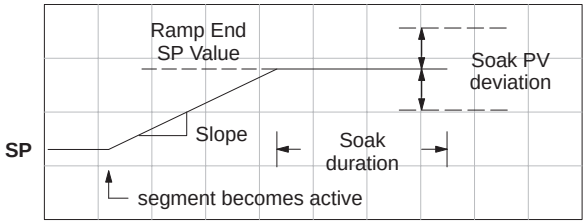
The parameters which define the ramp/soak profile for a loop are in a ramp/soak table. Each loop may have its own ramp/soak table, but it is optional. Recall the Loop Parameter table consists a 32-word block of memory for each loop, and together they occupy one contiguous memory area. However, the ramp/soak table for a loop is individually located, because it is optional for each loop. An address pointer in location addr+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*SOFT32, 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

8

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.

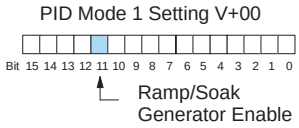
Ramp/Soak Table Flags

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

Ramp/Soak Generator Enable

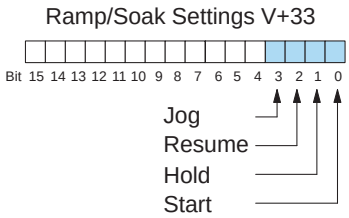
Bit	Ramp / Soak Flag Bit Description	Read/Write	Bit=0	Bit=1
0	Start Ramp / Soak Profile	write	–	0-1 Start
1	Hold Ramp / Soak Profile	write	–	0-1 Hold
2	Resume Ramp / soak Profile	write	–	0-1 Resume
3	Jog Ramp / Soak Profile	write	–	0-1 Jog
4	Ramp / Soak Profile Complete	read	–	Complete
5	PV Input Ramp / 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)	

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



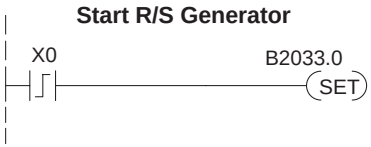
Ramp/Soak Controls

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*SOFT32 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.



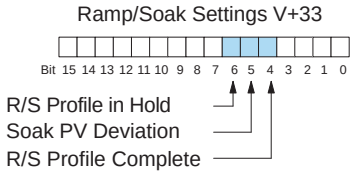
The normal state for the ramp/soak control bits is all zeros. Ladder logic must set only one control bit at a time.

- **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.

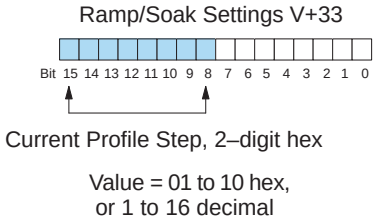
Ramp/Soak Profile Monitoring

You can monitor the Ramp/Soak profile status using other bits in the Ramp/Soak Settings addr+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 addr+33

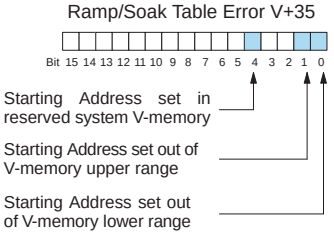


The number of the current step is available in the upper 8 bits of the Ramp/Soak Settings addr+33 word. The bits represent a 2-digit hex number, ranging from 1 to 10. Ladder logic can monitor these to synchronize other parts of the program with the ramp/soak profile. Load this word to the accumulator 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*SOFT32 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*SOFT32'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.

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 addr+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 DirectSOFT32, it displays 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 addr+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 addr+27 is too low.
- Check your ladder program to verify it is not writing to the SP location (addr+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 addr+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 (bumpless transfer feature).
- Check your ladder program to verify it is not writing to the SP location (addr+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 DirectSOFT32 work okay, but these values do not work properly when the ladder program writes the data.

A. The PID View in DirectSOFT32 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 in hex, so DirectSOFT32 converts them for you. The values in the table range from 0 to FFF, for 12-bit unipolar format.

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, 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.

Bibliography

Fundamentals of Process Control Theory, Third 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 Author: Gregory K. McMillan Publisher: Instrument Society of America ISBN 1–55617–483-7
Instrument Engineer's Handbook, Volume 2: Process Control, Third Edition Author (Editor-in-Chief): Bela G. Liptak Publisher: Chilton. . . . . ISBN 0–8019–8242–1	Instrument Engineer's Handbook, Volume 1: Process Measurement, Third Edition Author (Editor-in-Chief): Bela G. Liptak Publisher: Chilton. . . . . ISBN 0–8019–8197–2

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.

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.

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.

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 DL06'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.