

## Introduction

The purpose of this tutorial is to demonstrate the PID function of the i<sup>3</sup> by programming a simple temperature controller logic diagram.

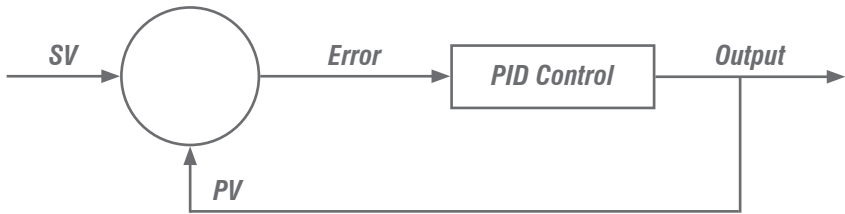
This first program will read a voltage signal and control a relay output to manipulate the control variable. For this example we will use an i3AX12X/13C14-SEHF. The second program will directly read in a thermocouple value and output a 4-20mA signal to operate the control valve. This program will use the model i3AX12X/10E24-SEHF.

PID control is a Closed Loop control algorithm used in a wide range of process control applications from temperature control to flow control. The i<sup>3</sup> has analogue and digital outputs to which the PID controlled result can be outputted to. With the i<sup>3</sup>, we could communicate with a separate PID controller over Modbus or better still incorporate this function in the i<sup>3</sup> and have the output controlled from a PID control function in the ladder logic program.

Every PID control loop must match the environment they operate in. This is called tuning the system. Tuning requires some mathematical calculations or as in the i<sup>3</sup>, we can auto-tune, where the maths can be performed automatically. Understanding the PID process is not easy but the i<sup>3</sup> has features that make implementing the process straightforward.

## PID Background Theory

PID is made up of 3 controlling characteristics: Proportional, Integral and Derivative, hence PID controllers are known as three term controllers.



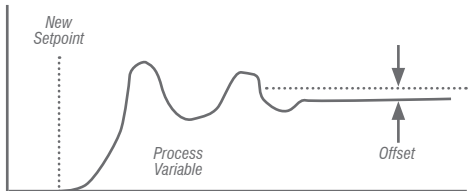
The three terms have different effects on the output:

Proportional	Integral	Derivative
Delivers an output that is proportional to the size of the difference in set point (SP) and process value (PV).	Removes the steady state control offsets by ramping the output in proportion to the amplitude and duration of the difference in SP and PV.	Proportional to the rate of change of the process value.

## Proportional Control

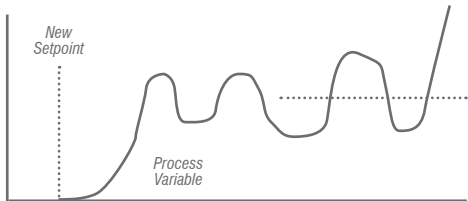
A controller, which performs the action below is known as a Proportional Controller. In practice, Error is actually a portion (often expressed in percent) of the full-range error. In the example below, if the Error is 150 degrees, the controller might be programmed to add only 20% - 30% of the full error value. The process takes longer to change since it is not being driven as hard, but full control is more accurate.

The following is a graph of a typical process under Proportional control only:



Proportional control often has an Offset factor. That is, the process almost never has 0 error. This can be caused by a variety of reasons, all of which are outside the realm of control of the Proportional function.

On the other hand, adding too much Proportional control can cause the process to oscillate and go further out of control:



There is almost always a lag or time delay in the process. Most Process Variables cannot change instantly. This is especially true of heat-related processes. Change in heat can be very slow. Pressure changes and flow rates can also be tardy. These are all due to physical factors in the system and are usually outside the realm of control of the Process Controller.

## Bias

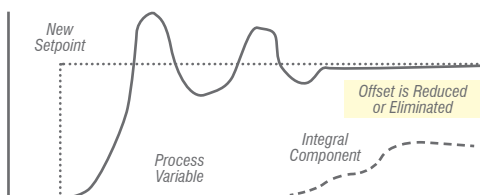
If the offset in a process is constant, it can be removed by simply adding an equal-but-opposite value, called BIAS. This is a fixed value, which is determined by the user but is not changed or operated on by the PID control. Many processes can be effectively controlled using on Proportional control and a little bias.

## Integral Control

Integral functions are added to reduce the offset error amount. The Integral function works by measuring how long an error lasts and produces an additional error value that is added into the equation. This value is tuned such that it almost completely eliminates the Proportional Offset error.

The collecting and smoothing of values over time is known as integration. Because of the integrating action, the Integral portion of the control does not take full effect until the Process Variable starts to approach a steady-state (i.e., correction value become less and less significant) value. Quick changes in error are “smoothed out” by the integrating action, and have less effect on the process. As the Process Variable approaches steady state, the Integral Error value becomes more important, and thus, serves to reduce the offset introduced by the Proportional control.

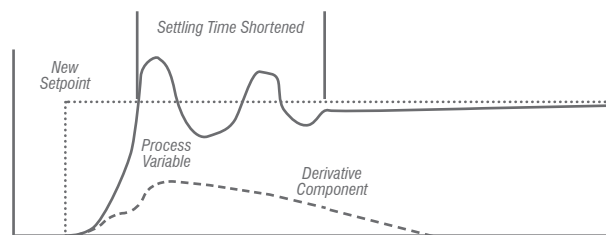
The problem with Integral control is that it does not respond well to quick changes in either the Set point or Process Variable. Although Integral control helps keep the process at a particular Set point, if either the Set point or Process Variable changes quickly, the Integral control has little effect.



Many processes respond well to Proportional-plus-Integral Control. In this case, Bias is reduced to 0 (zero).

## Derivative Control

The Derivative Control is introduced to handle quick changes in the process. The Derivative Control produces yet a third error signal based on the slope of the Error, or how much the error value changes in a given time period. When a change is first requested, the Error Slope is relatively steep and the Derivative portion of the error is significant. As the process reaches steady state the Error Slope will be shallow, and the effect of the Derivative control is reduced.

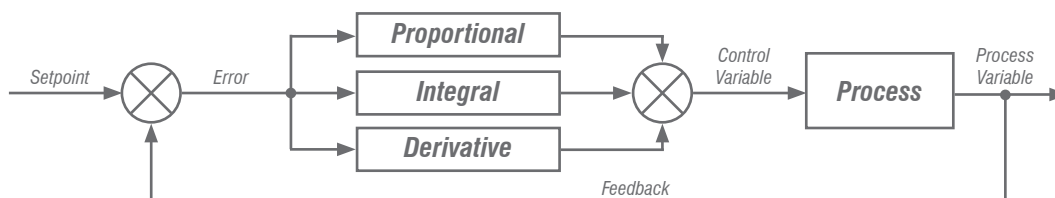


## PID

Proportional-only control is sufficient for a large number of processes, but neither Integral nor Derivative control alone is sufficient to control a process. Integral and Derivative actions support the proportional action, and respond to differing conditions in the process.

PID is an acronym for Proportional Integral Derivative. PID is a function that applies all three methods simultaneously in order to generate the controller output value.

1. Proportional function is concerned with the raw error
2. Integral function considers how long the error has been in effect
3. Derivative function takes account of how quickly the error value is changing.



When the process is first disrupted, the Proportional component attempts to make changes in the Controller Output. The Derivative aspect measures how great those changes are and adds a bit more of its value, thus making the controller act more aggressively to bring the process back to the set point. The Integral aspect has little effect here, because the error values vary greatly.

As the process comes more into control, the magnitude of the Error begins to reduce. The Proportional component is still driving the process towards the set point, but with the change in errors becoming smaller and smaller, the Derivative component begins to be reduced. The Integral component, seeing that the error value is approaching a steady state value, begins to assert itself in order to reduce the errors due to offset.

Once the process reaches steady state, the Proportional component is producing very small error values and is attempting to produce some offset value. The Integral component measures how long the Error stays at one value, and produces its own error signal to compensate. Since the rate of change in Error is small, the Derivative component is almost non-existent.

There are two common methods of implementing a PID function

1. The Independent Method
2. The ISA Method.

## 1. Independent PID

$$CV_{out} = (K_p * Error) + (K_i * Error * dt) + (K_d * Derivative) + CV_{Bias}$$

## 2. ISA PID

$$CV_{out} = K_p * (Error + (Error * dt / T_i) + (T_d * Derivative)) + CV_{Bias}$$

$dt$  = Internal elapsed time clock - previous elapsed time clock

Derivative =  $(Error - \text{previous Error})/dt$

--or--

Derivative =  $(pv - \text{previous PV})/dt$   
[User selectable during configuration].

$T_i$  = Integral time

$T_d$  = Derivative time

The Independent PID is considered the standard and is easier to tune. Although both methods provide the same results.

$CV_{Bias}$  is an additive term separate from the PID components. This is most commonly used where only the Proportional ( $K_p$ ) term is used (a proportional-only element). This forces CV Output to some non-zero value when the Process Variable (PV) is equal to the Set point (SP).  $CV_{Bias}$  is generally not used (set to 0) if the Integral term is used.

## Tuning PID Loops

The object of a PID loop is, given a change in either the Set point SP or Process Variable PV, generate a Control Variable CV such that PV is driven towards and eventually stabilized at a value equal to the SP. This is done as rapidly as possible and with minimum fluctuations about the final value.

In order to meet these goals, the PID system must be tuned. That is, proper values must be selected for  $K_p$ ,  $K_i$ , and  $K_d$  such that for any disruption in the process the process is returned to the desired value as quickly and as accurately as possible. These two requirements are usually mutually exclusive. A process can be controlled quickly but with less accuracy, or slower but more accurate. It is up to the process engineer to determine the optimal compromise between these two points, and make adjustment to the PID function (tune it) accordingly.

### PID Tuning is considered difficult.

Users often use the “trial and error” method of tuning. Adjust the  $K_p$ ,  $K_i$ , and  $K_d$  parameters and watch the process handle the next disturbance. If the control of the process is adequate, quit. Otherwise, tweak another control and try again. This process is time-consuming.

Simply put, the  $K_p$  (proportional) control is the major factor in controlling the loop. Most loops can be brought into approximate control using Proportional only. The first step is to disable the Integral and Derivative controls and bring the process into alignment using only the Proportional Control. Using Proportional only usually results in an Offset Error. That is, the actual Process Variable value differs from the Set point value by a small, relatively constant amount. If the offset is small and remains constant, it can often be cancelled using the  $CV_{Bias}$  value. Otherwise, set  $CV_{Bias}$  to 0 (zero) and try using Integral control.

The  $K_i$  (Integral) control was intended to reduce this error by adding an offset that is based on how long a specific error is present. The longer the error is present, the more effect the Integral control has. So, with the Proportional Control properly set, begin to increase the  $K_i$  until the error is minimize, if not completely eliminated.

Most processes respond well to just these two adjustments, proportional and integral. However, one can find that the PV “wobbles” too much around the final value. This is known as a damped oscillation.  $K_p$  need to be adjusted just below the point that the process begins to oscillate and goes further out of control.

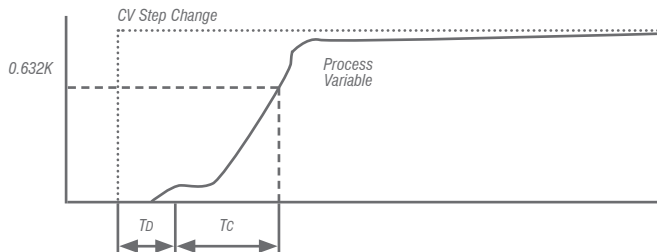
These oscillations can sometimes be further damped using the  $K_d$  (Derivative) control. The Derivative Control works on how fast the PV (and thus the resulting error) changes. The maximum rate of change occurs just after any disturbance, which is also when the  $K_p$  is oscillating. By increasing the  $K_d$ , the oscillations can be further damped to bring the process into control more quickly.

## Ziegler-Nichols Tuning Method

PID tuning depends on the user's knowledge of the process to be controlled. Kp, Ki, and Kd are determined by the processes' characteristics, which must be understood before tuning can be performed.

There are two things that must be known about the process:

1. How big is the change in Process Value when Control Value is change by a fixed amount?
2. How quickly does Process Value change in response to a change in Control Value?



The change in PV is simply measured. When compared with CV using a simple equation, the OPEN LOOP GAIN (K) of the system is obtained:

$$\text{Open Loop Gain (K)} = \text{PVstep} / \text{CVstep}$$

If a step change in CV causes an identical step change in PV, the Open Loop Gain (K) is one (unity). If a step change in CV causes a step change in PV that is less than CV, the Open Loop Gain (K) is less than 1. If a small step change in CV causes a large change in PV, the Open Loop Gain (K) is greater than 1.

Most processes won't see any change in PV for some time after CV changes. This is called Pipeline Delay Time (Tp) or Dead Time. (Not to be confused with DEAD BAND.)

The Time Constant (Tc) of the process is defined as the time between when the PV first starts to change and the time when PV reaches 63.2% of the expected final PV value.

Find K and Tc

Some experimenting must be done in order to obtain the desired values. This is best done by placing the PID Element into the MANUAL mode, make a small change in CV, and then plot the change in PV. For slow processes this can be done manually, but a strip chart recorder might be helpful.

The change in CV is large enough to cause a measurable change in PV but not so large as to completely disrupt the process being controlled.

The plot looks similar to the above graphic, and K, Tc, and Tp are easily measurable.

Tune the Process

If K, Tc, and Tp are known we can use the following equations can be used to estimate starting values for Kp, Ki, and Kd in a Proportional / Integral / Derivative (PID) control:

$$K_p = (1.2 * T_c) / (K * T_p)$$

$$K_i = (0.6 * T_c) / (K * T_p * T_p)$$

$$K_d = (0.6 * T_c) / K$$

Tc and Tp are time units. It is important to ensure that both are expressed in identical units (i.e., milliseconds, seconds, hours, or whatever time frame is appropriate to the process). However, for use in the i<sup>3</sup> Configurator PID TUNE dialog, these values must be expressed 10mS intervals (e.g.: "100" = 10mS \* 100 = 1 second).

If Proportional-only control (Ki and Kd = 0) is desired, use the equation:

$$K_p = T_c / (K * T_p)$$

Or for Proportional / Integral control (Kd = 0), use the equations:

$$K_p = 0.9 * T_c / (K * T_p)$$

$$K_i = 0.3 * K_p / T_p$$

These equations are known as the Ziegler-Nichols tuning method, which were developed by John Zeigler and Nathaniel Nichols in the 1940's.

## Programming PID Functions

We are going to write two programs that use the PID functions to control a heater to a set temperature. The only difference in the two programs will be that the first will have an analogue output and the second will have a relay digital output. Both programs will read an analogue signal in that represents a temperature. The user will be able to enter a set point and view the current value.

## Setting up the PID Function Block

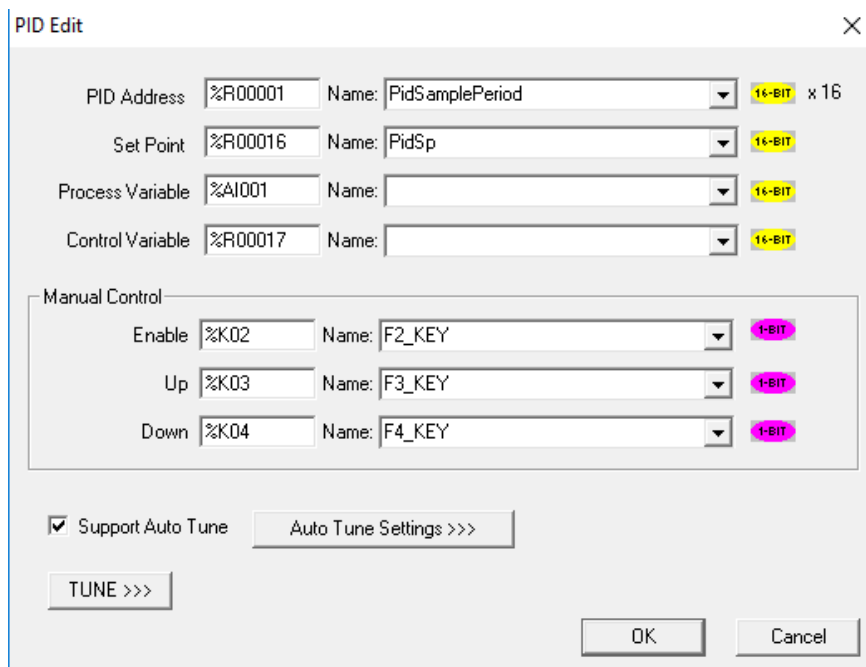
There are two different types of PID Independent and ISA. The only difference is the equation they perform. We will use the Independent function block.

### Independent PID

$$CV_{out} = (K_p * Error) + (K_i * Error * dt) + (K_d * Derivative) + CV_{Bias}$$

### ISA PID

$$CV_{out} = K_p * (Error + (Error * \frac{dr}{dt}) + (T_d * Derivative)) + CV_{Bias}$$



The PID Edit dialog box contains the following fields and controls:

- PID Address:** %R00001, Name: PidSamplePeriod, 16-BIT x 16
- Set Point:** %R00016, Name: PidSp, 16-BIT
- Process Variable:** %AI001, Name: , 16-BIT
- Control Variable:** %R00017, Name: , 16-BIT
- Manual Control:**
  - Enable:** %K02, Name: F2\_KEY, 1-BIT
  - Up:** %K03, Name: F3\_KEY, 1-BIT
  - Down:** %K04, Name: F4\_KEY, 1-BIT
- ☒ **Support Auto Tune** (Auto Tune Settings >>> button)
- TUNE >>>** button
- OK** and **Cancel** buttons

<b>PID Address</b>	This is the base address of fifteen consecutive WORD (%Rxxxx) registers that the PID element uses to store its parameters.
<b>Setpoint</b>	This is the location of the User-defined Process set-point value. This value cannot be a decimal constant
<b>Process Variable</b>	This is the location (typically %AI) of the Process Variable value coming in from the process. This value cannot be a decimal constant.
<b>Control Variable</b>	This is the location (typically %AQ) of the Control Variable value going out to the process.
<b>Manual Input</b>	This register is a Boolean register, presumably %T. This switches off the PID control and the CV is updated by the Up and Down Input.
<b>Up Input</b>	This register is a Boolean register, presumably %T. Used to UP the CV in manual mode
<b>Down Input</b>	This register is a Boolean register, presumably %T. Used to down the CV in manual mode.
<b>Support Auto Tune</b>	Select this option to enable support for auto tune PID.
<b>Auto Tune Settings</b>	Click on this button to set the registers and options used for auto tuning. See PID Auto Tune for more information
<b>Tune</b>	Click on the TUNE button to invoke the PID Element Tuning Dialog

The initial PID Address specified is only the starting register and the following consecutive 15 registers are used. The registers are the same for both PID functions.

Each register has a specific parameter setting.

Offset	Parameter	Units	Range	Description
0	Sample Period	10mS	0 to 65535	The shortest time, in 10mS increments, allowed between PID solutions.
1	Dead Band +ve	PV Counts	0 to 32000	Defines the Upper and Lower Dead Band limits in terms of PV counts. Set both to 0 (zero) if no dead band is required. Both should be set to 0 (zero) until the PID is tuned.
2	Dead Band -ve	PV Counts	0 to 32000	A Dead Band might then be necessary to prevent small changes in CV values due to slight variations in error.
3	Proportional Gain (Kp)	Percent	0 to 327.67%	Sets the Proportional Gain (Kp) factor in terms of percent. 100 sets unity gain (gain of 1).
4	Derivative Gain (Kd)	10mS	0 to 327.67 seconds	Entered as a time with a resolution of 10 mS. In the PID equation this has the effect: $K_d * \Delta \text{Error} / dt$ .
5	Integral Rate (Ki)	Repeats per 1000 second	0 to 32.767 repeats per second	Entered as a number of repeats per second, effectively the integration rate. In the PID equation this has the effect: $K_i * \text{Error} * dt$ .
6	CV Bias	CV Counts	-32000 to +32000	Number of CV counts added to the output before the rate and amplitude clamps.
7	CV Upper Clamp	CV Counts	-32000 to +32000	Number of CV Counts that represent the highest and lowest value for CV. CV Upper Clamp must be more positive the CV Lower Clamp.
8	CV Lower Clamp	CV Counts	-32000 to +32000	Setting these to 0 and 1, can allow us to use a relay output.
9	Minimum Slew Time	Seconds of full travel	0 to 32000 seconds to move 32000 CV counts	Determines how fast the CV value can change.
10	Config Word	N/A	N/A	<b>Internal Use - Do not modify this value.</b>
11	Manual Command	CV Counts	Tracks CV in Auto mode; sets CV in Manual Mode	In the Automatic mode this register tracks the CV value. In the Manual Mode, this register contains the value that is output to the CV within the clamp and slew limits.
12	Internal SP	N/A	N/A	Tracks SP in . Used by iHMI
13	Internal PV	N/A	N/A	Tracks PV in . Used by iHMI
14	Internal CV	N/A	N/A	Tracks CV out . Used by iHMI

Each PID element must use a distinctly separate Reference Array, even if the values are identical to an exiting PID element. There can be no overlapping of PID elements. i.e. the same register should not be reused. There is however no limit on how many PID functions can be used only the memory limitations (9999 registers).

Registers at offset 0 through 9 must be configured before the PID element is used and can be these registers can be manipulated by the ladder program as well.

The easiest method of configuring a PID element is to Auto-Tune the device. This can take a lot of time however and requires the i3 to be situated in the application. These registers can, however, be manipulate by the ladder program as well.

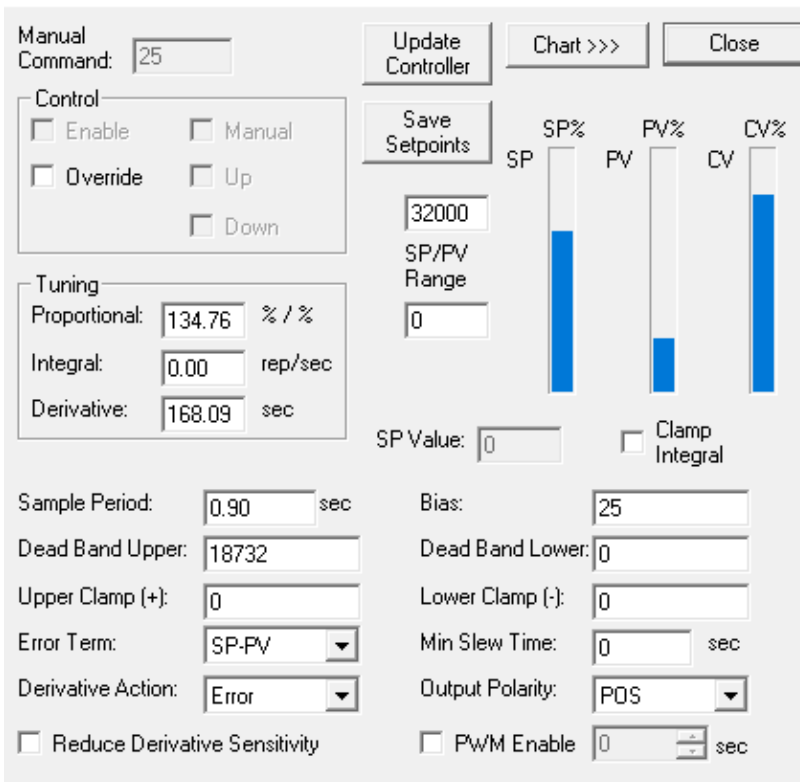
## Tuning the Function Block

When the tune input goes high, the PID function starts the auto tune process. If power is removed from the tune input before the tuning process is complete, the auto tune procedure is aborted. Note to eliminate the anti-windup feature of the PID loop, the integral term is set to zero when the auto tune starts.

When the auto tune process completes, the done output goes high and the PID loop continues running with the newly calculated coefficients.

The PID Tuning Dialog allows the PID loop to be tuned. Once acceptable values are determined, these values can be moved into the PID Element registers using the First Scan (FST\_SCN, %S01) bit and Register Move elements or by using the downloadable set points.

Tune PID Function - Offline



Manual Command: 25

Control: ☐ Enable ☐ Manual ☐ Override ☐ Up ☐ Down

Tuning: Proportional: 134.76 % / % Integral: 0.00 rep/sec Derivative: 168.09 sec

Update Controller Chart >>> Close

Save Setpoints

SP 32000 SP/PV Range 0

SP Value: 0 ☐ Clamp Integral

Sample Period: 0.90 sec Bias: 25

Dead Band Upper: 18732 Dead Band Lower: 0

Upper Clamp (+): 0 Lower Clamp (-): 0

Error Term: SP-PV Min Slew Time: 0 sec

Derivative Action: Error Output Polarity: POS

☐ Reduce Derivative Sensitivity ☐ PWM Enable 0 sec

Usually unchecked.

If checked MANUAL and SP VALUE boxes are enabled. This allows for modifications for tuning purposes.

The controller does not get these inputs from as specified in the function block, but uses the values entered from this tuning screen.

After any of these items is changed, the UPDATE CONTROLLER button must be pressed to send the new data to the controller.

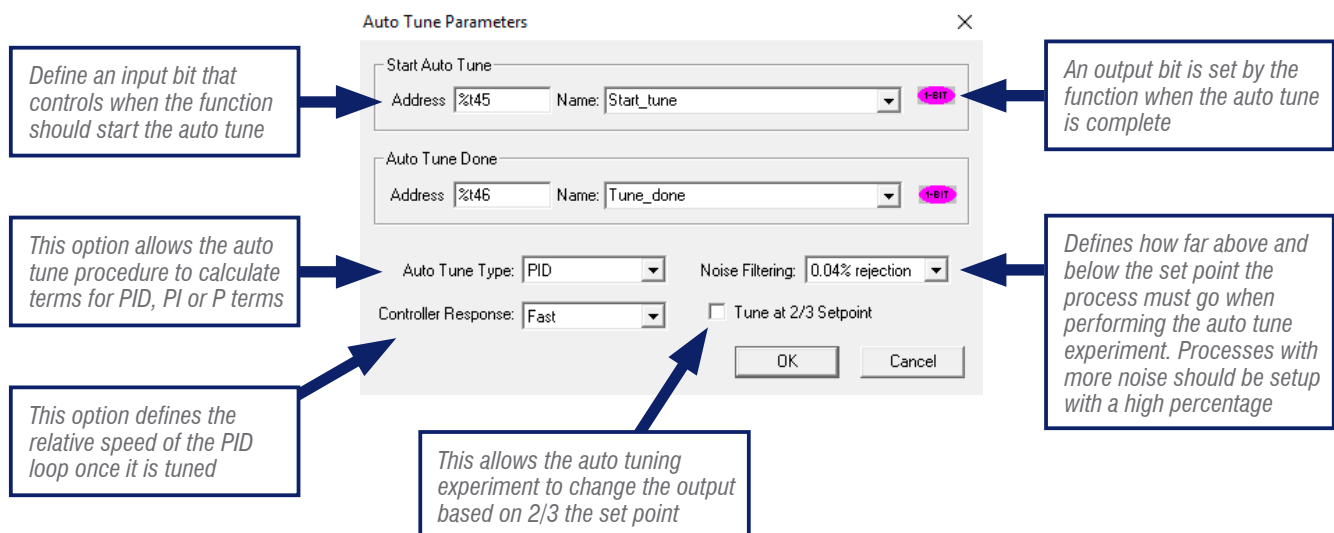
## Tuning Operation

A Boolean AUTOTUNE input starts the auto tuning cycle and needs to remain high until the cycle has been completed otherwise the tuning will stop. Once the cycle has been completed the auto tune done output will go high. Once the cycle has been completed the input can be removed and the done output will reset.

At the conclusion of the AUTOTUNE cycle, the specified controller coefficients are updated and the AUTOTUNE DONE output from the block is set to true. The PID block now reverts to the previous state, either automatic or manual. The block will then be ready for another auto tune cycle. The new tuning coefficients are available in their respective registers.

## Auto-Tune Parameters

The auto tuning function is an addition to the PID blocks and all other parameters are the same. Auto tune PID allows the PID controller to perform an experiment on your process and use the results to calculate PID coefficients that match your process and the desired PID operation. When auto tune PID is enabled additional parameters need to be set.



Start Auto Tune

Address: %t45 Name: Start\_tune 1-BIT

Auto Tune Done

Address: %t46 Name: Tune\_done 1-BIT

Auto Tune Type: PID Noise Filtering: 0.04% rejection

Controller Response: Fast ☐ Tune at 2/3 Setpoint

OK Cancel

Define an input bit that controls when the function should start the auto tune

An output bit is set by the function when the auto tune is complete

This option allows the auto tune procedure to calculate terms for PID, PI or P terms

Defines how far above and below the set point the process must go when performing the auto tune experiment. Processes with more noise should be setup with a high percentage

This option defines the relative speed of the PID loop once it is tuned

This allows the auto tuning experiment to change the output based on 2/3 the set point

## Using the Auto Tune Function

Prior to auto tuning it is necessary to partially set up the PID block as before. Specifically, the following parameters need to be set correctly:

- Set point
- Error Term
- Sample Period
- Output Polarity
- Upper and Lower Clamp

Please note the previous values of the proportional, integral, and derivative coefficients do not affect the results of auto tuning.

The default settings for the auto tune cycle produce Proportional, Integral, and Derivative coefficients using the standard Ziegler-Nichols rules. This is suitable for many typical processes with delay and one or two equal lags and with a fairly quiet process variable.

Non-default settings may be selected to improve the auto tuning behaviour in certain circumstances. These selections only affect the generation of auto tuning coefficients.

The controller type field defaults to PID but can be set to PI, Proportional/Integral, or just P, Proportional control.

- PI control tends to be more stable with processes that do not have any delay, just lags.
- Full PID control can give better response for processes with delay.

The full PID tuning rules assume that the process has a moderate delay and may not be suitable for other processes. These modes are produced by the auto-tuning algorithm by setting the unused coefficients to zero. These may subsequently be manually increased to enable the other modes.

The response field can be used to increase the controller damping to decrease overshoot and ringing. For a typical Ziegler-Nichols process

- FAST response produces some overshoot and a 4:1 decay ring down.
- MEDIUM produces a slight overshoot.
- SLOW produces no overshoot.
- With processes that are outside the optimum range for Zeigler-Nichols rules, the VERY SLOW response may be necessary for adequate response.

During auto tuning the algorithm detects the process passing above and below the active set point. Hysteresis is applied to the set point to avoid false indications due to process noise. The default hysteresis band is 0.04% of full scale. For noisy processes, this may need to be increased for proper auto tuning. The NOISE SUPPRESSION setting results in the following noise rejection values.

Higher noise rejection values also cause the auto-tuning algorithm to select somewhat slower, more stable coefficients. For noisy processes, it is also recommended that PI rather than PID control be selected.


## How Auto Tuning Works

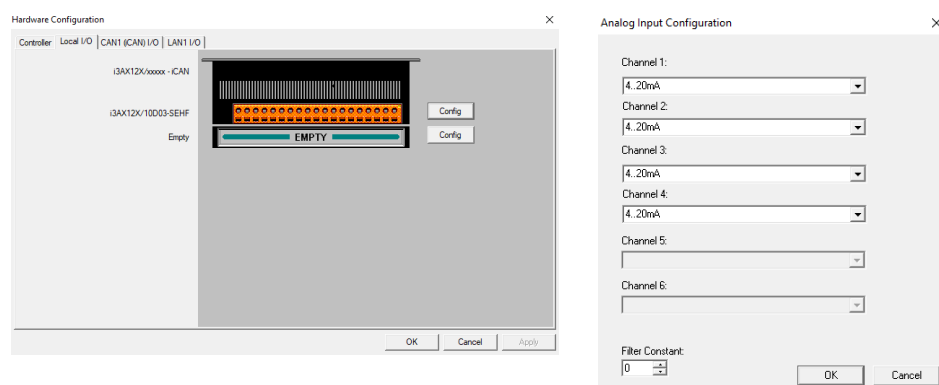
The auto tuning function block performs an experiment on the process to be controlled and uses the results to calculate the PID coefficients. While auto tuning the output is moved back and forth between the upper and lower clamps. The time for the process to move from a percentage (based on noise filtering) above and below the set point is recorded along with overshoot and undershoot readings. Once this experiment is complete, the data collected is used to calculate the new PID coefficients.

## Program 1. PID Digital Output

In this example we are going to use the PID function to control a Relay output on the i<sup>3</sup>. This could then be used to switch on a simple heating element in a Temperature control system.

### Configure the I/O

Connect the i<sup>3</sup> (either a i3AX12X/10B04-SEHF or a i3AX12X/10D03-SEHF) relay model to the computer and click on the icon  to configure the I/O. Select the Auto configure button so that the correct i3 is selected, then click on the Config button to configure the Module Setup.




The image shows two screenshots from the IMO software. The left screenshot is the 'Hardware Configuration' window, which displays a list of modules: 'i3AX12X/xxxx - CAN', 'i3AX12X/10D03-SEHF', and 'Empty'. The 'i3AX12X/10D03-SEHF' module is highlighted, and a 'Config' button is visible next to it. The right screenshot is the 'Analog Input Configuration' window, which shows six channels. Channels 1 through 4 are set to '4...20mA', while Channels 5 and 6 are empty. A 'Filter Constant' is set to 0. The 'OK' and 'Cancel' buttons are at the bottom.

The relay model i<sup>3</sup> 's doesn't have a thermocouple input, so I have used one of IMO's TCHead Thermocouple transmitters to covert the signal to mA's.

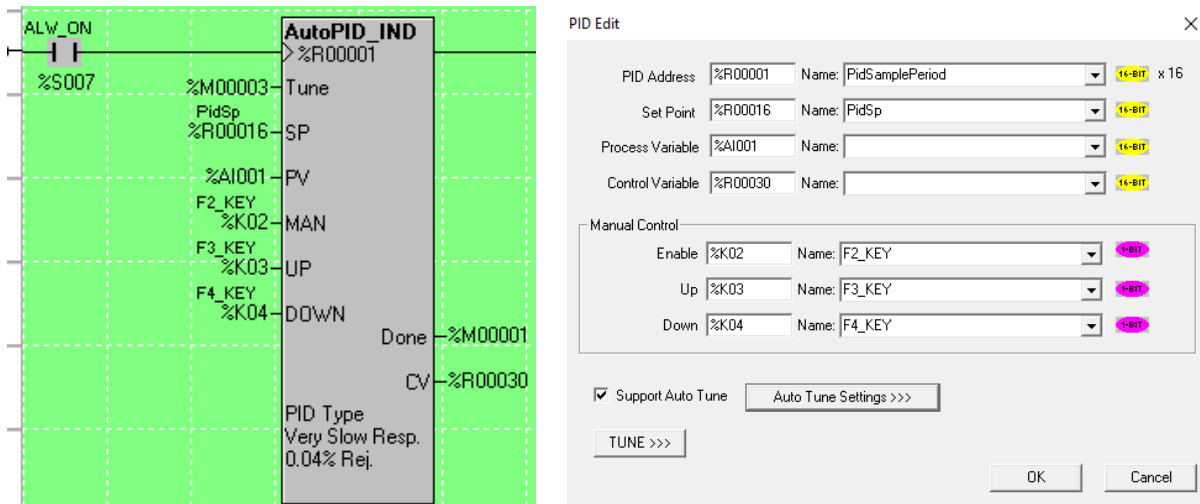


## Ladder Logic Programming

To manipulate the digital output, we are going to clamp the upper and lower CV limits to 10 and 0. The PID Function will manipulate the CV value between 0 and 10 to control the process. This CV output will be inputted to a compare function block and if it is greater than 5 the Q1 will be high, if it is less 5 the Q1 will be off. There are two ways to adjust the clamp values, the first is to enter the values in the associated registers directly and the second is to edit the clamp values through the "Tune" menu.

Insert a NO contact and assign it to "always on" %S7. We will use this to power the PID function. Next select the PIDIND function  from the Special Operations menu and insert it into the ladder logic.

Double click on the function to open the properties page and set up the following values.



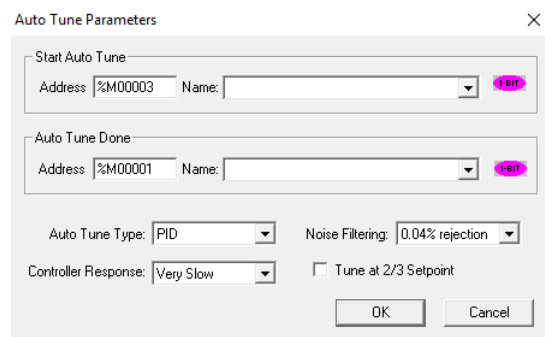
The screenshot shows the 'PID Edit' dialog box on the right and the 'AutoPID\_IND' function block in the ladder logic on the left. The dialog box has the following settings:

- PID Address: %R00001, Name: PidSamplePeriod, 16-BIT x 16
- Set Point: %R00016, Name: PidSp, 16-BIT
- Process Variable: %AI001, Name: , 16-BIT
- Control Variable: %R00030, Name: , 16-BIT
- Manual Control:
  - Enable: %K02, Name: F2\_KEY, 1-BIT
  - Up: %K03, Name: F3\_KEY, 1-BIT
  - Down: %K04, Name: F4\_KEY, 1-BIT
- ☒ Support Auto Tune, Auto Tune Settings >>>
- TUNE >>> button
- OK and Cancel buttons

The 'AutoPID\_IND' function block in the ladder logic has the following settings:

- Done: %M00001
- CV: %R00030
- PID Type: Very Slow Resp. 0.04% Rej.

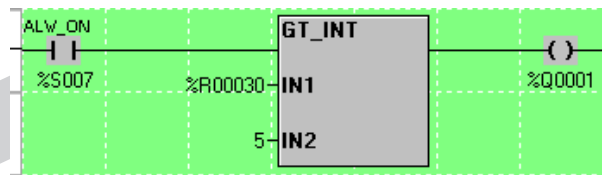
Set up the Auto Tune by clicking on the "Auto Tune Settings" button.



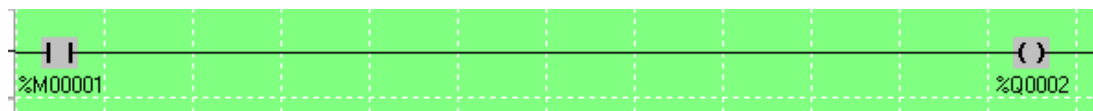
The 'Auto Tune Parameters' dialog box has the following settings:

- Start Auto Tune: Address %M00003, Name: , 1-BIT
- Auto Tune Done: Address %M00001, Name: , 1-BIT
- Auto Tune Type: PID
- Noise Filtering: 0.04% rejection
- Controller Response: Very Slow
- ☐ Tune at 2/3 Setpoint
- OK and Cancel buttons

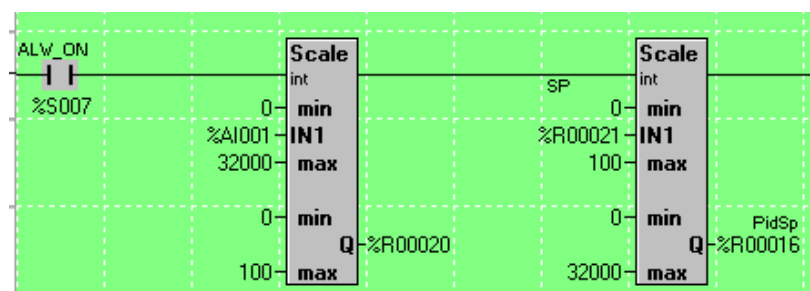
We will then use the output CV from the PID function block in a Greater Than or Equal to function. If it is greater than or equal to 5 then the output will be high, if it is less than 5 then the output will be low. Assign a NO coil at the end of the GE function to %Q0001.



Add an indicator to the signal when the tuning process has finished. Assign %M1 to a NO contact switching on an output coil %Q2.



To make the screen more user friendly we are going to scale our set point and present value analogue input from 0-32000 to 0-100 degrees Celsius.



## Screen Editor Programming

On the user screen we need to have a button to switch the auto-tuning feature on, and a lamp to indicate when it has finished. We will also display the raw analogue input value (0-32000) and the scaled value (0-100°C), as well as having a data entry for the scaled Set Point.

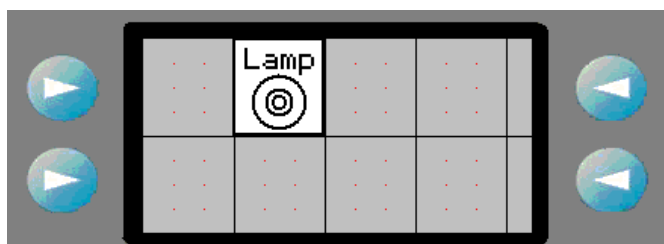
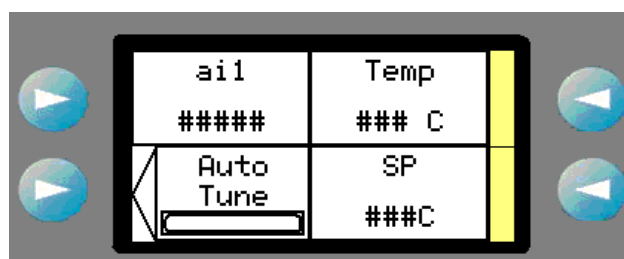
To achieve this, we need to insert 3 numeric data functions, making only the Set Point editable. Enter the functions by clicking on the icon . Click them onto the centre of the screen and then move them to where you wish them to be positioned on the screen. Double click on the boxes to enter the properties of each function.

Address	Legend	Digits	Units
%Ai1	Ai1	5 digits	
%R20	Temp	3 digits	*C
%R21	SP	3 digits	*C

For the tuning features assign one button to start / stop the tuning and a lamp to %M1.

Address	Legend	Type
%M1	Auto-tune	Toggle Button
%M2	None	Round Lamp

The screen should now look similar to the one below.



## Running the Program

Download the program into the i<sup>3</sup> as normal and the program will run as expected if you have calculated the P, I & D values. If not by double clicking on the PID IND function to open up its properties, you can now select to TUNE the i<sup>3</sup> controller.

PID Edit

PID Address

%R00001

Name: PidSamplePeriod

16-BIT x 16

Set Point

%R00016

Name: PidSp

16-BIT

Process Variable

%Ai001

Name:

16-BIT

Control Variable

%R00030

Name:

16-BIT

Manual Control

Enable

%K02

Name: F2\_KEY

1-BIT

Up

%K03

Name: F3\_KEY

1-BIT

Down

%K04

Name: F4\_KEY

1-BIT

☒ Support Auto Tune
 

Auto Tune Settings >>>

TUNE >>>

OK

Cancel

**Tune PID Function** [X]

Manual Command:

[Update Controller] [Chart >>>] [Close]

Control

☐ Enable ☐ Manual

☐ Override ☐ Up ☐ Down

Tuning

Proportional:  % / %

Integral:  rep/sec

Derivative:  sec

Save Setpoints

SP:  SP/PV Range:

SP%  PV%  CV%

SP Value:  ☐ Clamp Integral

Sample Period:  sec Bias:

Dead Band Upper:  Dead Band Lower:

Upper Clamp (+):  Lower Clamp (-):

Error Term:  Min Slew Time:  sec

Derivative Action:

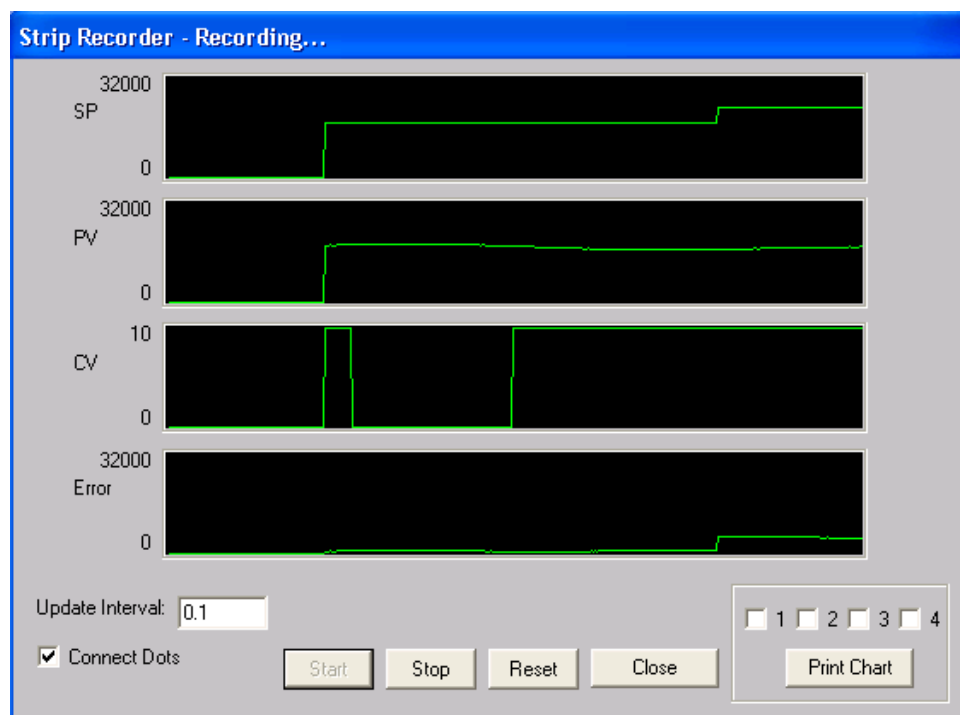
☐ Reduce Derivative Sensitivity ☐ PWM Enable  sec

New values can be entered for the P, I & D then click "Update Controller" to send them to the i<sup>3</sup>.

The sample period, dead bands and all of the other 0-15 registers used can be altered here then the controller can be updated.

We have clamped the output to between 0 and 10

Monitoring the process is also made easy as the SP, PV and CV values can all be tracked as well as the error signal. These graphs can also be printed out.



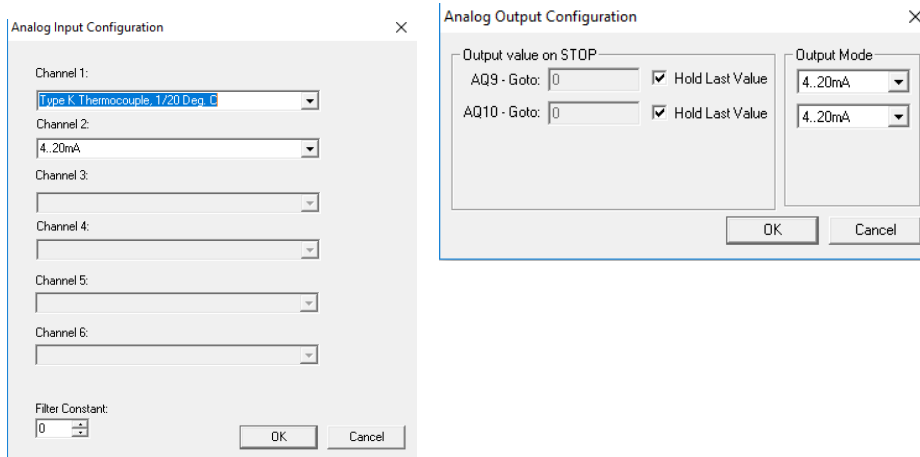
Please see PIDExample-relay-op.csp program file for the i<sup>3</sup>.

## Program 2. PID Analogue Output

Using the i3AX12X/13C14-SEHF with its thermocouple input and Current or Voltage analogue output we can repeat example 1 but provide the output with a more accurate PID control.

### Configure the I/O

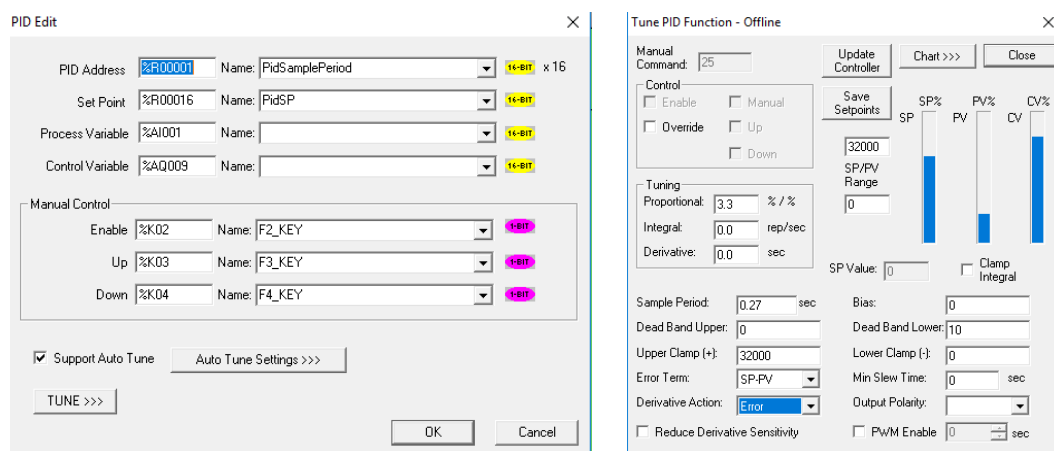
Just as before, connect the i<sup>3</sup> and Auto-Configure the unit. This time however we are going to configure Ai1 to a K-type thermocouple and the Aq9 (1st analogue output) to a 4-20mA signal.



The image shows two configuration windows. The 'Analog Input Configuration' window on the left shows Channel 1 set to 'Type K Thermocouple, 1/20 Deg. C'. The 'Analog Output Configuration' window on the right shows 'Output value on STOP' for Aq9 and Aq10 set to 'Hold Last Value' and 'Output Mode' set to '4..20mA'.

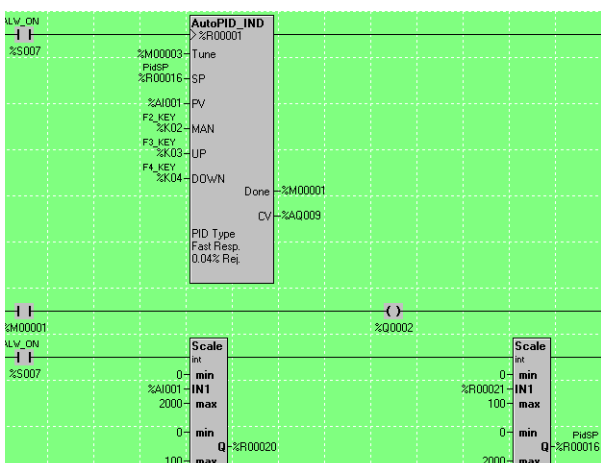
## Ladder Logic Programming

We just need to change the output from the PID function to the analogue output address %Aq9 and adjust the clamps to have the full range 0-32000.



The image shows two windows. The 'PID Edit' window on the left shows PID Address %R00001, Set Point %R00016, Process Variable %AI001, and Control Variable %AQ009. The 'Tune PID Function - Offline' window on the right shows tuning parameters: Proportional 3.3, Integral 0.0, Derivative 0.0, and various clamps and settings.

We also have no need for the GE function and we need to adjust the scaling function blocks. The code should now look like the one below.

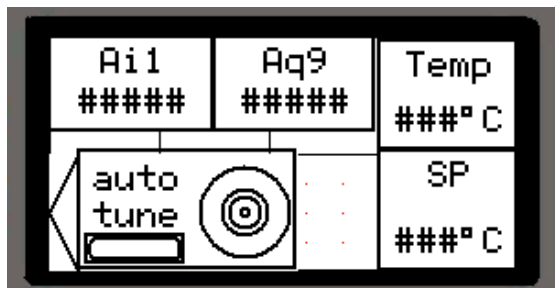


# i<sup>3</sup> PID Tutorial

## Screen Editor Programming

The only change we are going to have on the screen is to add the true register value of the output. Enter a new numeric function and assign it to %AQ9, giving it the legend "Aq9".

The screen should now look similar to the one below.



## Running the Program

The program will operate in a very similar fashion and the tuning functions will still apply. The only difference will be the output will be more controlled and accurate.

The screenshot shows the 'Tune PID Function' dialog box. It has a 'Manual Command' field set to '9740'. The 'Control' section has 'Enable' checked. The 'Tuning' section has 'Proportional' set to '65.12 % / %', 'Integral' set to '0.260 rep/sec', and 'Derivative' set to '327.67 sec'. The 'SP Value' is '620'. The 'Output Polarity' is 'POS'. A callout box points to the 'Tuning' section with the text: 'After Auto-tuning the newly found values of the P, I and D values have been inserted into the registers.'

After Auto-tuning the newly found values of the P, I and D values have been inserted into the registers.



[www.imopc.com](http://www.imopc.com)

## IMO Precision Controls Ltd

Unit 3, The Interchange, Frobisher Way  
Hatfield, Hertfordshire AL10 9TG UK

Tel: +44 (0)1707 414 444  
Fax: +44 (0)1707 414 445

Email: [sales@imopc.com](mailto:sales@imopc.com)  
Web: [www.imopc.com](http://www.imopc.com)



Bespoke technical training courses  
available at our Hatfield training facility.  
Call 01707 414 444 for more information