Implementation of PID control in the Laboratory: Software and features.

You have learned about PID control and seen its potential ability to control the output of a process variable… If the control constants are chosen correctly, the variable approaches and maintains a desired value, its setpoint. This variable could be a temperature, a reactor conversion, the effluent conversion in a separation, etc. These controllers can be implemented mechanically, by hard-wired electronics or within the software of a computer. Your undergraduate laboratory uses Labview® to accomplish control. The purpose of this article is to help you understand how control is implemented in Labview or in a Basic software program.

PID Control:

PID (or PI) control starts by measuring the variable that is to be controlled. This is accomplished by an A to D (Analog to Digital) converter. This measures a voltage or current and outputs a numerical value to the computer corresponding to this measurement. These measurements are typically 12 to 16 bits and are measured many times per second.

Based in the control operation, a voltage is set by the use of a D to A (Digital to Analog) converter, This sets the voltage or current to an electronic component based on a variable chosen in software inside the computer program.

The basic equation which reflects PID control uses the difference between a measured voltage, V_D, and the setpoint , S_P, to set a desired voltage, V_{out}. Three constants are employed to reflect the relative amounts of proportional, integral or derivative control

$$V^{c}_{out} = C\left(P*(V_D\text{-}S_P) + \int_0^t I*(V_D\text{-}S_P)\,dt + D*\frac{d(V_D\text{-}S_P)}{dt}\right)$$

If we call the difference between the measured variable and the setpoint the error, E, this can be rewritten:

$$V^{c}_{out} = C\left(P*(E) + \int_0^t I*(E)\,dt + D*\frac{dE}{dt}\right)$$

Several other equivalent forms are often employed:

$$V^{c}_{out} = P'*(E) + \int_0^t I'*(E)\,dt + D' * \frac{dE}{dt} \sim K_c\left((E) + \int_0^t \frac{1}{\tau_I}*(E)\,dt + D' * \frac{dE}{dt}\right)$$

It is apparent that there are several parts of the equation that need to be evaluated before control can be performed. The operator specifies set point, S_P, and values of the controller constants: P, I and D. The computer must then evaluate the following:

The error, $E = V_D – S_P$,

The time difference between measurements, $dt = t - t_{old}$

The integral of the error with time = $\sum (E * dt) = (\sum E * dt)_{old} + E * dt$

The derivative of the error with time = $dE/dt = (E – E_{old})/dt$

These are then assembled to give the instantaneous value to set the Voltage out for this control:

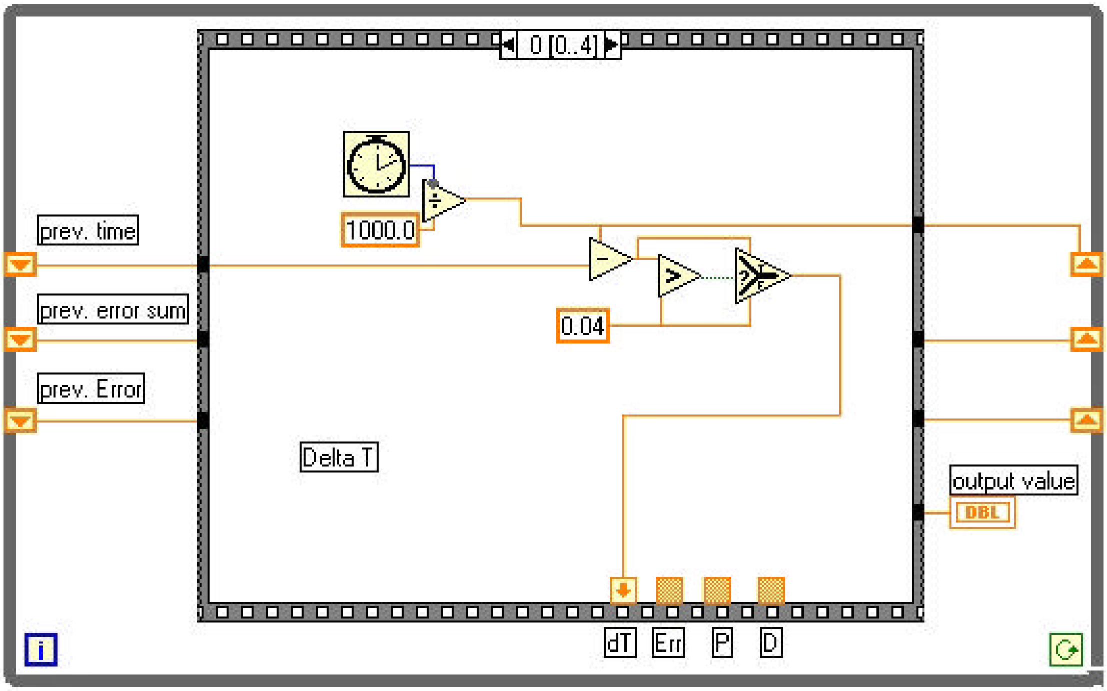
$$V_{out} = P*E + I*(\int (E * dt)) + D* dE/dt$$

The following is a series of computer code that can perform PID control setting an output variable [AOT(1), analog output channel 1] based on the difference between a measured temperature (TM) from an analog input [AIM(1)] and a desired temperature (set point, Tset). P. I and D constants are chosen (line 100) and are employed in the control loop (lines 200 to 300) which loop every second for 1000 seconds.

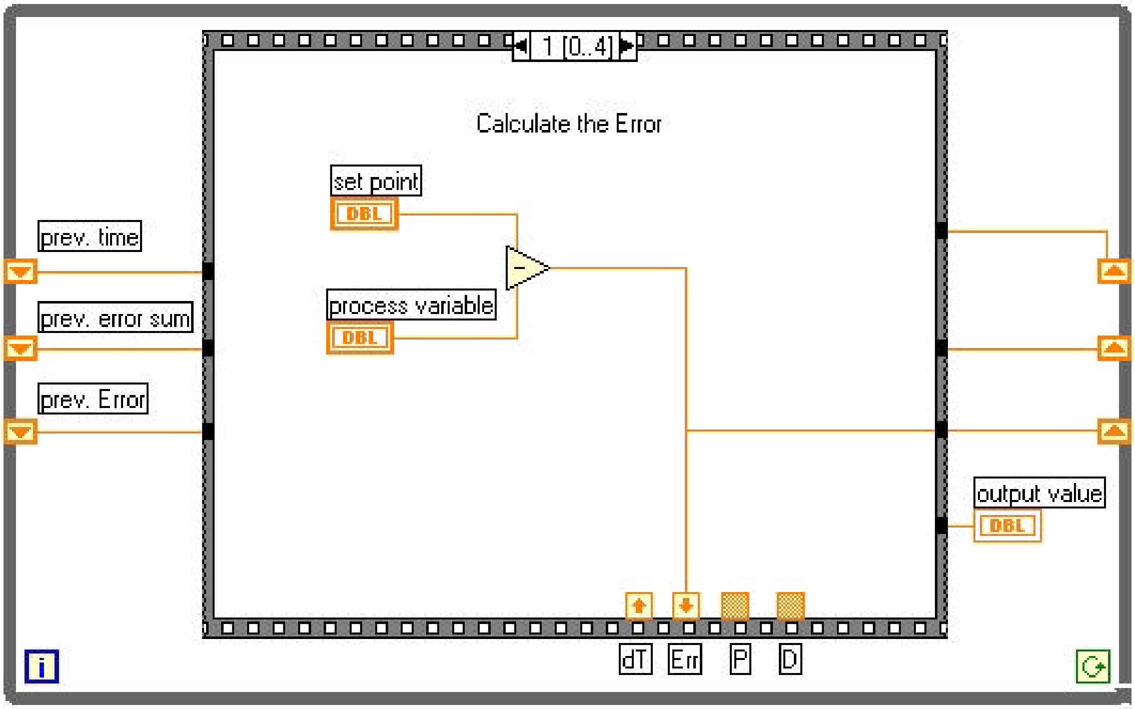
```
100 Input “What are the control constants (P, I and D)?” Pc, Ic, Dc @ Input control constants
110  Input “What is the desired temperature?” Tset  @ Input for the desired temperature

120 to = 0, INT = 0, Ero = 0          @ Set old time. Integral and error to zero

200  For I = 1 to 1000       @ Starting the control loop
210 TM=Ain(1)                @ Measuring the instantaneous Temperature
220  Er = TM – Tset          @ Calculating the error (difference between desired and measured T)
230 dEr = Er – Ero           @ Calculating change in error
240  tn = time               @ Reading the current time
250  Dt = tn - to            @ Calculating delta time, Dt
260 INT = Er*Dt + INT        @ Updating the integral
270 AOT(1)= P*Er + I*INT + D*(dEr/Dt)        @setting the voltage output
280 Ero = Er, to = tn        @ Now define “old” error and time
290 wait 1                   @ Waiting for one second (can be selected)
300 next  I                  @ control loop return
```

The Lab view version of these elements are shown on the attached figures:

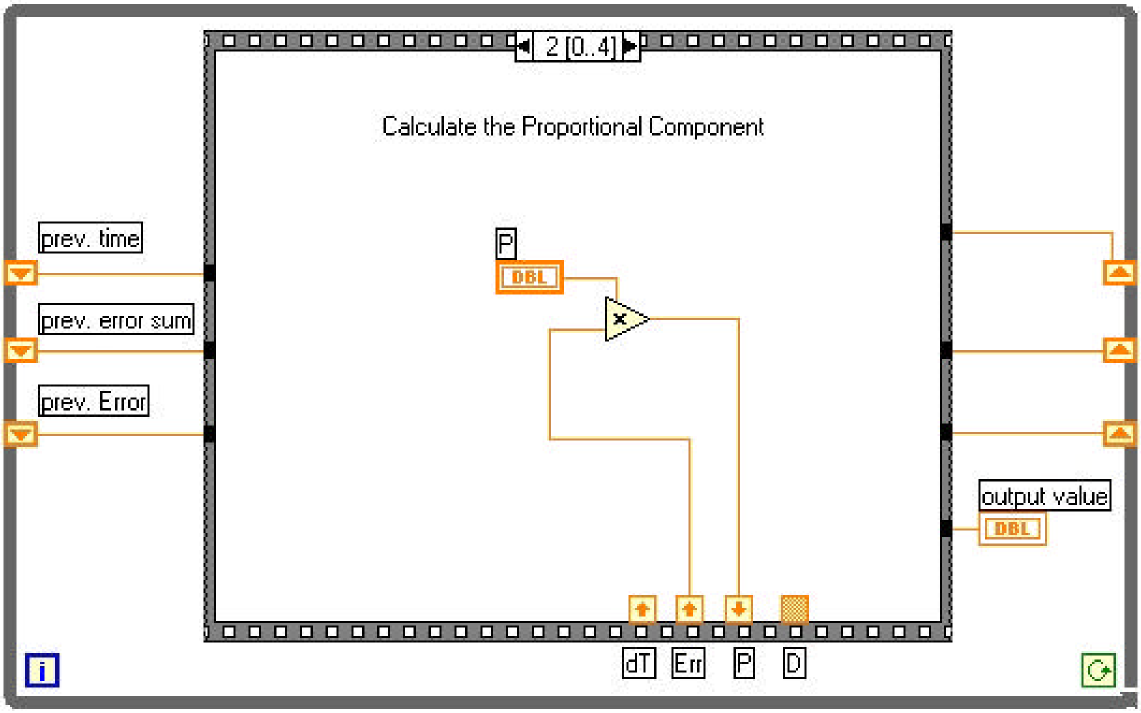
Delta t is measured/calculated:

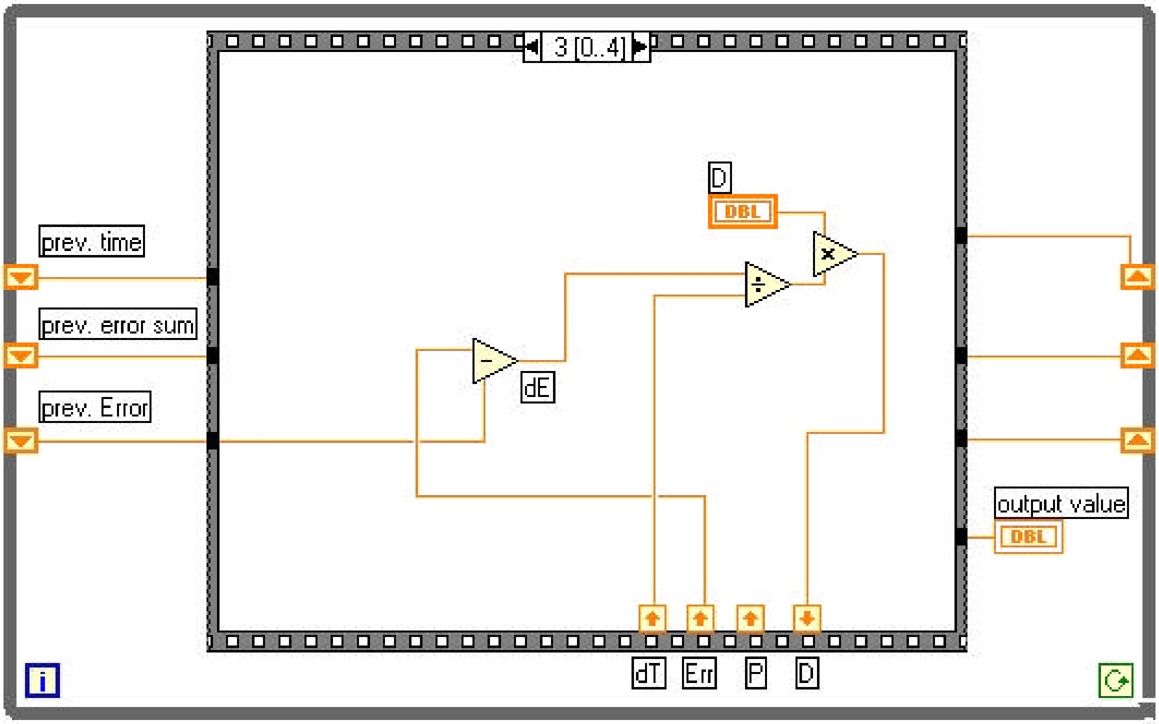


The Error is calculated from the measured variable and set point.

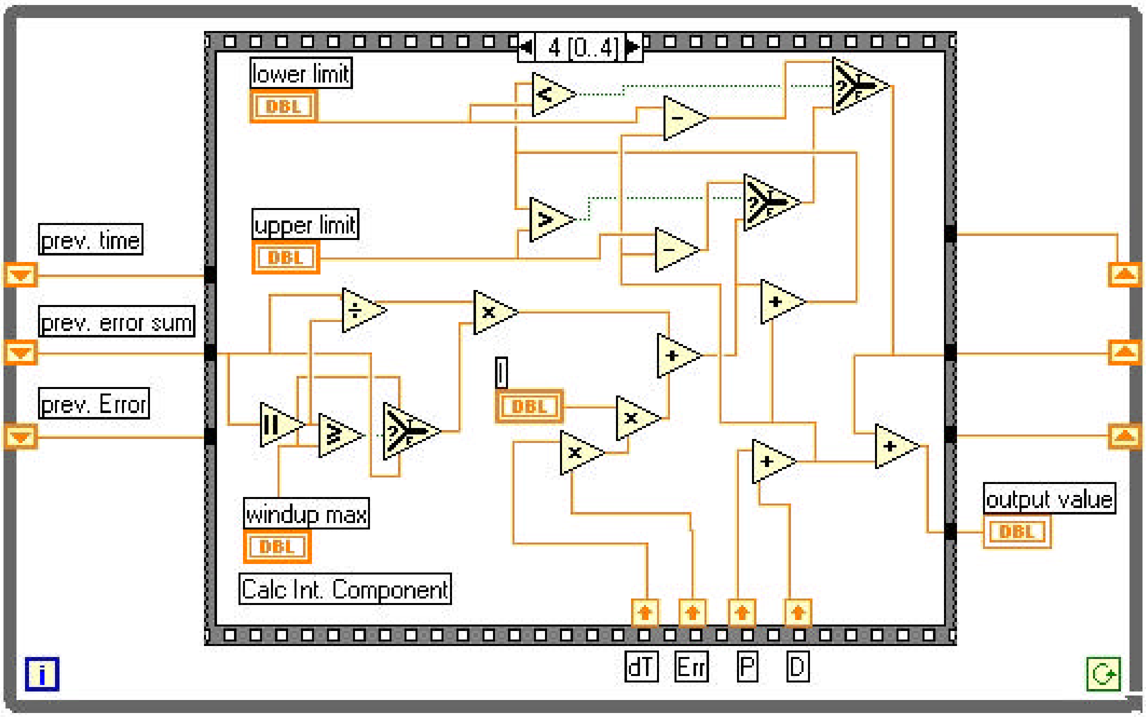


The Proportional Part of the control comes from P, the constant, times the error.





The derivative portion of the control comes from the derivative constant D timed d(error)/dt



Finally the integral portion of the control is added to the P and D portions of the control. These are then sent out to set the output variable. You will, however, notice that there is some protection that limits the integral control portion and that defines the limits of the output variable. Why are these needed?