# EZRack PID Basics

Version 2.3

EZRack Editor V2.3

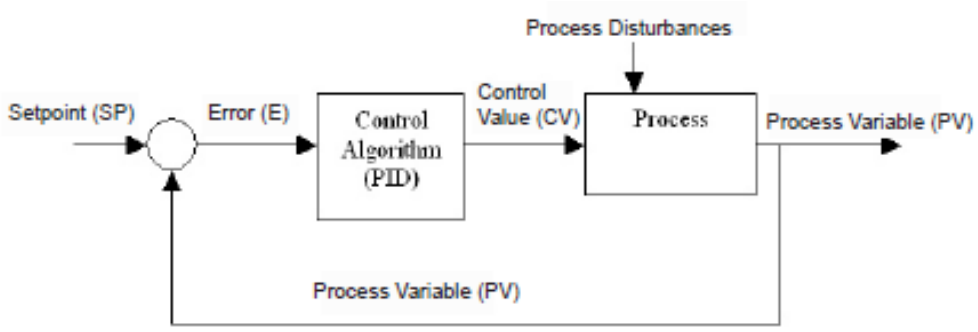
5/2019

## Contents

## Introduction to PID

Industrial Manufacturing Processes involve many variables such as temperature, pressure, flow, etc. It is important to control these variables for proper operation of the process.

There are several methods to control process variables. PID is one of the most popular control algorithms used in the industry. PID, as applied to Industrial Process Controls, stands for Proportional, Integral and Derivative control algorithm. The algorithm computes control action by using a mathematical equation which contains Proportional, Integral (Reset) and Derivative (Rate) terms. With proper choices of P, I, and D terms, a user can maintain a process value very close to the Setpoint. In addition, if the Setpoint or the process dynamics changes, the PID algorithm can bring the process back under control quickly. Selection of appropriate P, I and D coefficients is critical to the proper operation of the PID control. A block diagram of a generic process control is given below:




As shown in the figure, the user sets a target or Setpoint for the process. The system compares the actual Process Variable against the Setpoint and generates an Error value. The PID algorithm uses this error and computes a Control Variable as a function of the error. The computation function contains P, I, and D terms with user defined coefficients. The PID algorithm's goal is to minimize the error. If the Setpoint changes or the process is disturbed (resulting in a change in the Process Variable), a new error value is generated which results in a new Control Variable that should bring the Process Variable closer to the Setpoint.

## PID Terminology

Before we discuss more of the details involved with the PID Loop, you should have an understanding of some of the terms used in PID.

**Manufacturing Process** - A process that transforms a material's properties. The transformation may involve physical or chemical changes in the material. Examples of processes are: Steam Generation, Air conditioning, Milk Pasteurization, Oil refinement, etc.

**Process Variable (PV)** - Materials that have physical measurable properties, such as temperature, volume, viscosity, pressure, etc. A Process variable is a measurable physical property that we want to control. For example, in the air conditioning of a building, we want to control temperature, and therefore temperature is the Process Variable.

**Setpoint Value (SP) -** The target or desired value of the Process Variable. The purpose of PID loop is to maintain the Process Variable as close to the Setpoint as possible.

**Control Variable (CV)** - The Control Variable is calculated by a control algorithm. It depends on the error and PID coefficients.

**Error (E) -** Error equals the algebraic difference between the process variable and the setpoint. Magnitude and variation of the error depends on the process dynamics as well as on the PID coefficients. A well designed system will keep the error to a minimum value.

**External Disturbance** - Something that changes the equilibrium of the process. This results in a change in the control action to bring the process back into range. For example, in an air conditioned building, open doors and rainstorms are all changes that can affect the temperature.

## PID Algorithm used with EZRack PLC

EZRack PLC products support up to 8 PID loops. For each loop the user defines several parameters (such as Setpoint, Proportional, Integral (Reset) and Derivative (Rate) Gains, Limits, etc. (further discussed in the next section). You can change most of these parameters at run time using ladder logic by using the EZRack PLC Designer Pro software in online mode.

PID Algorithms used in EZRack PLC

The EZRack PLC uses the following algorithms for PID computations:

$SP_n$ = Setpoint at sample instance n
$PV_n$ = Process Variable at sample instance n
$CV_n$ = Control Variable at sample instance n
$K_p$ = Gain, Proportional term
$CV_0$ = Control Variable offset
$T_i$ = Reset (integral) time in seconds
$T_d$ = Derivative or React time, in seconds
$T_s$ = Sample time in seconds
$E_n$ = Error at sample instance n

The Error is computed as follows:

$E_n = PV_n - SP_n$ for Direct Acting

$E_n = SP_n - PV_n$ for Reverse Acting

Then the $CV_n$ is computed as follows:

**PID Position Algorithm:**

$$CV_n = K_p \times \left[ E_n + (T_s/T_i) \times \sum_{i=0}^{n} E_i + (T_d/T_s) \times (E_n - E_{n-1}) \right] + CV_0$$

**PID Velocity Algorithm:**

$$CV_n = K_p \times \left[ E_n + (T_s/T_i) \times \sum_{i=0}^{n} E_i + (T_d/T_s) \times (PV_n - PV_{n-1}) \right] + CV_0$$

*Position and Velocity Algorithms effect the derivative portion of the equation so if using PI control the same equation is used for both Algorithms.*

**PI Position & Velocity Algorithm:**

$$CV_n = K_p \times \left[ E_n + (T_s/T_i) \times \sum_{i=0}^{n} E_i \right] + CV_0$$

*Note: There are options in the setup that will modify the CV computations. For example, the user can choose to use PV Square root instead of PV in error computations. Please see the PID Setup Settings section where these options are discussed.*

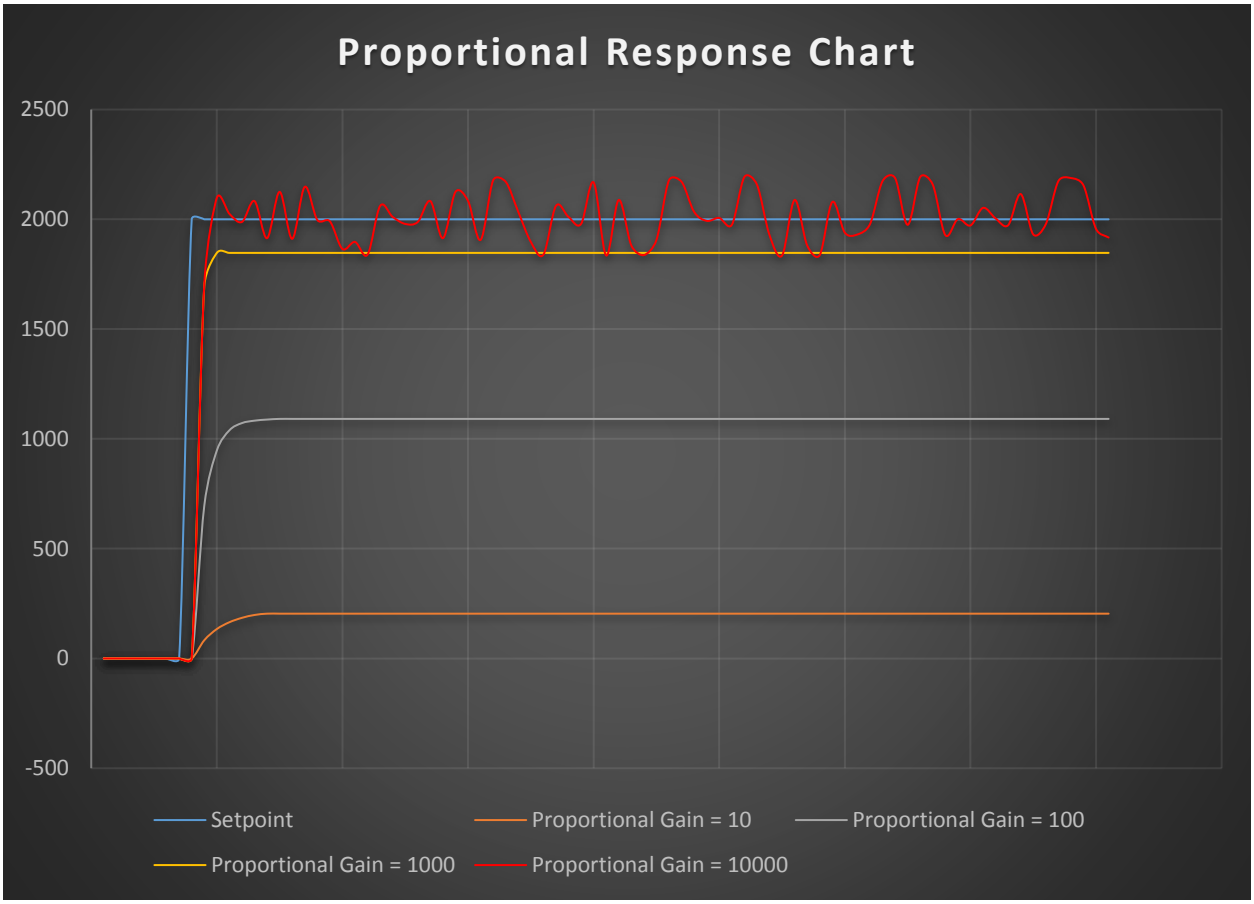
## PID Response on EZRack

This section will show how different variable effect the basic results. This section will show trends of effect not how they would behave in your application. These trends were developed with a simple and theoretical model and therefore should not be used as a basis for your application. Instead you should develop a starting point using the auto tune sequence (discussed in later section), and then use experimental results to finalize your PID settings.

### Proportional Gain ($K_p$)

The Proportional Gain effects how big the response to an error is and what the final offset from the setpoint will be. But the bigger the response the easier it is to make it unstable. This is why the Integral and Derivative are used, to stabilize and improve the response. The graph shown below shows a basic Proportional response to a step change at different Gain values.

$$CV_n = K_p \times \left[E_n + (T_s/T_i) \times \sum_{i=0}^{n} E_i + (T_d/T_s) \times (E_n - E_{n-1})\right] + CV_0$$



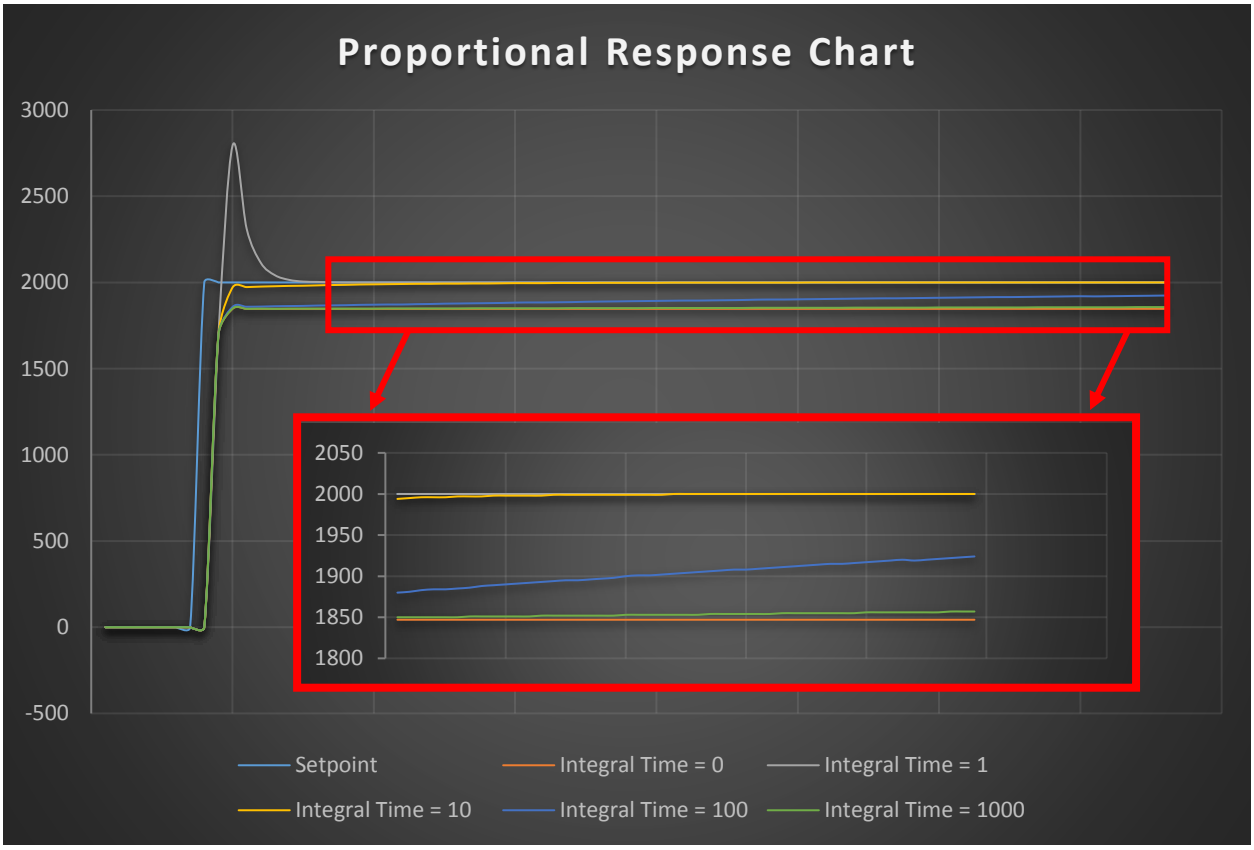

*This is done with theoretical models on the EZRack PLC (Sample Rate = 5, Integral = 0, Derivative = 0).*

### Integral (Reset) Time ($T_i$)

The Integral term increases action in relation not only to the error but also the time for which it has persisted. Therefore if the Proportional Gain does not bring the Error to zero this term will change Control Variable so that error decrease. This term will increase the longer the error exists. With only an "I" controller the initial response would be slow which is why PI controller is used in the example below. The Integral term has an Inverse Effect on CV, therefore the bigger the value the smaller the impact. You can use the Autotune to find a good starting point for your "I" value and then manually tune the response to your process. The sample rate also effects how fast this term is calculated.

$$CV_n = K_p \times \left[E_n + (T_s/T_i) \times \sum_{i=0}^{n} E_i + (T_d/T_s) \times (E_n - E_{n-1})\right] + CV_0$$

Please see Anti-Windup as a solution for overshoot in cases where the CV response saturates (goes to maximum value).




*This is done with theoretical models on the EZRack PLC (Sample Rate = 5, Prop = 1000, Derivative = 0).*

### Derivative (Rate) Time ($T_d$)

A Derivative term does not consider the error (meaning it cannot bring it to zero: a pure D controller cannot bring the system to its setpoint), but the rate of change of error, trying to bring this rate to zero. Usually derivative term is not needed in most process therefor you should first try and see if your process will work as PI control. The derivative term is used to reduce both overshoot and oscillation. It will dampen the effect of the response. The "D" term has direct effect so the bigger the term the greater the change based on error change. This term is directly changed based on the equation type (Position or Velocity).

Position Algorithm

$$CV_n = K_p \times \left[ E_n + (T_s/T_i) \times \sum_{i=0}^{n} E_i + (T_d/T_s) \times (E_n - E_{n-1}) \right] + CV_0$$

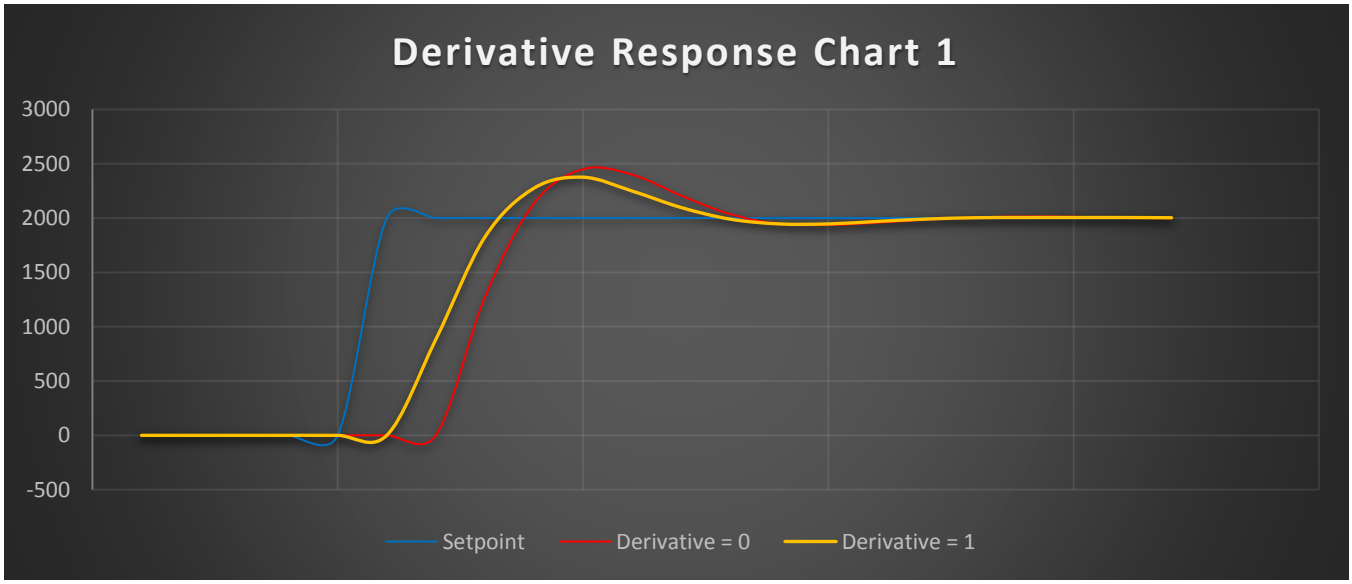
Velocity Algorithm

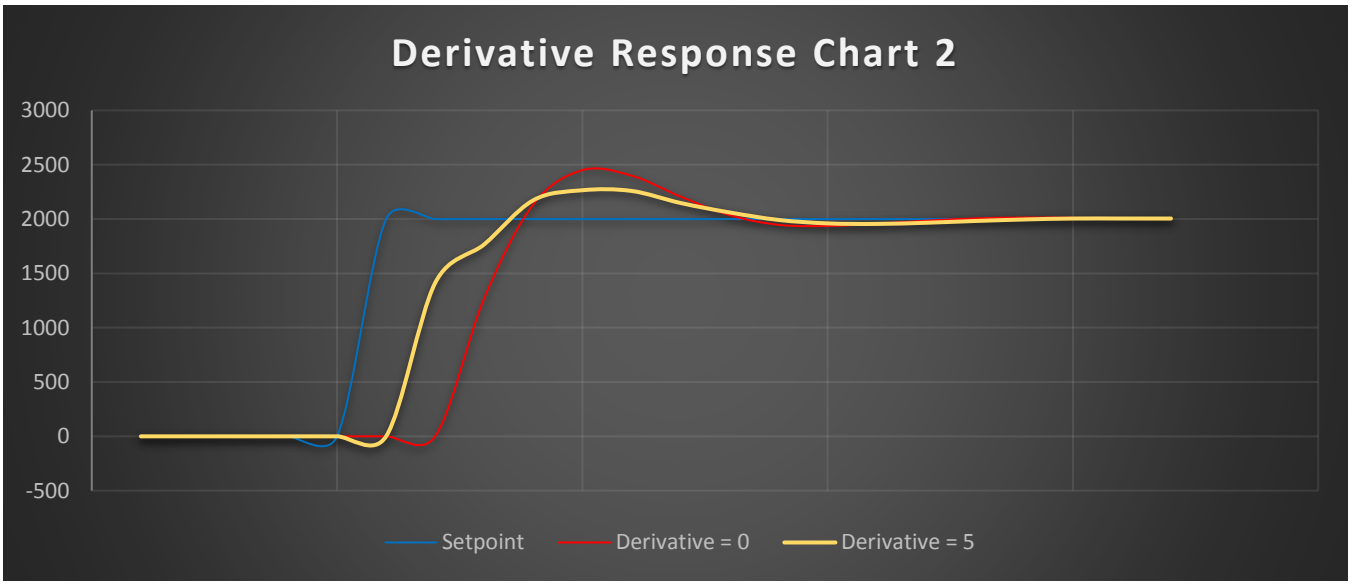
$$CV_n = K_p \times \left[ E_n + (T_s/T_i) \times \sum_{i=0}^{n} E_i + (T_d/T_s) \times (PV_n - PV_{n-1}) \right] + CV_0$$

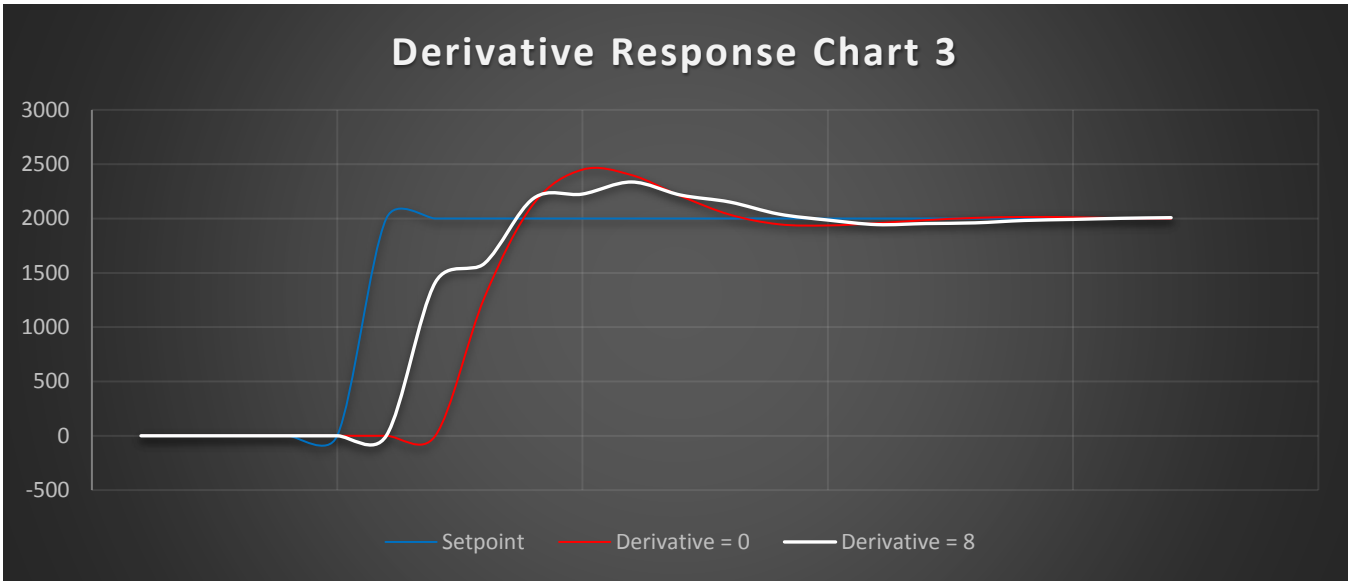
**Important Points:**

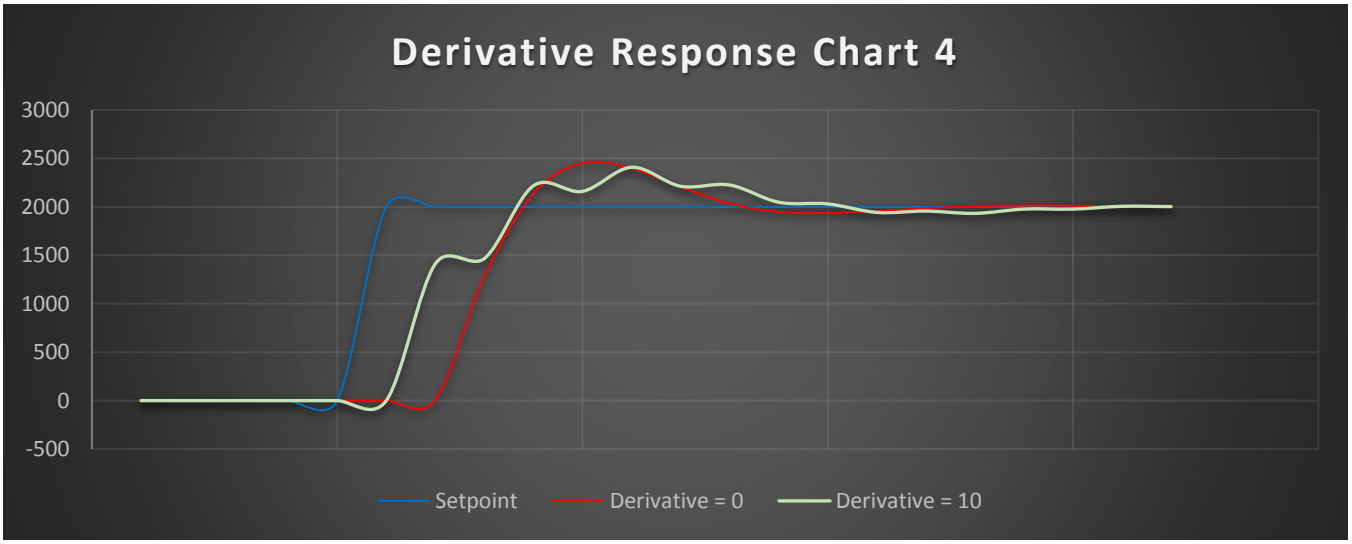
- **The Derivative allows for faster response (if this is not necessary for your application you should try the "PI" Controller).**
- **The PID Controller is much harder to tune then a PI therefore expect to have to spend more time tuning.**
- **The Derivative term can very easily cause the process to be unstable (See Chart 4 where response is pretty unstable)**

*All graphs are done with theoretical models on the EZRack PLC (Sample R. = 100, Prop = 50, Integ = 10).*




Derivative Response Chart 2
3000
2500
2000
1500
1000
500
0
-500
Setpoint
Derivative = 0
Derivative = 5


Derivative Response Chart 3
3000
2500
2000
1500
1000
500
0
-500
Setpoint
Derivative = 0
Derivative = 8


Derivative Response Chart 4
3000
2500
2000
1500
1000
500
0
-500
Setpoint
Derivative = 0
Derivative = 10

### Sample Rate ($T_s$)

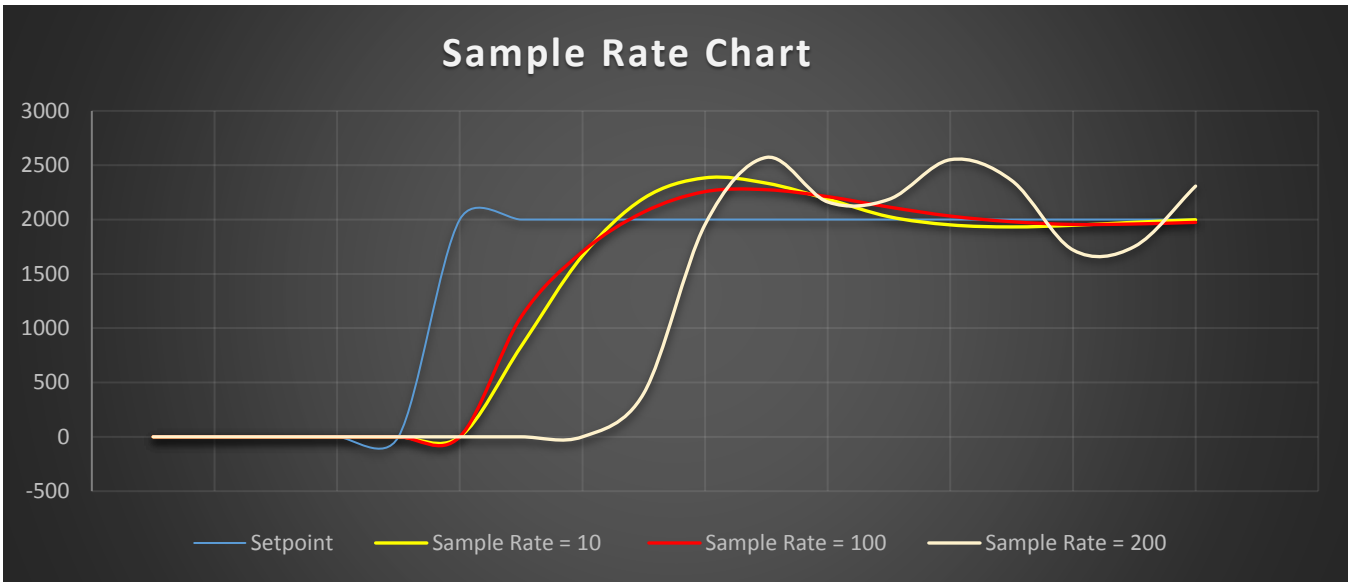
The Sample Rate is how often the PID loop will calculate the Control Variable. This term is included in the equation as well to compensate for the normal PID equation being continuous while this equation being discrete. Your sample rate should be such that any major change can be reacted to and not so fast you will be changing the CV too fast for your process to respond.

$$CV_n = K_p \times \left[E_n + (T_s/T_i) \times \sum_{i=0}^{n} E_i + (T_d/T_s) \times (E_n - E_{n-1})\right] + CV_0$$

**Important Points:**

- **The Sample Rate cannot be 0 because at that point the PID will not function.**
- **Do not use too big or too small of Sample Rate (in chart below Sample rate of 100 should be good)**
- **Base sample rate on known response of your setup or you can experiment and see where you get better response**

*Graph below is done with theoretical models on the EZRack PLC (Prop = 50, Integral = 10, Derivative = 6).*



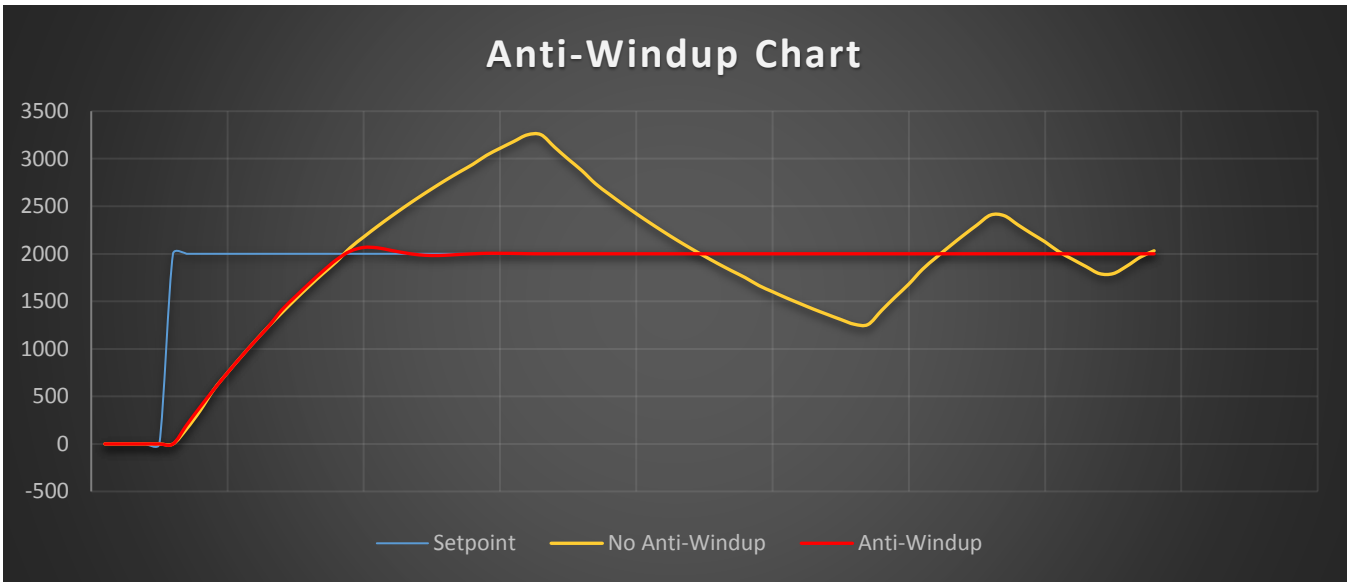
**Anti-Windup**

Anti-Windup works with the integral term to stop accumulating error when the response has saturated. The response has saturated when the CV High Limit is reached. At that point if the Anti-Windup is not enabled the integral term will continue to accumulate error and cause overshoot going in the other direction. With Anti-Windup errors will stop accumulating and hopefully decrease the oscillation caused by integral term.

The below term will stop adding Error when response is Saturated. The **Saturation Discrete** will also turn ON (Discrete Base + 3).

$$CV_n = K_p \times \left[E_n + (T_s/T_i) \times \sum_{i=0}^{n} E_i + (T_d/T_s) \times (E_n - E_{n-1})\right] + CV_0$$

*Graph below is done with theoretical models on the EZRack PLC (Sample Rate = 10, Prop = 1000, Integral = 1, Derivative = 0).*



### PV Square Root

If PV Square Root is enabled then the Process Variable will be square rooted before it is used for calculating error. This is mostly used if your process is non-linear, for example when using area calculations. The equations are modified as seen below.

$E_n = \sqrt{PV_n} - SP_n$ for Direct Acting
$E_n = SP_n - \sqrt{PV_n}$ for Reverse Acting

Then the $CV_n$ is computed as follows:

**PID Position Algorithm:**

$$CV_n = K_p \times \left[E_n + (T_s/T_i) \times \sum_{i=0}^{n} E_i + (T_d/T_s) \times (E_n - E_{n-1})\right] + CV_0$$

**PID Velocity Algorithm:**

$$CV_n = K_p \times \left[E_n + (T_s/T_i) \times \sum_{i=0}^{n} E_i + (T_d/T_s) \times (\sqrt{PV_n} - \sqrt{PV_{n-1}})\right] + CV_0$$

### Deadband

The Deadband is used if small changes are normal in your process and they would cause your PID loop to be unstable. Therefore a change must be bigger then your Deadband before the PID process will act.

PID Process will check following:

If ($E_n < Deadband$)
    Then (PID do nothing)
    Else (PID react)

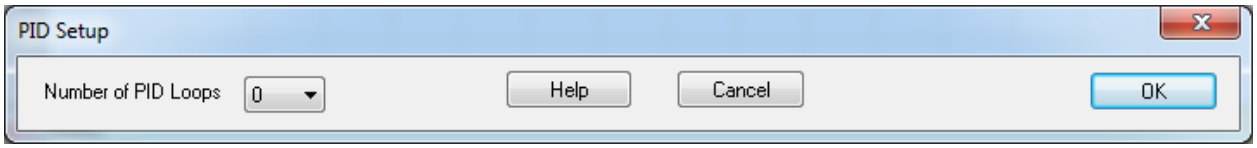
This table outlines if response if Deadband = 10 and Current Setpoint = 1000.

| Current Process Variable | Calculation | PID Response |
|---|---|---|
| **1005** | **1005 – 1000 < 10** | **No Response** |
| **1015** | **1015 – 1000 > 10** | **PID Correct CV to lower PV** |
| **993** | **1000 – 993 < 10** | **No Response** |
| **980** | **1000 – 980 > 10** | **PID Correct CV to raise PV** |

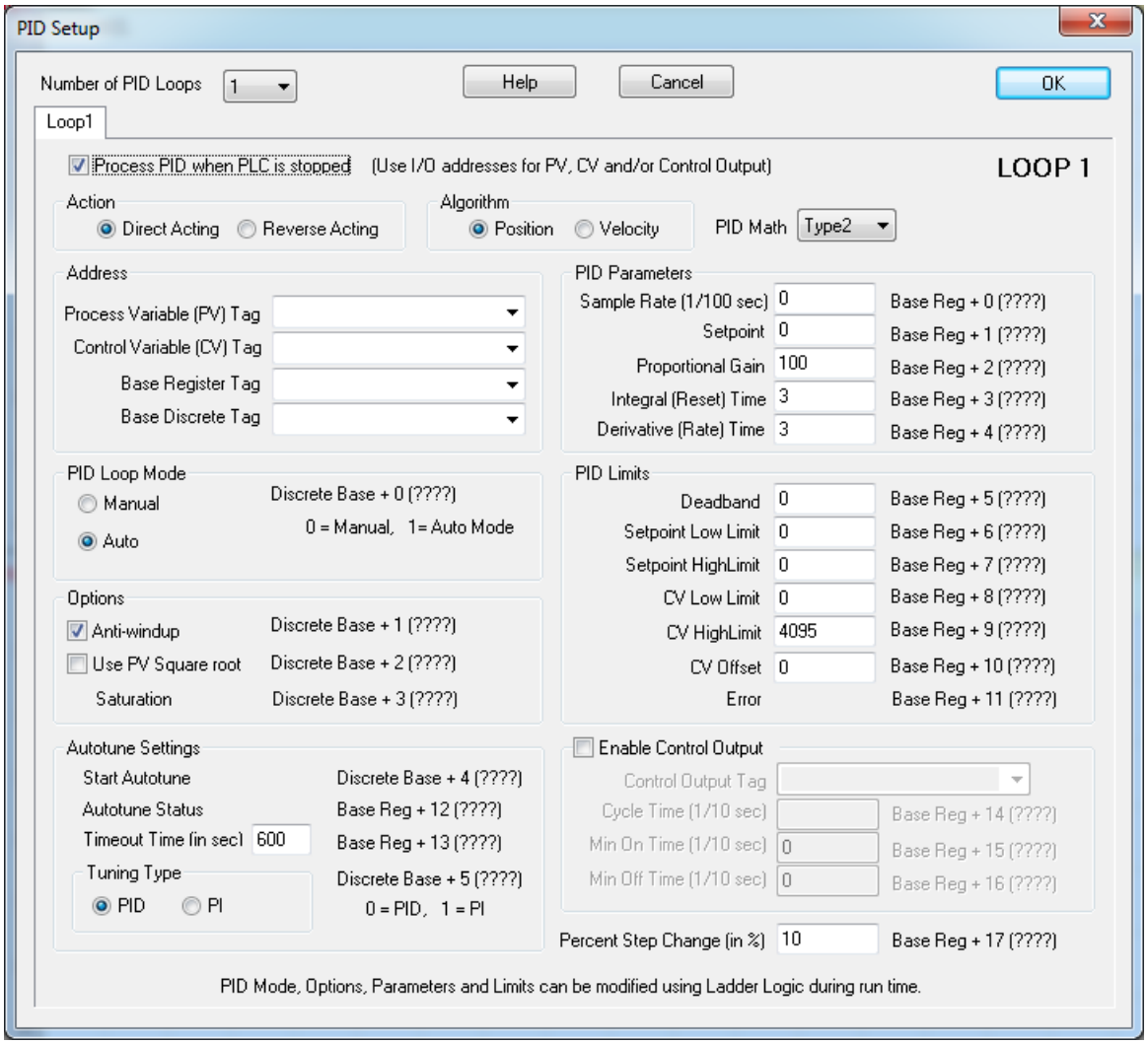
## PID Setup

The following section will explain how to setup a PID loop using your EZRack PLC Designer Pro software. To access the PID Setup, perform the following steps:

1. Select the **Setup Menu > PID Setup…** option or select the **PID Setup** from the left navigation bar. The following dialog box will appear (If you have already defined one or more loops, the image below will be different).




2. Use the drop-down arrow to select the Number of PID Loops you would like to use (you can select up to 8 PID Loops).
3. As soon as you select a number of loops other than 0, the following dialog box will appear:

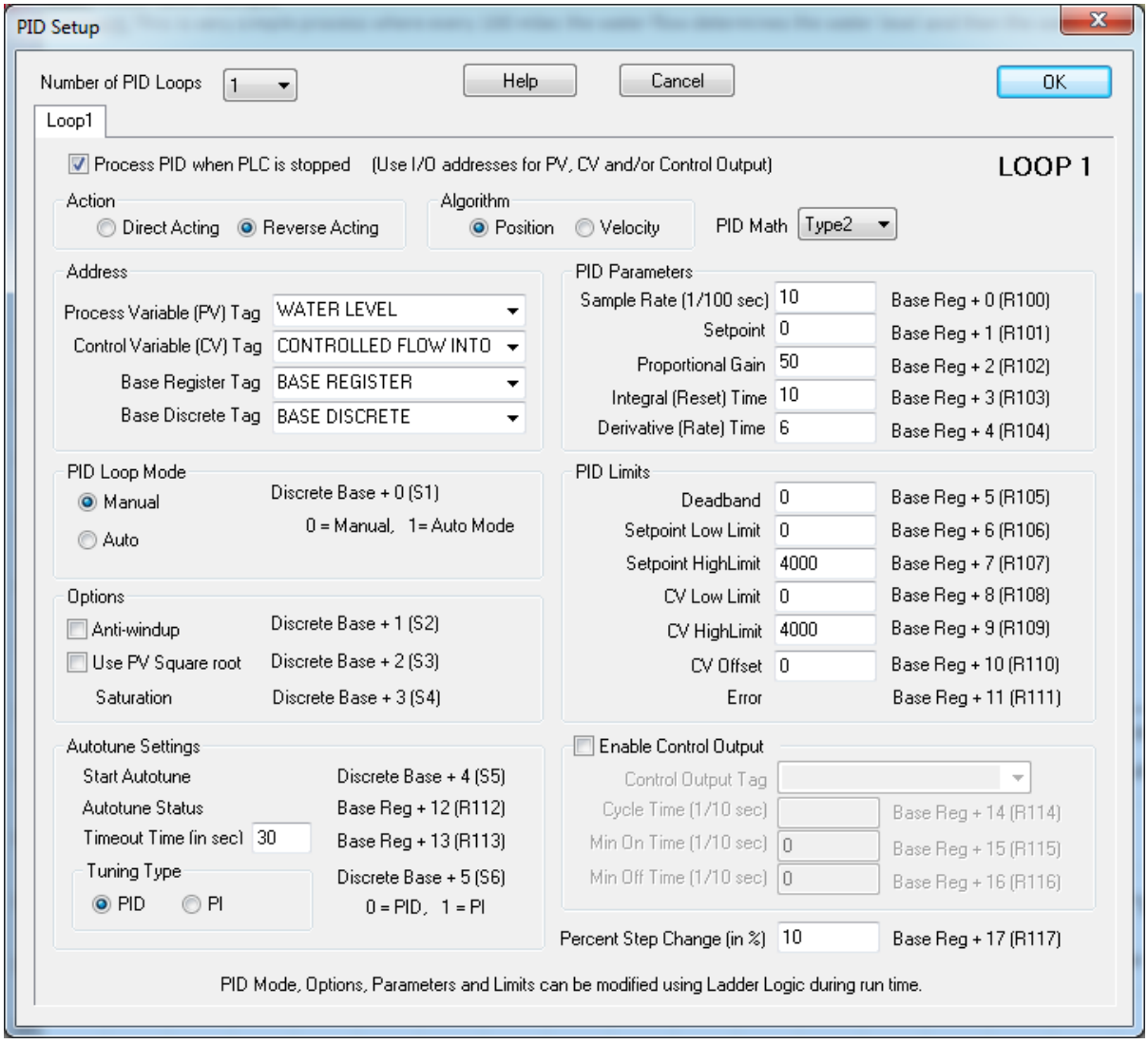


4. Next enter your Process Variable (PV) Tag and the Control Variable (CV) Tag. These can be either an Unsigned Int 16 or Unsigned Int 32. You can use outputs or input directly for these tags.

5. Next enter a Base starting Register and Base starting Discrete. These should be selected such that 32 registers and 8 discretes can be created for the rest of the PID control after them. For example if a tag exists at R10 then R1-R9 cannot be used. If you are not worried about register address location then the Automatic Addressing will select a good starting register.

6. Next fill out all the settings for your PID. These do not have to be your final settings but they should be settings such that an autotune can be run. The **next section** will go over all the options for all settings.

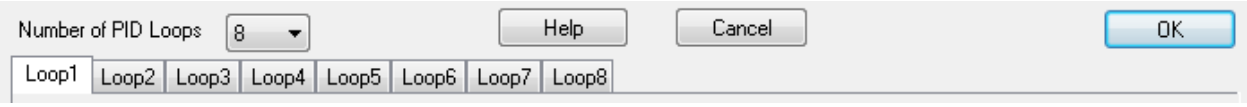
7. After all setting have been setup, your final PID will look similar to the image below. Where the base Register is R100 and the base Discrete is S1.



## PID Setup Settings

This section will go over all the possible selections and/or options in the PID setup.

### Number of Loops




The Dialog box above allows you to define all your PID Parameters. It will show as many tabs as the number of PID loops selected. The tabs are labeled Loop1, Loop2, and so on. Each PID loop requires a contiguous block of 32 Registers and a contiguous block of 8 discrete for storing parameters and status.

### Loop Parameters / Tags

Basic parameter information is outlined below. More details in sections afterwards.

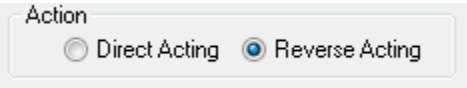
| Name | Address | Data Type | Memory | Permissions | Description |
|---|---|---|---|---|---|
| Process Variable (PV) | User Select | U16 / U32 | R, IR | PID Read Only, User R/W | User/Process controlled variable that should be moved to setpoint |
| Control Variable (CV) | User Select | U16 / U32 | R, OR | PID R/W, User R/W | PID controlled variable used to change PV |
| Base Register | User Select | U16 | R | See Sample Rate | Used to create all PID Register tags |
| Base Discrete | User Select | Discrete | S | See Loop Mode | Used to create all PID Discrete tags |
| Control Output | User Select | Discrete | S, O | PID R/W, User R/W | PID controlled discrete used with Cycle Time to change PV |
| **Registers** | | | | | |
| Sample Rate | Base R + 0 | U16 | R | PID Read Only, User R/W | Sample period (1/100 sec) |
| Setpoint | Base R + 1 | U16 | R | PID Read Only, User R/W | The value the PID is moving the PV to |
| Proportional Gain | Base R + 2 | U16 | R | PID Read Only, User R/W (Autotune R/W) | Proportional Gain Value the PID is using in Calculations |
| Integral Time | Base R + 3 | U16 | R | PID Read Only, User R/W (Autotune R/W) | Integral Time Value the PID is using in Calculations |
| Derivative Time | Base R + 4 | U16 | R | PID Read Only, User R/W (Autotune R/W) | Derivative Time Value the PID is using in Calculations |
| Deadband | Base R + 5 | U16 | R | PID Read Only, User R/W | How much PV need change before PID act |
| Setpoint Low Limit | Base R + 6 | U16 | R | PID Read Only, User R/W | Minimum Setpoint value |
| Setpoint High Limit | Base R + 7 | U16 | R | PID Read Only, User R/W | Maximum Setpoint value |
| CV Low Limit | Base R + 8 | U16 | R | PID Read Only, User R/W | Minimum CV value (PID will set CV to at least Min when running) |
| CV High Limit | Base R + 9 | U16 | R | PID Read Only, User R/W | Maximum CV value (PID will not exceed this value) |
| CV Offset | Base R + 10 | U16 | R | PID Read Only, User R/W | Constant Value added to CV at all times |
| Error | Base R + 11 | S16 | R | PID R/W, User Read Only | PID reported Error btw Setpoint and PV (updated when PID run) |
| Autotune Status | Base R + 12 | U16 | R | Autotune R/W, User Read Only | Used to report PID status when Autotune running |
| Timeout Time | Base R + 13 | U16 | R | PID Read Only, User R/W | PID Autotune timeout time |
| Cycle Time | Base R + 14 | U16 | R | PID Read Only, User R/W | Cycle time for Digital Output calculations |
| Min ON Time | Base R + 15 | U16 | R | PID Read Only, User R/W | Min ON time for Digital Output |
| Min OFF Time | Base R + 16 | U16 | R | PID Read Only, User R/W | Min OFF time for Digital Output |
| Percent Step Change | Base R + 17 | U16 | R | PID Read Only, User R/W | Percentages the CV will change when Autotune running |
| 18-32 | Base R + 18-32 | U16 | R | NA | Reserved for future use |
| **Discretes** | | | | | |
| Loop Mode | Base D + 0 | Discrete | S | PID Read Only, User R/W | Select whether PID will run or not |
| Anti-Windup | Base D + 1 | Discrete | S | PID Read Only, User R/W | Used to turn ON or OFF Anti-Windup |
| Use PV Square Root | Base D + 2 | Discrete | S | PID Read Only, User R/W | Used to turn ON or OFF whether $\sqrt{PV}$ is used |
| Saturation | Base D + 3 | Discrete | S | PID R/W, User Read Only | PID loop will indicate whether CV High or Low Limit reached |
| Start Autotune | Base D + 4 | Discrete | S | Autotune Read Only, User R/W | Used to turn ON the Autotune Sequence |
| Tuning Type | Base D + 5 | Discrete | S | Autotune Read Only, User R/W | Selects whether Autotune will calculate PI or PID values |
| 7-8 | Base D + 6-7 | Discrete | S | NA | Reserved for future use |

**Process PID when PLC is stopped**


When PLC is stopped (not in Run Mode), it does NOT process ladder logic or Update I/O. However in some cases, it may be desirable to continue the PID loop even when the PLC is stopped. Use this check box to indicate that the PID should be processed when the PLC is stopped. The default is to continue PID processing.

*Note: If you want the PID to be processed with PLC stopped, make sure to use physical addresses for the PV, and CV/Control Output tags. The reason is that when PLC is stopped no ladder executes, and ONLY PID related I/O would be updated if the "Process PID when PLC is stopped" is checked.*

**Controller Action**




To determine whether the Controller Action needs set as Direct Acting or Reverse Acting, it is helpful to understand the difference between "Process Action" and "Controller Action." First, ascertain what type of process (direct acting or reverse acting) best describes your system.

- Direct Acting **Process**: Control Variable and Process Variable follow the same direction i.e. Increase in Control Variable increases the Process Variable and vice versa.
    - For example: In a heating application, the more power through a heater (CV) increases the temperature (PV) or a decrease in the heater's output will result in a decrease in temperature.
- Reverse Acting **Process**: The CV and PV move in opposite direction such as an increase in CV decreases the PV or vice versa.
    - For example: In an air conditioning or cooling application, more power is applied to create a reduction or decrease in the temperature.

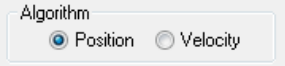
For simple systems, after identifying the type of process that describes your system, set the Controller Action as the opposite of the process. For instance, if your system utilizes a direct acting process, the Controller Action will need set as Reverse Acting. If your system utilizes a reverse acting process, the Controller Action will need set as Direct Acting. (See below for further explanation.)

- Controller Action = Reverse Acting: The controller monitors PV. As PV decreases, the Controller will increase CV and vice versa.
    - For example: In the heating application listed above (a direct acting process), the controller would need set as Reverse Acting in order to manage the temperature in relation to the setpoint. So that if the temperature decreases due to outside variables, the controller will increase the heater to restore the temperature back to desired level. Conversely, if the temperature increases due to outside variables, the controller will decrease the heating output to restore the temperature back to the established setpoint.

- Controller Action = Direct Acting: The controller monitors PV. As PV increases the controller will increase CV and vice versa.
    - In the example of the cooling application (or the reverse acting process), the controller would need set as Direct Acting. Therefore, if the temperature increases due to outside variables, the controller will increase the cooling power in order to restore the temperature to desired level. Conversely, if the temperature decreases due to outside variables, the controller will decrease the cooling output to restore the temperature back to the established setpoint.

EZRack PLC computes error term, based on this choice, as follows:
$E_n = PV_n - SP_n$ for Direct Acting
$E_n = SP_n - PV_n$ for Reverse Acting

**Algorithm (Position or Velocity)**




EZRack PLC supports two PID algorithms, known as Position and Velocity algorithms. Use this option to select whether you use the Position Algorithm or Velocity Algorithm. This is only useful if you are using PID, if using PI controller then this section does not matter.

**PID Position Algorithm:**

$$CV_n = K_p \times \left[E_n + (T_s/T_i) \times \sum_{i=0}^{n} E_i + (T_d/T_s) \times (E_n - E_{n-1})\right] + CV_0$$

**PID Velocity Algorithm:**

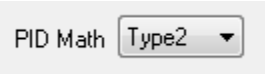
$$CV_n = K_p \times \left[E_n + (T_s/T_i) \times \sum_{i=0}^{n} E_i + (T_d/T_s) \times (PV_n - PV_{n-1})\right] + CV_0$$

*Position and Velocity Algorithms effect the derivative portion of the equation so if using PI control the same equation is used for both Algorithms.*

**PI Position & Velocity Algorithm:**

$$CV_n = K_p \times \left[E_n + (T_s/T_i) \times \sum_{i=0}^{n} E_i\right] + CV_0$$

**PID Math**




Use this selection to decide between Type1 and Type2 Math. Below you will see the details of how each functions:

**Type1:** This math provides a faster response with the Integral and Derivative terms having a bigger impact on the error response. This faster response though makes this less stable. Also the step change percentage is constant at 10%.

**Type2:** (Default) This math provides a more stable response with both the Integral and Derivative terms having a smaller effect. Also the step change percentage for Autotune can be set. Finally the Digital output has more settings in terms of Min ON and OFF time.

**Process Variable (PV) Tag**


Use the drop-down arrow or enter a tag address where you would like the Process Variable to be stored. You can use R or IR register types. If you use an IR type tag, then you are reading the Process Variable directly from an Input Module. If you use an R-type tag, then the expectation is that you are scaling an input in the logic and moving the value to the R-type register so that PID computations can use the PV.

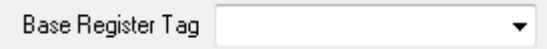
**Note: If you would like PID to run while the PLC is stopped, you should choose an IR type tag so that the PV is updated with the actual value.*

**Control Variable (CV) Tag**


Use the drop-down arrow or enter a tag address where you would like the Control Variable to be stored. The CV tag has the flexibility of using R or OR registers. If you use OR, then EZRack PLC writes the CV directly to an Output Module. If you use the R type for CV tag, you will have to move the actual CV (possibly after some scaling) using ladder to an output for control.

**Note: If you would like to PID to run when PLC is stopped, please use OR type tag for CV so that it can be updated, unless you are using Control Output.*

**Base Register Tag**




Base Register Tag/Address defines the starting address of a Contiguous Block of 32 registers that are used to store PID Parameters and Status information. Please see the dialog box to find the addresses of desired parameter within the block.

**Base Discrete Tag**


Base Discrete Tag/Address defines the starting address of a Contiguous Block of 8 discretes that are used to store PID Parameters and Status information.

**PID Loop Mode**


In Auto mode, the PID Loop calculates a new Control Variable value every sample period. In Manual mode, the Control Variable is controlled by user manually. The manual mode may be used for manual control of process. PID Monitor dialog box (**EZRack PLC > PID Monitor**) can be used to modify Control Variable in manual mode.

When the mode is switched from manual to auto, the integral term of the PID equation is set to the control value. This provides bumpless transfer from manual to auto.

**Anti-Windup**


This option inhibits integration when the control value is saturated. It controls the integral term of the PID equation when the control value is saturated. If Anti-Windup is selected, the integral term is not included when the output is saturated and the sign of the Error will cause the integral term to drive the output further into saturation. This help loops to come back into equilibrium sooner. You can see this in action in the PID Response section.

**Use PV Square Root**


If this option is selected, Square root of PV is used instead of PV in error computation.

$E_n = \sqrt{PV_n} - SP_n$ for Direct Acting

$E_n = SP_n - \sqrt{PV_n}$ for Reverse Acting

Then the $CV_n$ is computed as follows:

**PID Position Algorithm:**

$$CV_n = K_p \times \left[E_n + (T_s/T_i) \times \sum_{i=0}^{n} E_i + (T_d/T_s) \times (E_n - E_{n-1})\right] + CV_0$$

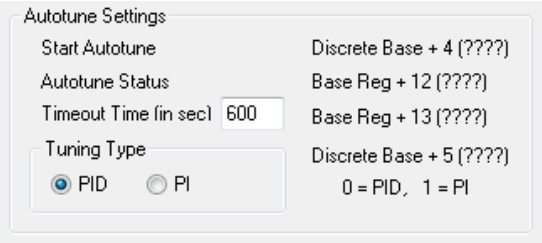
**PID Velocity Algorithm:**

$$CV_n = K_p \times \left[E_n + (T_s/T_i) \times \sum_{i=0}^{n} E_i + (T_d/T_s) \times (\sqrt{PV_n} - \sqrt{PV_{n-1}})\right] + CV_0$$

**Saturation**


This line is for information only. This line shows the address of the discrete bit that would be set if the Control Variable is saturated (i.e. if the Control Variable is either 0 or 4095). You may use this in ladder logic to monitor the saturation of control variable.

**Autotune Setup**




The EZRack PLC can autotune PID loops, i.e. it can estimate the values for the Proportional Gain, Integral (Reset) time, and Derivative (Rate) time for PID loop. The dialog box allows you to setup the loop for autotune. EZRack PLC uses Ziegler-Nichols method to estimate the PID parameters. For step by step guide please see the PID Autotune Directions section.

Following are the setup parameters for Autotune:

*Note: Autotune is performed by EZRack PLC observing open loop response to a known percentage (default 10%) step change in the control value. Before starting autotune, the process should be in a steady state. For best results, the steady state should be within the known percentage of the operating set point. During Autotune, watch the process variable closely for it to be within the safe limits.*

**Start Autotune**

Shown on the dialog box for information only. The Start Autotune discrete is at Discrete Base+4. EZRack PLC initiates autotuning of a loop when this bit transitions from 0 to 1. Autotuning of the loop is can only happen in “Manual” mode. Once Autotune is started, you can cancel it by setting this bit to 0. When Autotune is finished, either from success, error, or user canceled, setting this bit from 0 to 1 will return the Autotune to an idle state freeing any memory allocated for the Autotune.

**Autotune Status**

Shown on the dialog box for information only. During Autotune, EZRack PLC reports the status of Autotune in the register.

| Register Value | Description |
|---|---|
| 0 | Tuning Idle |
| 1 | Tuning in progress, collecting data |
| 2 | Tuning in progress, calculating values |
| 3 | Tuning done, Autotune successful |
| 4 | User canceled tuning |
| 5 | Control Value could not be incremented |
| 6 | The tuning algorithm failed to fit the curve |
| 7 | Division by zero error |
| 8 | Could not determine dead time |
| 9 | One or more of P, I, or D was out of range |
| 10 | Tuning low memory error |

**Timeout Time (in sec)**

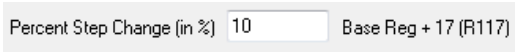
User programs Autotune timeout in seconds in this register. If EZRack PLC cannot finish autotuning within this time, the Autotune is aborted. User should program this field based on the dynamics of the process. This time should equal 1.5 x time to settle after the step change. Please see Autotune section for more information.

- When Autotune Status = 1 (collecting data), this will time down to 0 to indicate the progress of the data collection.
- While Autotune Status = 2 (calculating values), this register will be used as a decrementing counter counting down to 0 indicating the progress of the calculations

**Tuning Type**

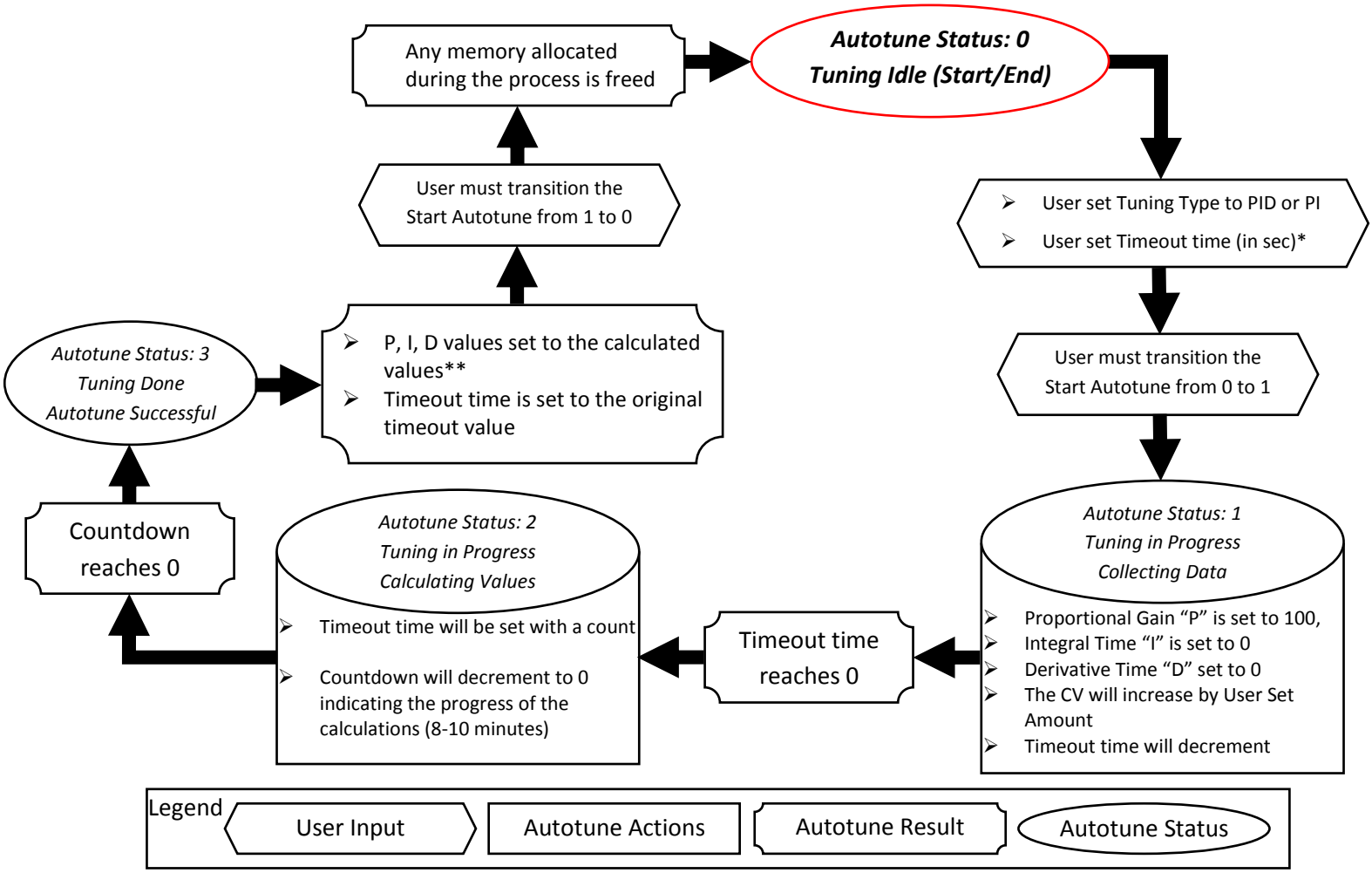
User selects if PI or PID tuning is required.

**Percent Step Change (in %)**




How much the CV will be incremented during the Autotune.

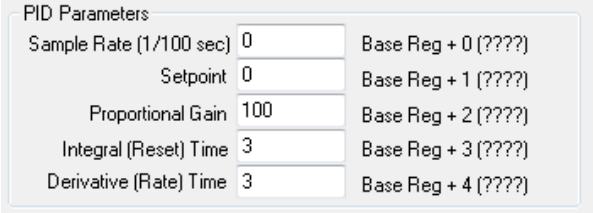
**Normal Autotune Sequence**




*The timeout time should be greater than the time it would take for the process to stabilize after a set% change in CV

**If PI controller is used the D term is set to 0

**PID Parameters**




The PID Parameters consist of the Sample Rate, Setpoint, Proportional Gain, Integral (Reset) Time, Derivative (Rate) Time. The following section briefly describes each of these parameters. For more information on how these effect the PID Loop please see the PID Response section.

**Sample Rate** ($T_S$)

Enter the desired Sample Rate in this field. The Sample Rate is seconds and can be changed from 0.05 to 99.99 seconds.

*Note: All numeric fields in this dialog box use Implied Decimal points. So to enter 0.05, you simply enter 5; the EZRack PLC assumes two digits after the decimal point for most of the numeric entry fields, except where noted.*

**Setpoint**

Enter the Setpoint in this field. This is the Setpoint used in the PID Loop calculation. The Setpoint is the desired process level.

**Proportional Gain ($K_p$)**

Enter the Proportional Gain in this field. This is the gain of the proportional term of the PID equation. The valid range is 00.00 to 99.99. Setting this to zero removes the proportional term from the PID equation.

*Note: The decimal point is implied. For example, "125" is 1.25. Default is 1.00*

**Integral (Reset) Time ($T_i$)**

The units for this time are in seconds. The Valid range is 00.0 to 6000.0. This (along $K_p$ and $T_s$) controls the integral term. Setting it to zero removes the integral term from the PID equation.

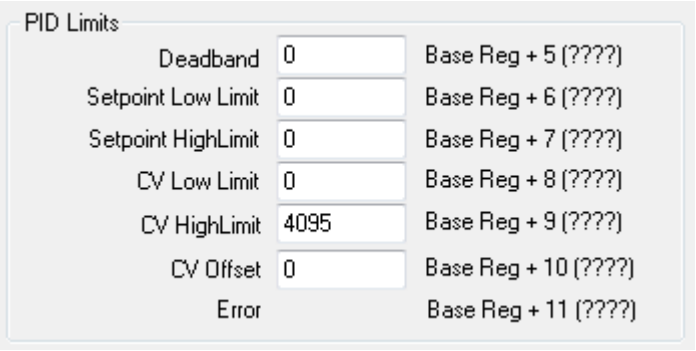
**Note: The decimal point is implied. For example, "125" is 12.5 seconds. Default is 0.3.*

**Derivative (Rate) Time ($T_d$)**

Enter the Derivative Gain in this field. This along with ($T_s$ and $K_p$) makes the coefficient of the derivative term. The units are in seconds. The valid range is 00.00 to 600.0. Setting this to zero removes the derivative term from the PID equation.

**Note: The decimal point is implied. For example, "125" is 12.5 seconds. Default is 0.3*

**PID Limits**




The PID Limits consist of Deadband, Setpoint Low Limit, Setpoint High Limit, CV Low Limit, CV High Limit, CV Offset and Error. The following section briefly describes each of these.

**Deadband**

Enter the Deadband value in this field. This value is compared with the error value at loop update. If the absolute value of the error is less than the Deadband value, then the error is considered as zero for PID computations.

**Setpoint Low Limit**

Enter the lower limit of your desired setpoint in this field. If the setpoint is below this value, then it will be set to the value you've entered in this field.

**Setpoint High Limit**

Enter the higher limit of your desired setpoint in this field. If the setpoint is above this value, then it will be set to the value you've entered in this field.

**Control Value (CV) Low Limit**

Enter the lower limit of the Control Value in this field. If the CV is below this value, then it will be set to the value you've entered in this field. Default is 0.

**Control Value (CV) High Limit**

Enter the higher limit of the Control Value in this field. If the CV is below this value, then it will be set to the value you've entered in this field. Default is 4095.

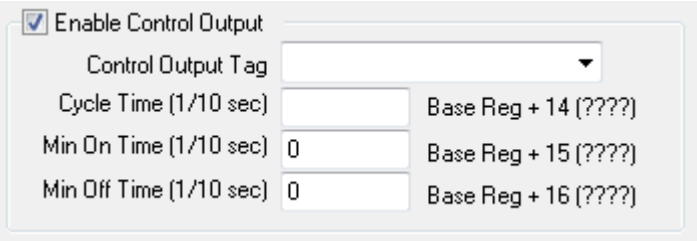
**CV Offset**

This is the constant offset that is added to the control variable. So, even when the Error is zero, the Control Variable equals offset.

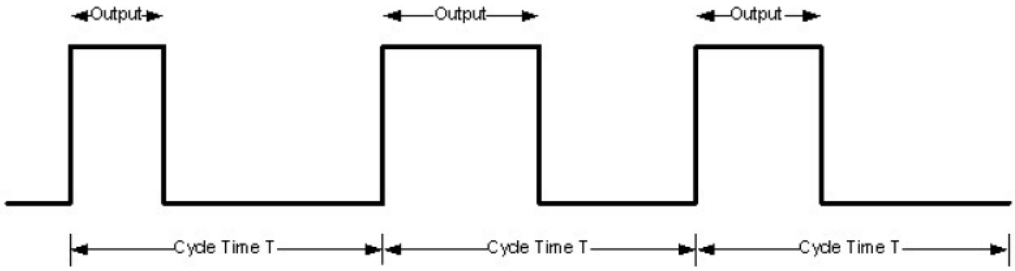
**Error**

Shown on the dialog box for information only. EZRack PLC used this register to store Error value. This is the only register which is Signed Int 16 due to a need to show +/- error values.

**Control Output**




EZRack PLC allows you to control a Digital output using PID control. The digital output provides a pulse out put on selected output address. The width of the pulse (within the cycle time) is proportional to the control value, as illustrated below:




The Output is ON for the time proportional to Control Variable settings.

$T_{out}$ =Time Output will be ON,

$T_C$ = Cycle Time

$$T_{Out} = \frac{CV_n}{4095} * T_C$$

The following fields are programmed for the Digital Control Output:

**Enable Control Output**

Check box to enable Digital Control Output. If the check box is unchecked, no digital output is provided.

**Control Output Tag**

Enter the discrete output address (O type) to provide Digital Control Output from the PID loop. The output module can be of any type (DC, AC or Relay type).

**Cycle Time ($T_C$)**

Enter the Cycle time for the control output in tenths of a second. While selecting cycle time, keep in mind the load type that the output would be driving. For EM relays, we suggest that keep this time as high as possible to extend relay life. This time is in 1/10 seconds so a value of 10 will be 1.0 Seconds.

**Min On Time ($T_{MON}$)**

The Min On Time is used when the digital output controlled device cannot be cycled ON at very fast rates. Therefore you can set the minimum time that output will be ON here. This time is in 1/10 seconds so a value of 10 will be 1.0 Seconds.

Logic behind Min On Time:

IF $(\frac{CV_n}{4095} * T_C < T_{MON})$ THEN $(T_{Out} = T_{MON})$ELSE $(T_{Out} = \frac{CV_n}{4095} * T_C)$

*Note: The $T_{MON}$ must be less than the cycle time or the PID Controller will not functions since any value of $T_{out}$ would never be greater than $T_{MON}$. With this section you also have to be careful since the CV high limits could also affect the maximum $T_{out}$.*

| | |
|---|---|
| Bad Example 1 (**DO NOT DO THIS**)<br>$T_C = 30$<br>$T_{MON} = 30$<br><br>Therefore the equation will always be such that maximum is $T_{MON} = T_C$<br>$\frac{CV_n}{4095} * 30 \leq 30$ | Bad Example 2 (**DO NOT DO THIS**)<br>$T_C = 60$<br>$T_{MON} = 30$<br>$CV_{high} = 2047$<br><br>Therefore the equation will always be such that maximum is $T_{MON} = T_C$<br>$\frac{2047}{4095} * 60 \approx 30 = T_{MON}$ |

**Min Off Time ($T_{MOF}$)**

The Min Off Time is used when the digital output controlled device cannot be cycled OFF at very fast rates. Therefore you can set the minimum time that output will be OFF here. This time is in 1/10 seconds so a value of 10 will be 1.0 Seconds. Unlike the On time the Off time will be added to the cycle time so the previous restrictions do not matter as much.

Logic behind Min Off Time:

IF $(T_C - T_{Out} < T_{MOF})$THEN $(T_C = T_{Out} + T_{MOF})$

Therefore the Cycle time will be extended if the Min Off Time does not fit with the cycle time.

Example:

$T_C = 30$

$T_{MOF} = 10$

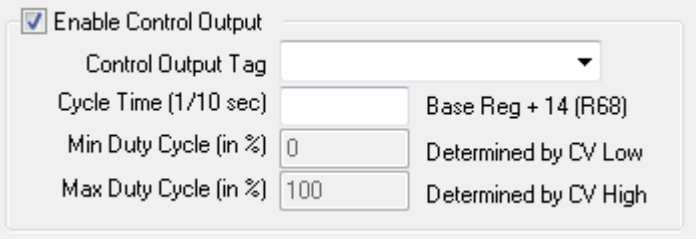
The PID loops calculates $CV_n = 3958$

$$T_{Out} = \frac{3958}{4095} * 30 = 29$$

$30 - 29 < 10$

Therefore the new Cycle time $T_C = 29 + 10 = 39$

**With Math Type1 these figures will be displayed. They are display purposes only.**




**Min Duty Cycle:** This field is for display only. It is computed from the CV Low Limit($\frac{CV_{LowLimit}}{4095} * 100$) and expressed in percentage. As the name suggest, the output will remain on for minimum time even if the computed control value falls below the CV Low Limit.

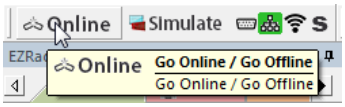
**Max Duty Cycle:** This field is for display only. It is computed from the CV High Limit($\frac{CV_{HighLimit}}{4095} * 100$) and expressed in percentage. As the name suggest, the output will remain on for this maximum time even if the computed control value is above the CV high Limit

## PID Monitor

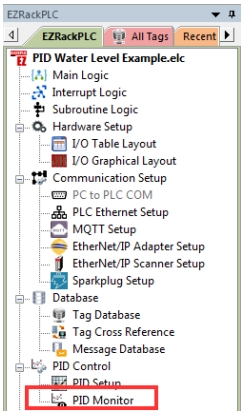
This section will explain how to setup and use the PID Monitor function within the EZRack PLC Designer Pro. You can use this function to monitor and make real-time changes to your PID Loop. To use the PID Monitor follow the directions below.

To use PID Monitor:

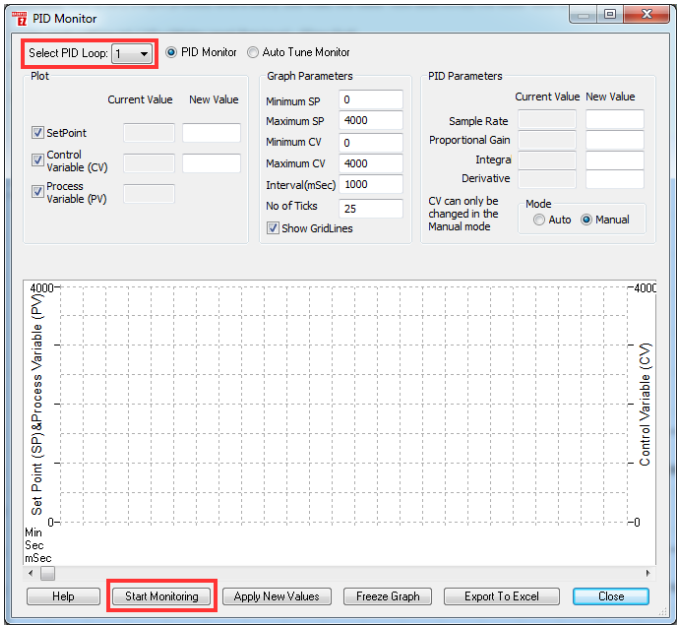
1. Open your PID project. Then **Go Online** with your project.




2. Once Online with your project select PID Monitor from the left Quick Access Bar, or go to **PLC > PID Monitor….**




3. In the PID Monitor select the PID Loop (1-8) you would like to monitor then press Start Monitoring. You can now modify your PID Loop, for more information on all the options please see next section.

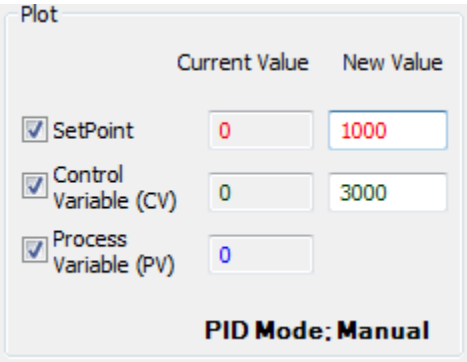


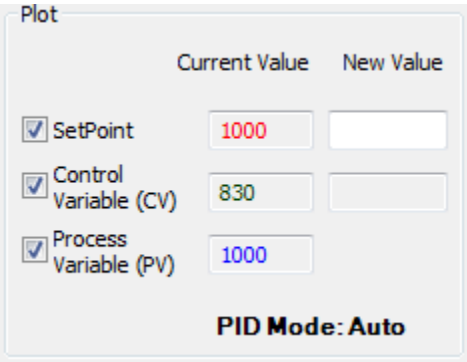
**PID Monitor Options**

All the below options are editable while not monitoring but changes are only reflected when you are monitoring your loop. The PID Monitor will not see all the different values that PID loop will use to calculate its error. The PID Monitor operates at inputted interval from Graph Parameters. The Auto Tune Monitor is discussed in the Auto Tune Section.

*Note: For values which exist as parameters, inputs, or outputs from the PID Loop the PID monitor will change the value of those corresponding registers. For example for Control Variable (CV) the tag inputted for the specified loop will be modified.*

**Plot Information**




**Setpoint**

This field displays the current value of your Setpoint. You can change the setpoint by entering a value in the New Value field and clicking the Apply New Values button at the bottom of the window.

**Control Variable (CV)**

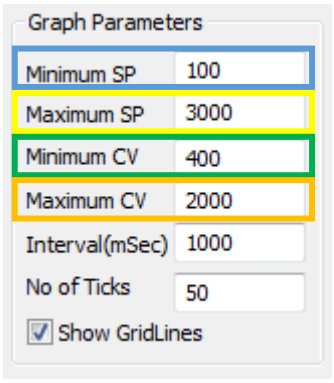
This field displays the current value of the Control Variable (CV). This value can be changed when the PID mode is Manual. This is the controlled Output register that the PID Loop in Auto Mode calculates.

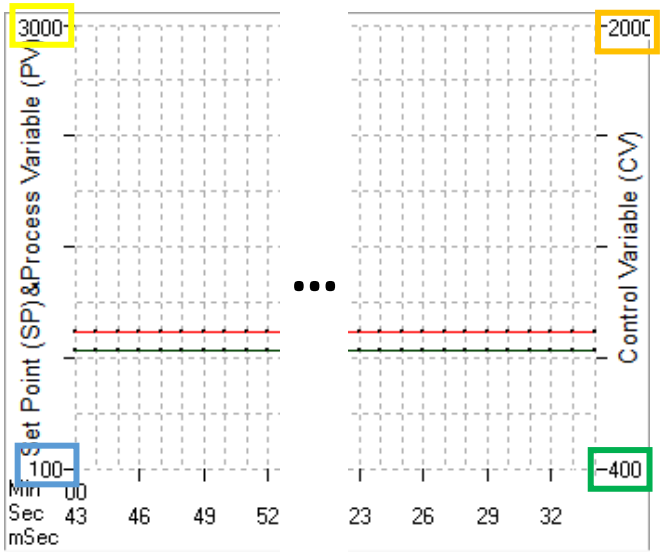
**Process Variable (PV)**

This field displays the current value of the Process Variable (PV). This is the value from the external process which the PID Loop in Auto Mode calculates the error from.

**Graph Parameters**

The Graph parameters will change the maximum and minimum values of the graph.




**Minimum SP**

Enter the Minimum Setpoint value in this field for the graph to display.

**Maximum SP**

Enter the Maximum Setpoint value in this field for the graph to display.

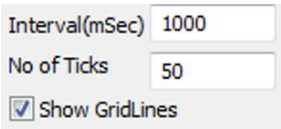
**Minimum CV**

Enter the Minimum Control Variable (CV) value in this field for the graph to display.

**Maximum CV**

Enter the Maximum Control Value (CV) value in this field for the graph to display.

*NOTE: When selecting your values for Minimum and Maximum SP, it's a good idea to choose a number relatively close to the Process Variable. That way, when your graph is created you will be able to see more detail. The greater the range between your Minimum and Maximum SP, the less detail your graph will display. The shorter the range, the more detailed your graph will be.*




**Interval (mSec)**

Enter the Interval value (in milliseconds) in this field. This is how fast your graph will refresh data as well. So to see more details about changes the smaller this value should be.

**No of Ticks**

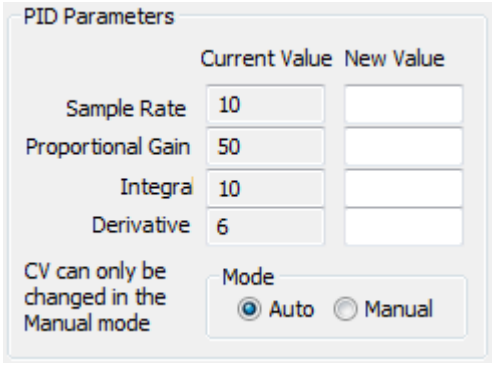
In this field, enter the Number of Ticks you would like to have displayed in the graph.

**Show Grid Lines**

Check this box if you would like Grid Lines to be displayed in your graph.

**PID Parameters**

The PID Parameters section allows changing of the PID Loop calculation while it is running. This section is recommended to be used for tuning the loop after an Auto Tune has been run and you wish to further refine or test your PID response. To see how this is used you can see the PID Autotune section and or some of the PID Application Examples.




**Sample Rate**

In this field you see the current Sample rate and you can enter the new Sample Rate you would like to change to. Sample rate determines how often the PID Loop checks the process.

**Proportional Gain ($K_p$)**

In this field you see the current Proportional Gain and you can enter the new Proportional Gain you would like to change to. This is the gain of the proportional term of the PID equation.

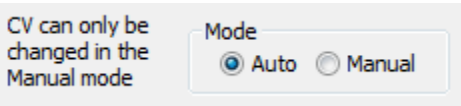
**Integral (Reset) Time ($T_i$)**

In this field you see the current Integral (Reset) Time and you can enter the new Integral (Reset) Time you would like to change to. The units for this time are in seconds. This (along $K_p$ and $T_s$) controls the integral term. Setting it to zero removes the integral term from the PID equation.

**Derivative (Rate) Time ($T_d$)**

In this field you see the current Derivative (Rate) Time and you can enter the new Derivative (Rate) Time you would like to change to. This along with ($T_s$ and $K_p$) makes the coefficient of the derivative term. The units are in seconds. Setting this to zero removes the derivative term from the PID equation.

**Mode**




In this box you can choose Auto or Manual (you can only change the Control Variable in the Manual Mode).

**PID Control Buttons**


**End Monitoring / Start Monitoring**

Press this button when you wish to stop / start the PID Monitor.

**Apply New Values**

Press this button once you have changed some of the parameters in PID Monitor and would like to begin monitoring those changes.

**Freeze Graph**

Press this button if you would like to see a still picture of the graph in its current state.

**Export to Excel**

Press this button to send all of the data within the graph to an Excel spreadsheet (you must have the Excel software installed onto your computer).

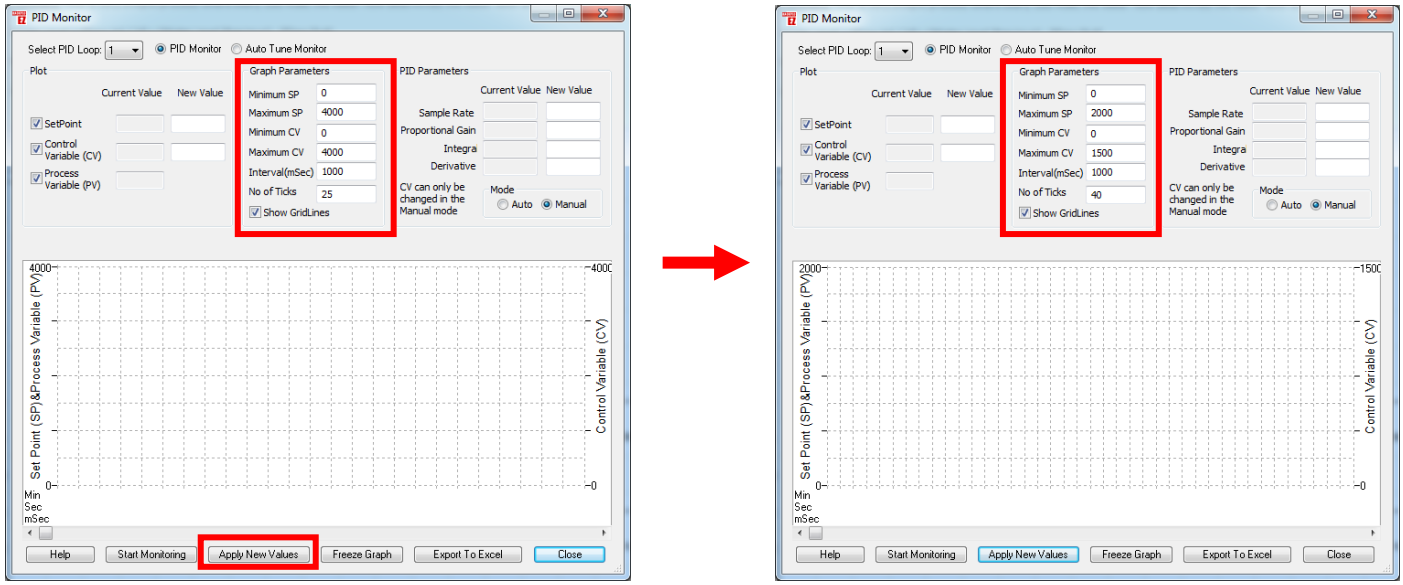
**Close**

Press this button stop the monitoring process and close the PID Monitor window.

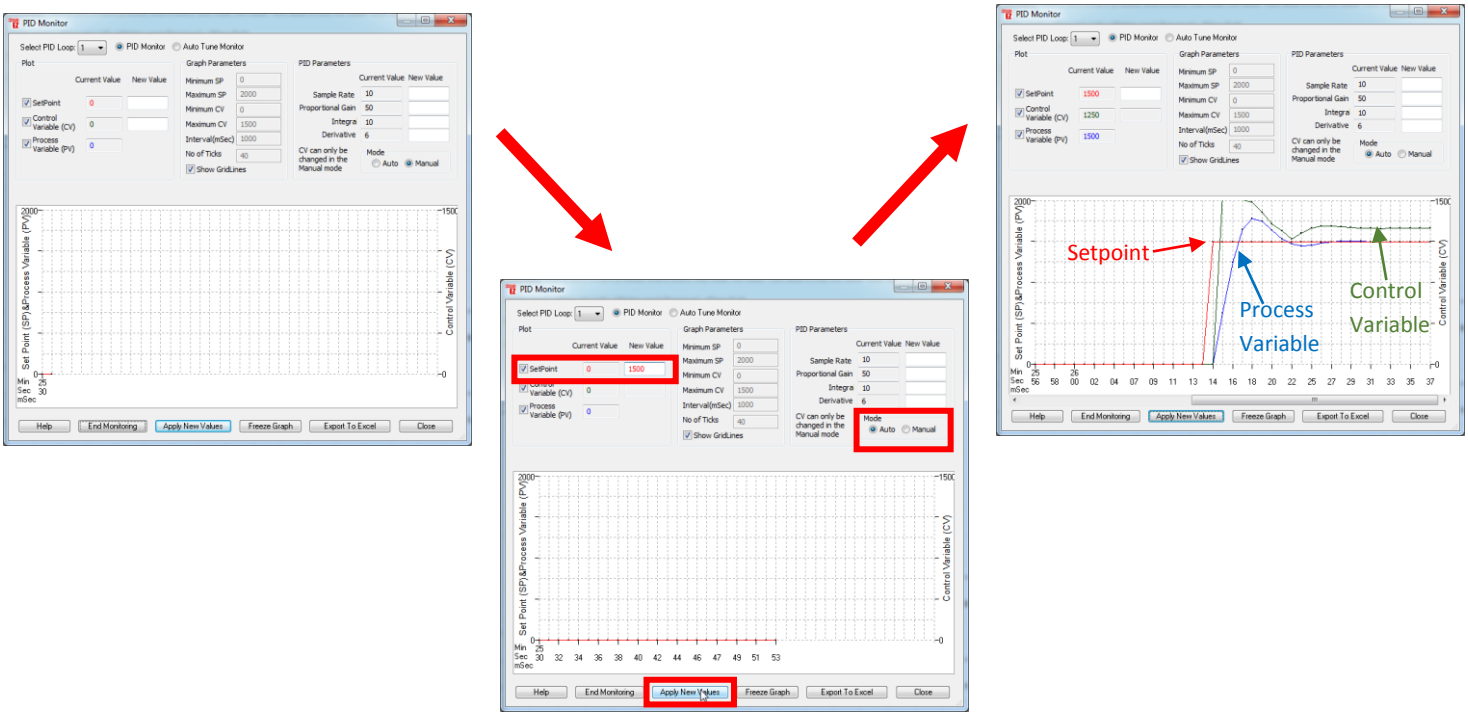
**PID Monitor Example Response**

This section will show how the PID Monitor will look and how to modify settings and to be able to finish tuning your PID response.

- When the PID Monitor is started it will look like this before you start monitoring. If you know your settings you can modify your Graph Parameter so your process is easier to monitor (press Apply New Values to see your changes).




- Now you can Start Monitoring. Once Monitoring you can switch to Auto Mode and Enter a value of 1500 into your setpoint. You can now watch the PID response of your loop.

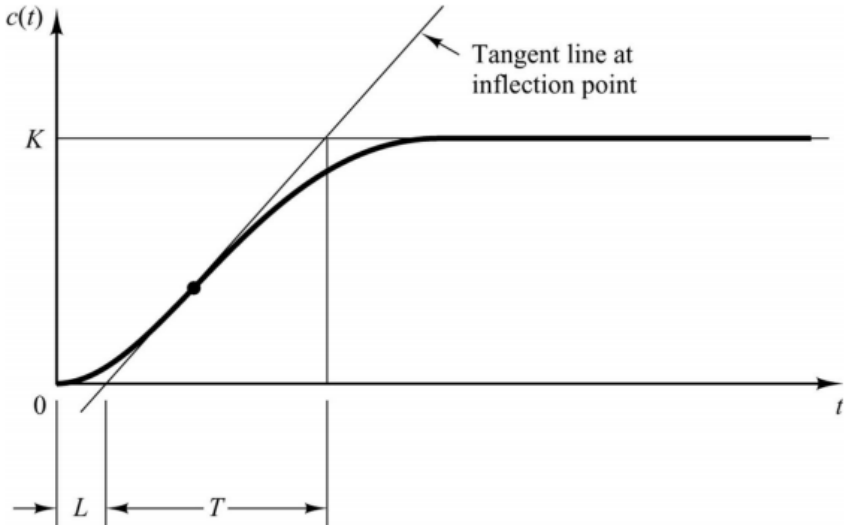


## PID Autotune Directions/Example

This section will outline how to setup an Autotune and how to run an Autotune to get correct values. One thing to remember that the Autotune results are a starting point for your tuning and most likely manual tuning of the PID will need to follow. First there are a couple important points to consider which are outline below. Then initial setup is discussed. Finally a step by step directions for the Online Autotune instructions are presented.

**Autotune Algorithm**

The Autotune process is based on the Ziegler–Nichols Tuning 1st Method S-shaped Step Input Response Curve. Therefore if your process does not work on this principle the Autotune will fail. There exist multiple other tuning calculations but those will need to be run manually. Below the basics of Ziegler-Nichols tuning are outlined but the most important point to remember is that if your process does not produce an S-shaped Step Input Response then this Autotune will not work.




The S-shaped reaction curve can be characterized by two constants, delay time L and time constant T, which are determined by drawing a tangent line at the inflection point of the curve and finding the intersections of the tangent line with the time axis and the steady-state level line.

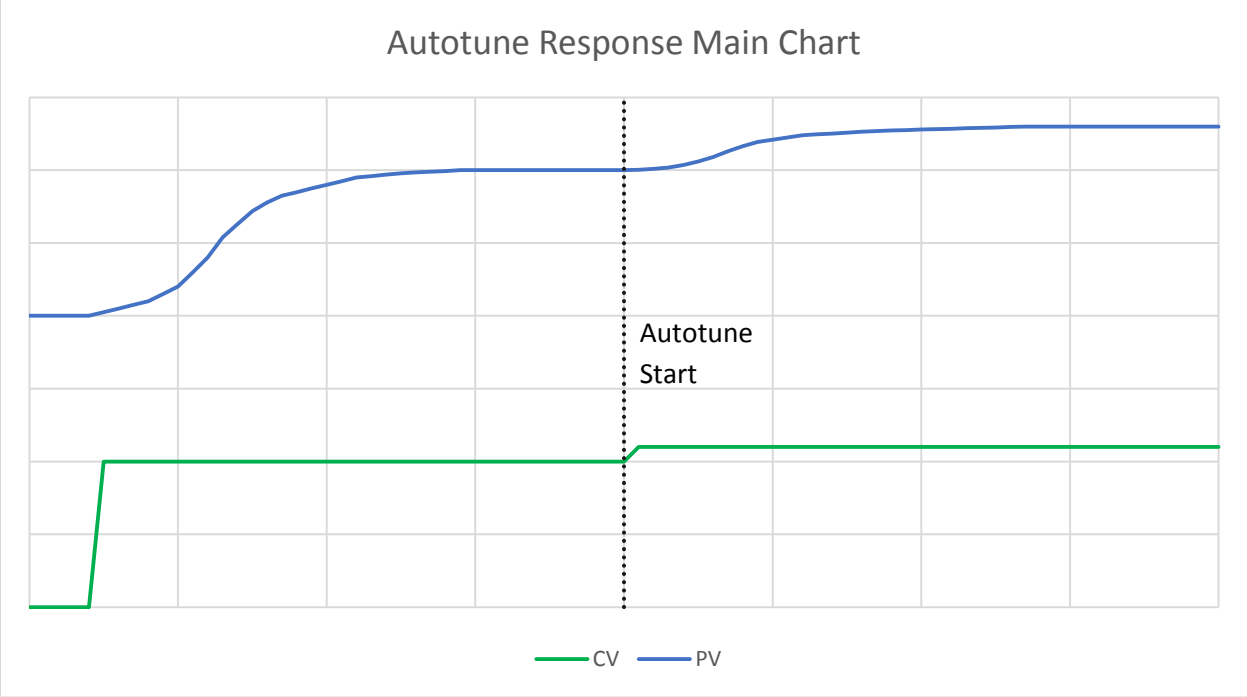
| Type of Controller | $K_p$ | $T_i$ | $T_d$ |
|---|---|---|---|
| P | $\frac{T}{L}$ | $\infty$ | 0 |
| PI | $0.9\frac{T}{L}$ | $\frac{L}{0.3}$ | 0 |
| PID | $1.2\frac{T}{L}$ | $2L$ | $0.5L$ |

The Autotune process on the EZRack automates the creation of this chart to calculate the above values for your PID Parameters.

Other Important Points to Consider before using Autotune:

- CV and PV Values used for Autotune
- Time Out Time vs Response Time
- Stability of Response

Note: Autotune works on any type of Control Variable and Process Variable. If a tag is able to be used for these then the Autotune will try to fit the data to its corresponding best fit PID parameters. But based on the process complexity the results accuracy will vary and a manual tune might be necessary.

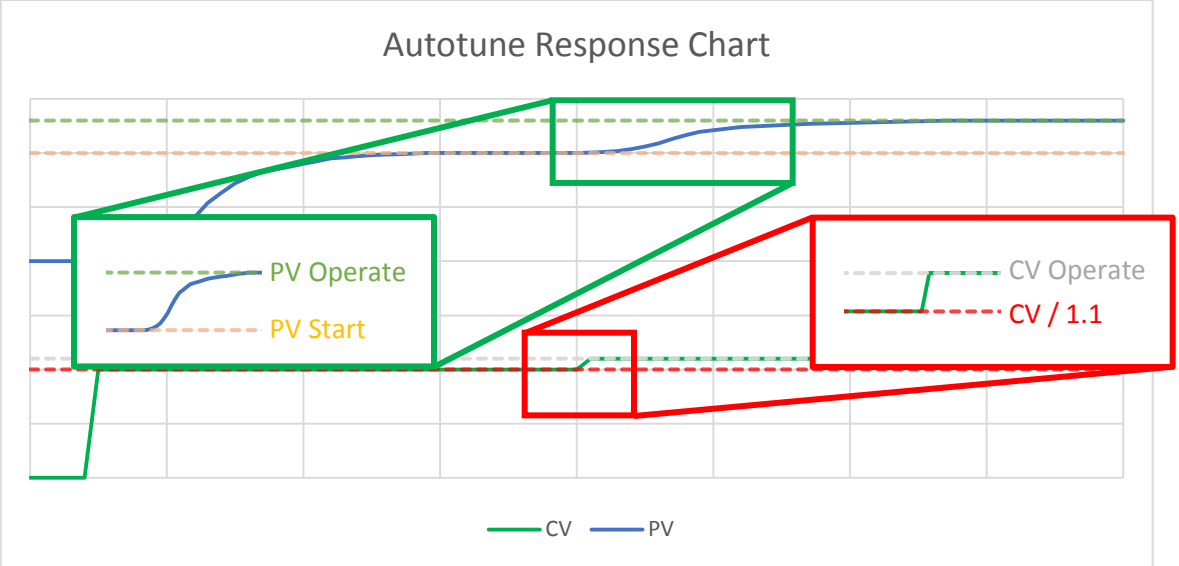



*The graph above will be used to illustrate the important points above. The graph represents idealized response during an autotune. The process starts at zero and the CV is manually set to 10% below the operating CV. You wait till the PV stabilizes and then Autotune is started. The CV is increased by 10% and data is taken while PV stabilizes at the new setpoint. This is used to calculate the Autotune PID parameters.*

**Control Variable / Process Variable Considerations**

For Autotune to calculate parameters that work for your setpoints, it is recommended to run Autotune at around your normal operating setpoint. The default autotune increments the CV by 10% and therefore it is recommended to run the autotune at a starting point where 1.1 * CV is equal to the operating CV. The example below illustrates this point.

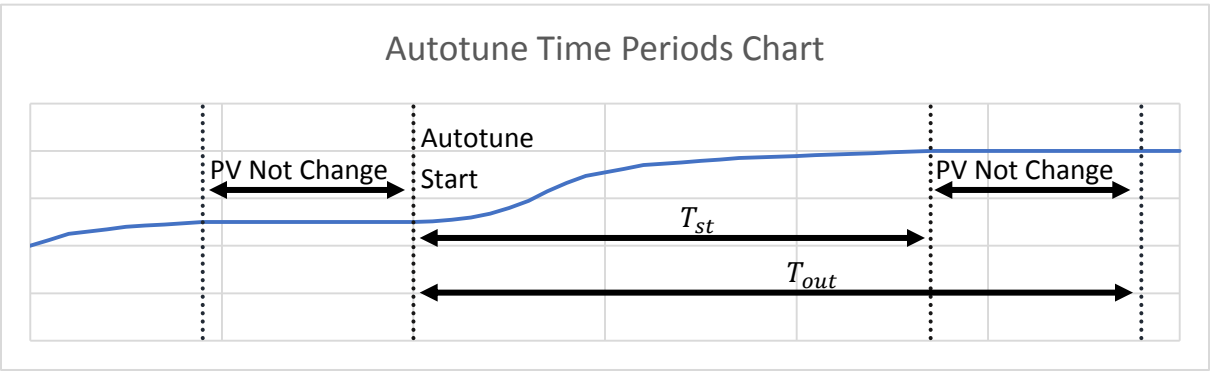
$CV_{op} = 1.1 * CV_{Start}$ where $CV_{op}$ is Operating CV and $CV_{Start}$ is Autotune start CV




**Time Out Time Considerations**

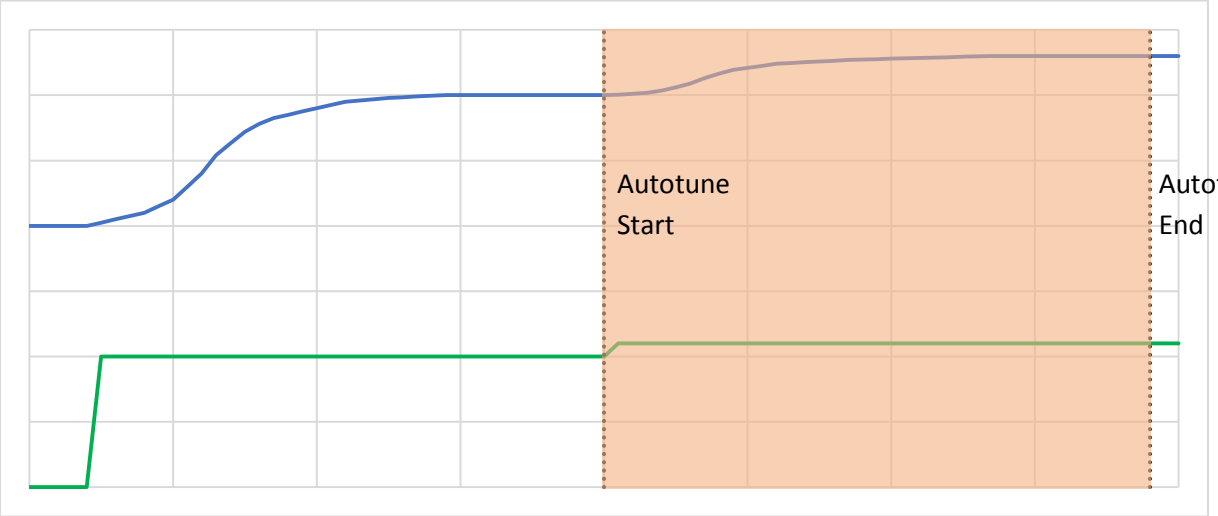
For Autotune to work correctly the process has to stabilize before the Autotune finishes taking data. This therefore means that your timeout time should be at least 1.5x bigger than the time it takes the process to move to the new stable point. Also Autotune should only be started when the process is entirely stable. For our purposes stable means that your value PV is only changing by a small margin (<1%) of error (such as your Deadband value).

$T_{out} = 1.5 * T_{st}$ where $T_{st}$ is the time from CV change till time where your PV is not changing



**Stability of Response**

One of the big variables that could influence getting correct values during an autotune is the stability of the process before and during the end of the Autotune.




Stability Consideration

- Your process must be able to reach a stable operating point. This means that when CV is not changing the PV changes must be very small (<1%). If your process is not stable than most likely Autotune will not converge to correct PID parameters. You can still run the Autotune to get an idea of starting PID parameters but you will have to do some manual tuning.
- To get the best results from PID tuning you must allow the process to stabilize before the Autotune starts and before the Autotune ends. Like mentioned in Time Out consideration section the process should end after stability is reached and process stays there for set amount of time.

**Autotune Considerations Summary**

If the response has an S shaped response and the above considerations are taken into account then most likely the Autotune will result in some values for your PID Parameters. Note that the above consideration do not guarantee results, just make results more accurate and useful. Once the Autotune process is done, please try the PID parameters and see if they make sense for the response you want. If they do not then you should attempt to do a manual tune of your process. The next section will walk through the actual setup and use of the Autotune process.

**Autotune Offline Setup**

This section will outline the necessary setup to run an Autotune. Before running an Autotune please know the following information about your process.

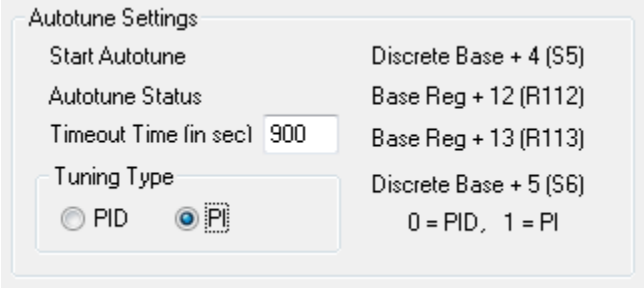
Necessary Information:

| Parameter | Explanation |
|---|---|
| **PID or PI** | The Autotune tunes either a PI or PID response, therefore this is necessary to select the correct one for your process. For more information the PID Response section outlines the differences. |
| **$T_{out}$** | The Time Out time is how long the Autotune process functions, please see the Time Out Time consideration section and use a correspondingly selected Time Out time. |
| **Sample Rate** | The Autotune uses the Sample Rate to know how fast it should sample, therefore the Sample Rate will affect the Autotune results. For more information see the PID Response section. |
| **Action** | The Action selection effects the Error calculation for the PID, therefore the Action selection must be correct or your PID parameters will not make sense. See the PID Setup Parameters section. |
| **Algorithm** | If using PID then the Algorithm will affect the equation that is used to create the PID parameters. For more information see the PID Setup Parameters section. |
| **CV Limits** | The Autotune calculation will operate within your CV limits so please make sure these are correct before using Autotune. |
| **CV Offset** | The CV Offset is used in the calculation for the PID Parameters so for most accurate results use the CV Offset your process will be using later ON. |
| **Optional** | |
| **PV Square Root** | If your process will use a PV Square Root make sure it is selected before running an Autotune. See PID Setup Settings section for more info. |
| **Step Change** | The Autotune uses default 10% step change, this can be changed if necessary. |
| **Digital Output** | |
| **Cycle Time** | Know and input a good cycle time for your process, see PID Setup Settings section for more info. |
| **Min ON Time** | Know and input the Min ON time for your devices, see PID Setup Settings section for more info. |
| **Min OFF Time** | Know and input the Min OFF time for your devices, see PID Setup Settings section for more info. |

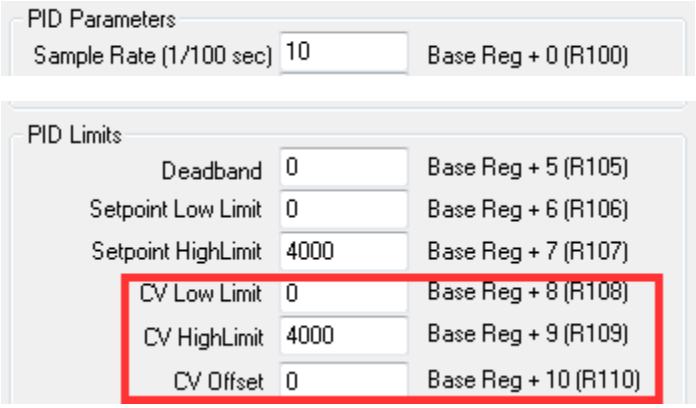
Setup:

1. Before setting up Autotune please follow the directions to setup the PID itself. For directions on this setup please see the PID Setup section. Based on the table above please make sure those sections are correct for your process.
2. For Autotune to function correctly the following sections will need to be set.
3. In PID Loop Mode, set the Mode to Manual.

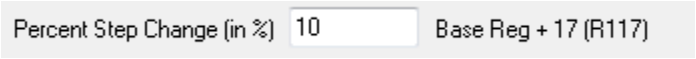
4. In Autotune Settings, input your process specific Time Out time (most likely default time will not work for your process).
5. Then select the Tuning type based on if you would like PI or PID control for your process.



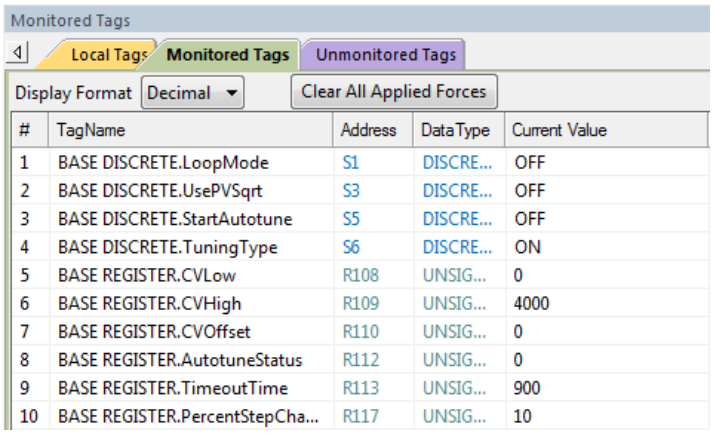

6. Next verify the Sample Rate and CV Limits / Offset to make sure that they are what you would operate the PID control at. This information should be based on your process information.




7. Finally if you would like to change the Autotune Step Change you can do that here as well.



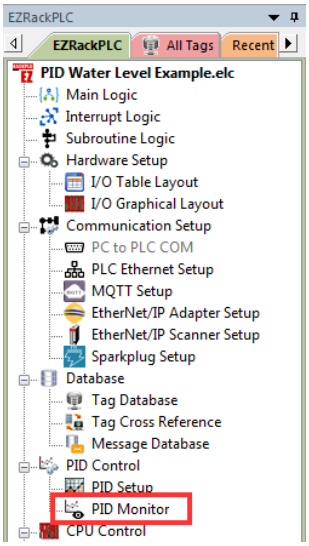

*Note: All these settings can be changed in your ladder logic using the defined registers. Therefore you can redo an Autotune with different setting without re-downloading the project.*



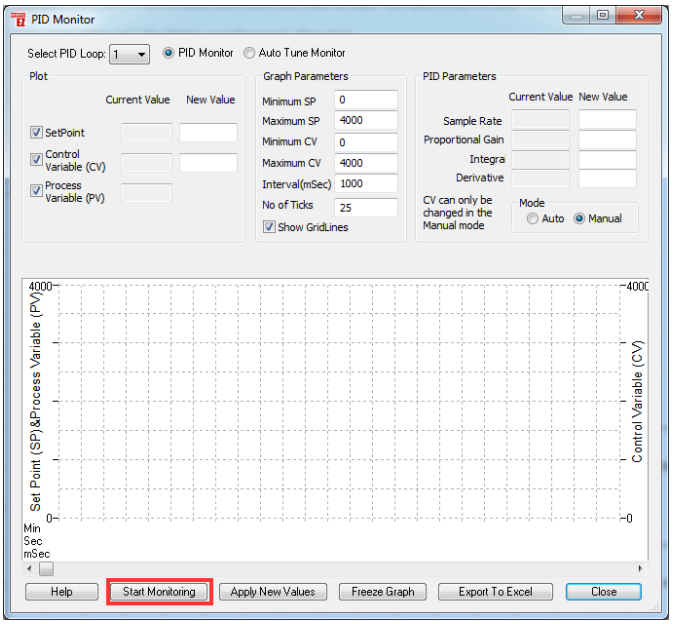

| # | TagName | Address | DataType | Current Value |
|---|---|---|---|---|
| 1 | BASE DISCRETE.LoopMode | S1 | DISCRE... | OFF |
| 2 | BASE DISCRETE.UsePVSqrt | S3 | DISCRE... | OFF |
| 3 | BASE DISCRETE.StartAutotune | S5 | DISCRE... | OFF |
| 4 | BASE DISCRETE.TuningType | S6 | DISCRE... | ON |
| 5 | BASE REGISTER.CVLow | R108 | UNSIG... | 0 |
| 6 | BASE REGISTER.CVHigh | R109 | UNSIG... | 4000 |
| 7 | BASE REGISTER.CVOffset | R110 | UNSIG... | 0 |
| 8 | BASE REGISTER.AutotuneStatus | R112 | UNSIG... | 0 |
| 9 | BASE REGISTER.TimeoutTime | R113 | UNSIG... | 900 |
| 10 | BASE REGISTER.PercentStepCha... | R117 | UNSIG... | 10 |

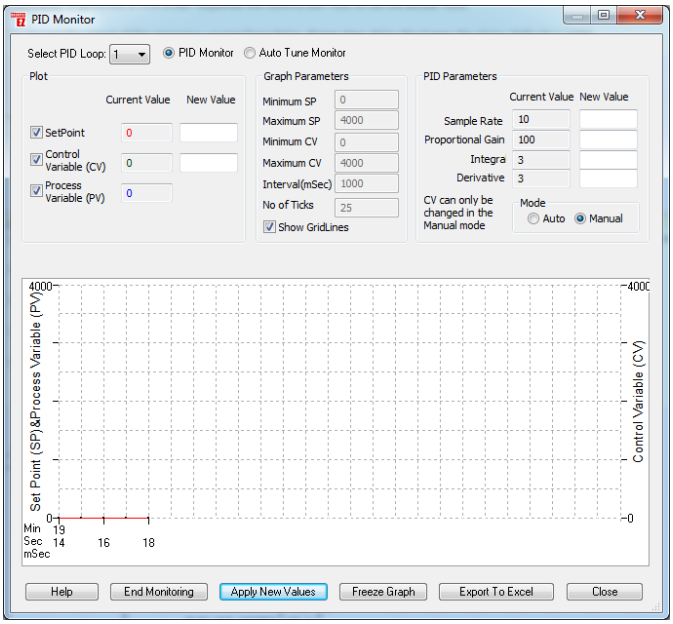
**Autotune Online Directions**

1. Before these directions, make sure to follow the above directions for Offline Setup.
2. Please make sure that your EZRack PLC is connected to your process, also that it can control and take accurate data.
3. Download your project and Go Online with the EZRack PLC. Make sure to stay in Edit Mode.
4. Now double click the PID Monitor option on the left (also in Setup > PID Monitor).

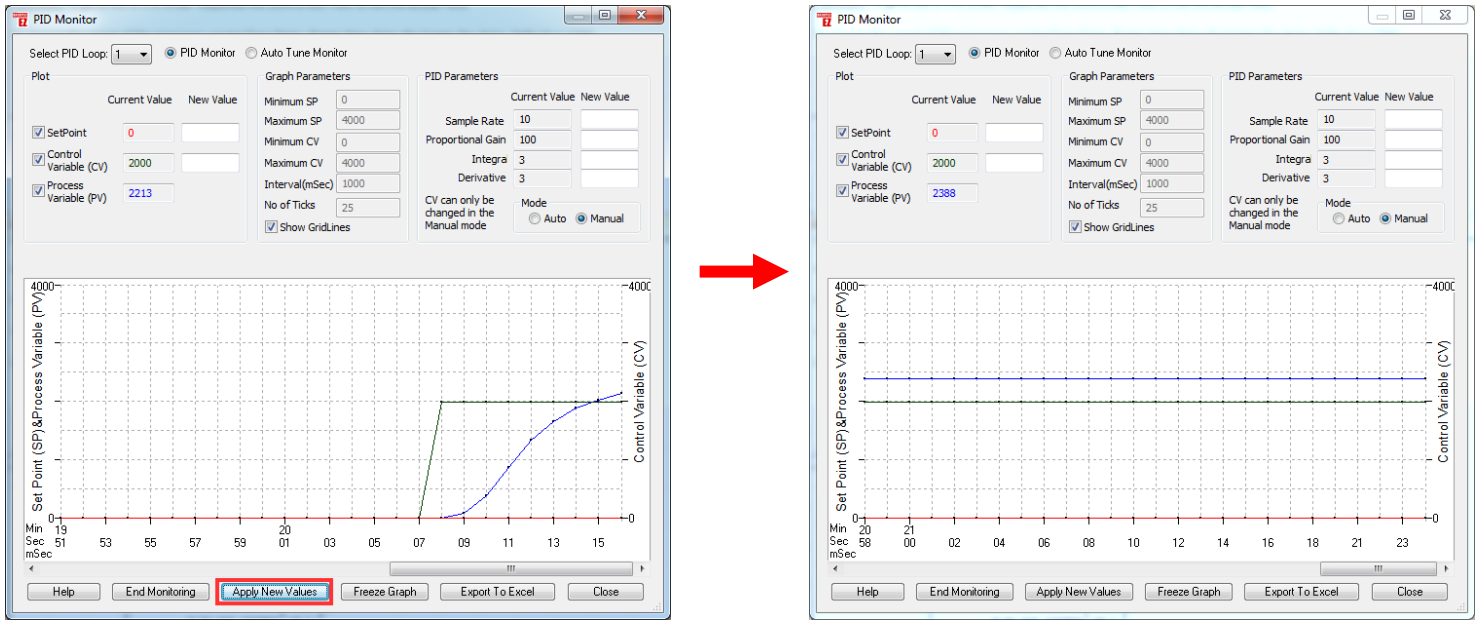



5. In the PID Monitor Click Start Monitoring. The PID Monitor will populate with data from your process. *Note: This includes any preliminary PID parameters you have inputed, these will not be used for your Autotune process.*

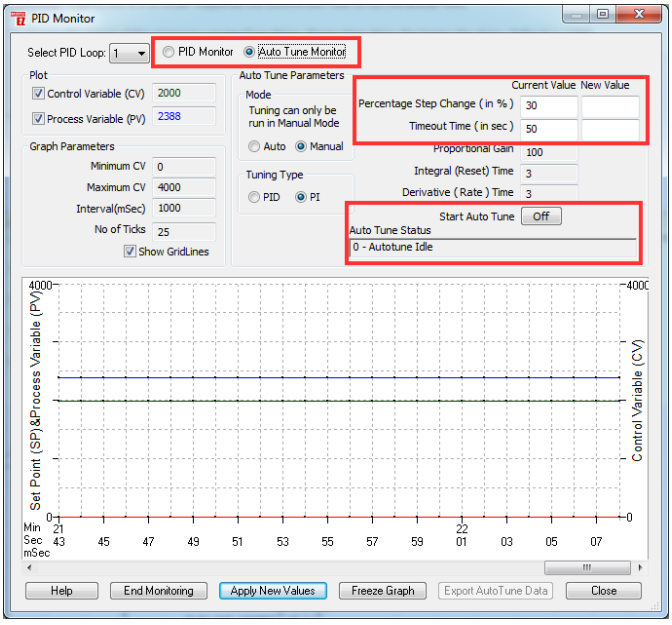


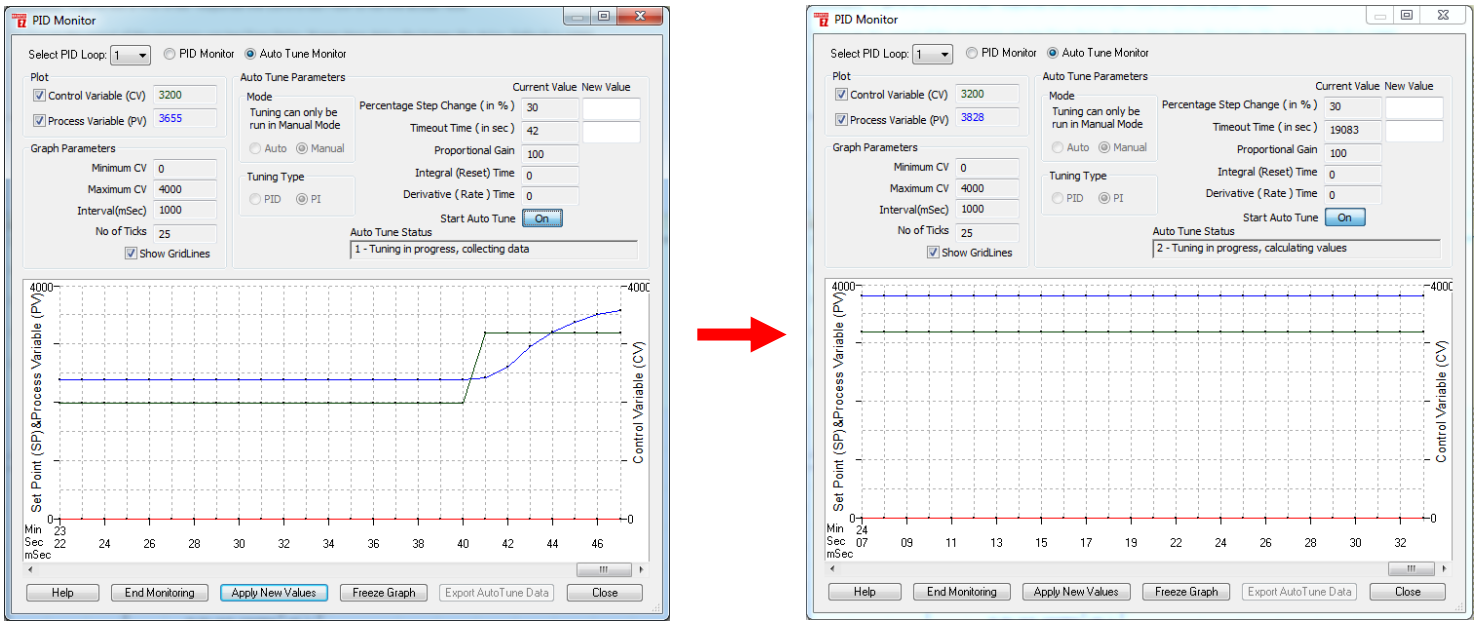
6. Next make sure the PID Loop selected is the one you would like to finetune. Then input your Autotune selected starting control variable. Make sure to press Apply New Values. For more information see the Control Variable / Process Variable considerations section.
7. Wait for the process to stabilize, then make sure to wait at least 30 seconds more than that.



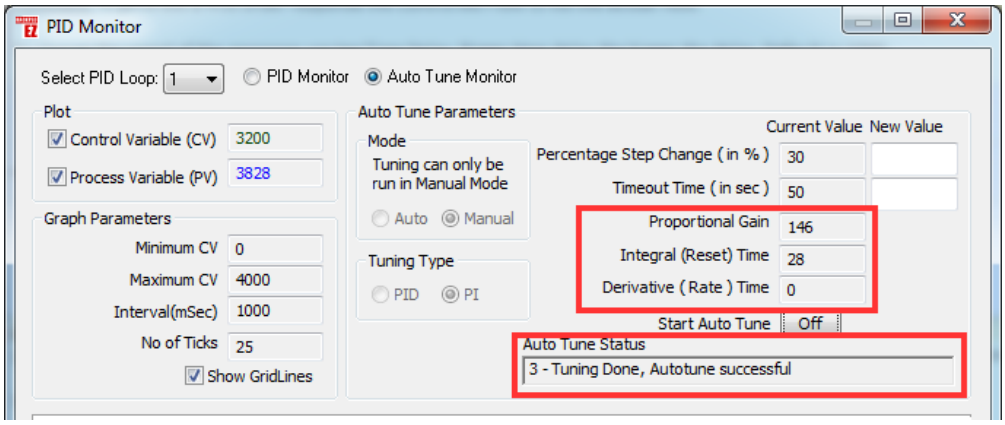

8. Next switch from PID Monitor to Auto Tune Monitor. Please make sure your Step Change % and Timeout time are correct then you can click Start Auto Tune. *Note: Autotune will only work in Manual Mode so if nothing is happening please make sure the Mode is Manual.*



9. You will now need to wait both the Timeout Time (Process Specific) and the Calculation Time (8 to 10 minutes). The Auto tune status will be displayed in the Auto Tune Status section. For a graphical Autotune representation you can see the next section. *Note: At any point the Autotune can be canceled by turning the Start Auto Tune to OFF.*



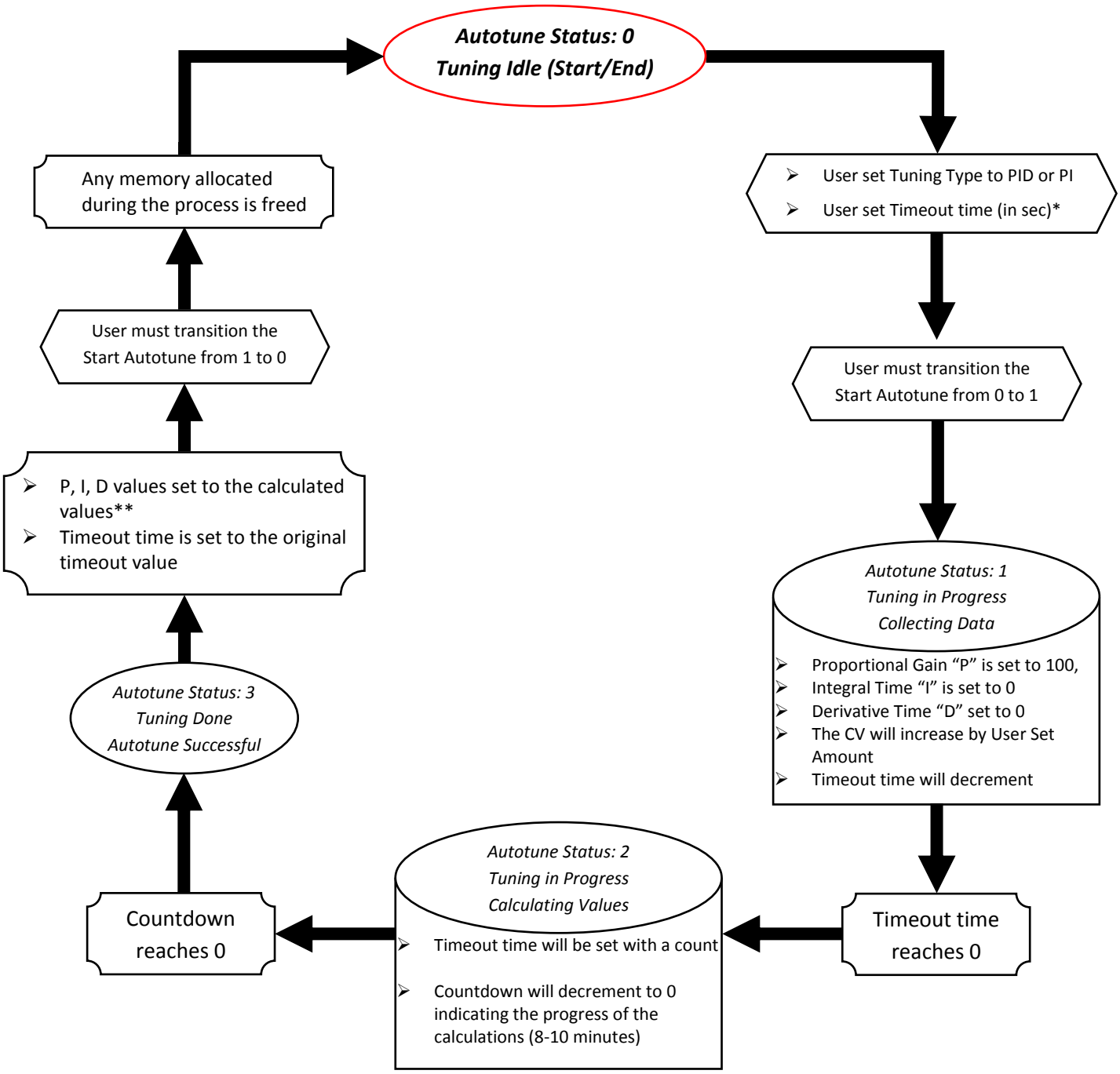

10. Once the Autotune finishes it will put in the calculated value into the PID Parameters. ***Note: These values are not saved offline so please make sure to record them and input them in your offline setup.***




11. Please make sure to press the Start Auto Tune Off button so that the process is finished.

The next section includes a graphical process sequence. Also there exist some Application Guide with example programs for PID Autotuning.

**Normal Autotune Sequence**




*The timeout time should be greater than the time it would take for the process to stabilize after a set% change in CV

**If PI controller is used the D term is set to 0

## PID Troubleshoot

This section will highlight some common errors and or problems that can be encountered when setting up PID.

**PID Not Function**

| Cause | Explanation | More Information |
|---|---|---|
| **Sample Rate Not Correct** | The Sample Rate controls how often the PID is evaluated. A larger value, or zero value could cause process and PID to not correlate correctly. | PID Response Section |
| **Action Wrong** | The PID process should be selected to be the opposite of how your process functions. If this is not correct your PID will not work. | PID Setup Settings Section |
| **CV Limits too small** | The PID can only work within your Control Variable limits, if the range is too small your process could be constrained and not function correctly. | PID Setup Settings Section |
| **Deadband too big** | If your error is within the Deadband the PID will not react. Therefore PID will not function till move out of Deadband limits. | PID Setup Settings Section |

**Process Overshoot**

| Cause | Explanation | More Information |
|---|---|---|
| **PID Parameters Incorrect** | Your PID parameters can be tuned to a point where overshoot is eliminated but your response will then be slower. Use the Autotune and manual tuning to achieve the response you would like. | PID Response Section<br>Autotune Section |
| **Anti-Windup not used** | Anti-Windup works with the Integral term to make overshoots much smaller. If not enabled your overshoot could be much larger than it should be. | PID Response Section |
| **CV Limits too small** | The PID can only work within your Control Variable limits, if the range is too small your process could be constrained and not function correctly. | PID Setup Settings Section |

**PID Not Going to Setpoint**

| Cause | Explanation | More Information |
|---|---|---|
| **Sample Rate Not Correct** | The Sample Rate controls how often the PID is evaluated. A larger value, or zero value could cause process and PID to not correlate correctly. | PID Response Section |
| **PID Parameters not tuned** | The PID parameters might be too small or big to achieve your desired setpoints. Please tune your parameters more or run an Autotune nearer to your operating setpoint. | PID Response Section<br>Autotune Section |

**PID Digital Output Always ON or OFF**

| Cause | Explanation | More Information |
|---|---|---|
| **Cycle Time incorrect** | The cycle time is the $T_c$ used in Digital Output calculation below. Therefore a too short or too long time could mean your Digital output could be ON/OFF for very long periods. $$T_{Out} = \frac{CV_n}{4095} * T_C$$ | PID Setup Settings Section |
| **Min ON Time too Long** | If the Min ON time is too long the Cycle time could be over before the Digital output could turn Off. This is an important consideration when deciding your Cycle Time. | PID Setup Settings Section |
| **Min OFF Time too Long** | The Min OFF time is added to the cycle time so if this is too long your cycle time could be extended too long and your process control not function correctly. | PID Setup Settings Section |