**PID LOOP 2007 Application Note**
**APP- 007**

**2007 PID LOOP CONTROL**

This application note will describe how to implement and tune the new XT PID 2007. The new PID loop can run in either a Cooling or Heat Mode.

**Theory**

PID Control works by making adjustments to the output at a constant rate called the update rate (usually 2-6 seconds). For every update that occurs, PID Control takes a reading from the input sensor , measures the distance between the input and the setpoint (error), makes a series of calculation, and adjusts the output percentage in such a way as to move the input towards the setpoint in the most efficient manner. The “calculations” that determine the new value of the output after each update are made by three different modes of control: Proportional (P) mode, Integral (I) mode, and Derivative (D) mode. Each mode of control makes its own adjustment to the output percentage, and the three adjustments are added to the previous output percentage to determine the new output percentage.

NEW OUT % = OLD OUT % + P + I + D

| | | |
|---|---|---|
| P | Proportion Mode | Tries to stop the error from changing. Measures difference between current and previous error, and adjusts output percentage to prevent any further movement. |
| I | Integral Mode | Tries to bring the error to zero ( input = setpoint). |
| D | Derivative Mode | Tries to slow or stop a rapidly changing error so P and I Modes may effectively work to eliminate it. |

**Proportional Constant Kp**

The Proportional constant is simply a multiplier that can be used to fine tune the size of the Proportional Mode adjustment. Raising the value of Kp results in a greater reaction to input value changes, while lowering it results in a smaller reaction.

**The P setting will only affect the output when it sees a change of input from the previous update. If the plen temperature does not change from one update to the next, the**

**P will have no effect on the output. This is completely different than the old PID. The P is always trying to stop change and create a steady state condition. If you are above SP but the Plen temp is dropping quickly, it will be the effect of the P that will actually cause the door to pulse close, even though you are above Setpoint.**

A good way to test the effect of P is to set the I and D to zero. Then you can watch the temperature change and see the exact amount that the output changes. Changing P at this point will have a significant effect. The key is to find the right balance between the P, I , and D. Caution, to high of a P, will cause the system to take large steps and be unstable. P does not care if you are above Setpoint or below, it can produce a positive or negative change in either case. It just wants to stop the temperature from changing. If you use P only with out I, you will eventually find a temperature away from Setpoint that will just flat line.

Proportional change = Kp * (current error - previous error)

This is a highly simplified version.

The error is defined as the difference between the plenum temp and the Setpoint.

**Integral Constant Ki**

The I value is what will cause the output to change when the plenum temp is steady state and not changing. For example if the plenum temp is sitting .5 degrees above or below and the output is not changing, you should increase I. The I will always produce a positive change when above the Setpoint and a negative change when below the Setpoint. Start with a small I and increase it for more action.

Integral change = Ki * (current error)

**Derivative Constant Kd**

The Derivative Mode constantly analyzes the rate of change of the error, makes a prediction about what the future error will be, and makes an adjustment to the output in an attempt to reduce the rate of change in the error. In a lot of systems, D is not used and is set to 0. The CR-110 and ER-110 both use no D. The D can be slowly added, is primarily responsible for the initial start up of the system, and the first approach to Setpoint from the output starting at 0. To little D and the plenum temperature may overshoot the Setpoint, to much D and the system can become very unstable.

Derivative change = Kd * (Error - (2 * previous error / time) + ( previous previous error / time)

For documentation purposes I have included the actual formula that calculates the PID.

The Correct value that is added or subtracted from the output is:

Correction = [Ki * U_val] / Ki_div +[Kp + 10 * Kd / U_val] * error - [Kp + 20 * Kd / U_val] * previous error + [10 * Kd / U_val] * previous previous error

**CURRENT PID VALUES BEING TESTED:**

| FRESH AIR DOORS | | REFRIGERATION | |
|---|---|---|---|
| FILTER | 6 | | |
| KP_P | 50 | KP_I | 20 |
| KI_P | 8 | KI_I | 4 |
| KD_P | 20 | KD_I | 10 |
| UPDATE_TP | 150 | UPDATE_TI | 100 |
| I_MULT_P | 0 | I_MULT_I | 3 |
| D_MULT_P | 50 | D_MULT_I | 50 |
| MIN_PW | 150 | | |

Note that the update times for both the fresh air doors and refrigeration are quite large in tens of a sec. These values worked very well here in the office, but may need to be tweaked in the field.

You must be running P1 New PID V2.09 or greater for the new PID and to use these values.

**When programming, read the EE as normal, program P1 with the new software, then make the changes to the PID parameters.**