QuickBuilder PID Reference

ctc
Control Technology Corp.

Table of Contents

QuickBuilder PID Reference Guide

Control Technology Corp.
25 South Street
Hopkinton, MA 01748
Phone: 508.435.9595 • Fax 508.435.2373

Document No. 951-530031-006

⚠ **WARNING:** Use of CTC Controllers and software is to be done only by experienced and qualified personnel who are responsible for the application and use of control equipment like the CTC controllers. These individuals must satisfy themselves that all necessary steps have been taken to assure that each application and use meets all performance and safety requirements, including any applicable laws, regulations, codes and/or standards. The information in this document is given as a general guide and all examples are for illustrative purposes only and are not intended for use in the actual application of CTC product. CTC products are not designed, sold, or marketed for use in any particular application or installation; this responsibility resides solely with the user. CTC does not assume any responsibility or liability, intellectual or otherwise for the use of CTC products.

The information in this document is current as of the following Hardware and Firmware revision levels. Some features may not be supported in earlier revisions. See www.ctc-control.com for the availability of firmware updates or contact CTC Technical Support.

Model Number	QuickBuilder Revision	Firmware Revision
5300	>=1.2.2596	>= 5.00.90.R63

1   Chapter 1: Overview

This document details the operation of QuickBuilder's PID object. The main purpose of this guide is to document the PID loop object used by QuickBuilder on Blue Fusion 5300 series controllers so that experienced users can best apply it in their applications. PID objects are set up using the QuickBuilder Automation Suite and then downloaded to a Blue Fusion Model 5300 controller. The PID object allows the Model 5300 automation controller to precisely control temperature, pressure, flow or even simple motion applications. (Note: for most motion control applications, CTC recommends using a dedicated motion module such as the M3-40 series).

What is a PID loop?

The term PID stands for **P**roportional plus **I**ntegral and **D**erivative control. The PID control loop is ideal for applications where a desired setpoint value needs to be accurately maintained by the output (known as a "Process Variable") of a control system even when the control system experiences load disturbances and / or measurement noise. And, most importantly for industrial applications, a PID loop when properly tuned will reduce the error between the setpoint and the process variable in the minimum possible time.

The PID loop does this by measuring the output of the process via some type of feedback sensor and then calculating the difference (error) between the output and the setpoint. If an error exists, the controller tries to minimize this error by adjusting the output to bring the process closer to the desired setpoint.

PID loops are calculated repetitively at precise intervals and are able to use the history of error information measured during previous cycles to determine how best to adjust the output in the current cycle. The way in which the PID loop parameters are set up will determine how the loop responds to a measured error. If the loop parameters are set too aggressively (under damped), it may cause the process to become unstable and oscillate. If the loop parameters are not aggressive enough (over damped), the system may require too much time to return to the setpoint.

The following diagram shows how a basic PID loop is calculated.
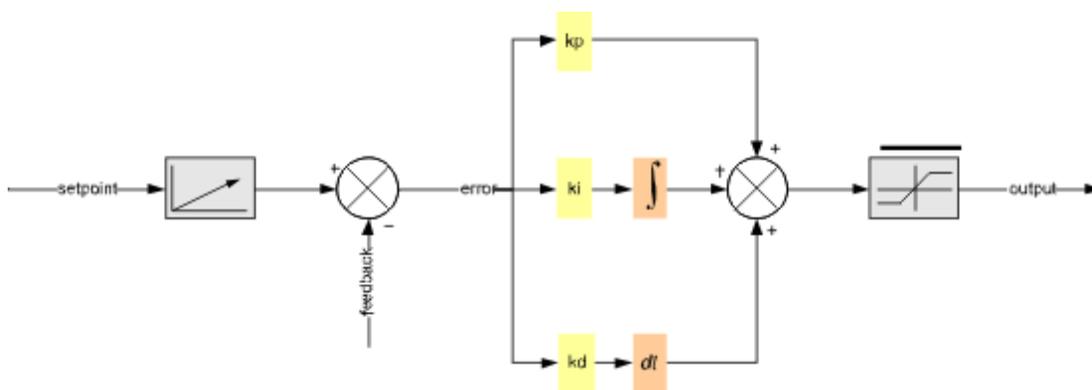
As can be seen from this diagram, the aggressiveness of the output response is directly controlled by the P, I, and D factors.

> The **Proportional** factor gives an immediate response that is directly proportional to the error. The larger in magnitude kp is, the greater the response to an error.

> The **Integral** factor is the term that allows the PID loop to eliminate steady-state errors. Increasing the value for ki will allow the error to be eliminated more quickly, but may also result in overshoot of the desired setpoint value.

> The **Derivative** factor gives the PID loop a "forward looking" input since it is based on the slope of the error. A larger value for kd will reduce overshoot and settling time, but will also make the system less responsive to short term disturbances.

Setting and adjusting the PID parameters is called tuning. While systems can be successfully tuned by trial and error, better results are obtained by personnel experienced with both the process to be controlled and PID loop tuning methods. For more information on tuning, see Appendix A.

2 Chapter 2: The QB PID Object

2.1 Features

The PID Object in QuickBuilder allows users to set up very sophisticated high performance PID loops on the Model 5300 automation controller. The QB PID Object has many advantages over more simplistic PID loop implementations. Some of the key features are outlined below:

Feature	Benefit
Up to 256 PID loops / CPU	The Model 5300 can tackle even the most demanding applications
Advanced PID loop equation	CTC uses a state-of-the-art loop equation with more than 20 settable parameters and multiple alarms and status outputs. This allows the QB PID Object to solve a wide variety of applications automatically without adding auxiliary control logic to the project.
Floating point calculations	Using 64-bit floating point calculations ensures the most precise results, yielding improved loop response.
Timing accuracy < 200 nanoseconds	Ensures quick response and fast returns to steady state conditions.
Fast loop update	User settable down to 1ms, it allows the QB PID loop to be used for a wider variety of applications.

Feature	Benefit
Individually settable loop update rates	Ability to optimize each loop independently. Also allows more control over CPU utilization.
Loops implemented as QB objects	Loops do not consume user memory, QuickStep steps, or user variables. Loops run automatically as background tasks and do not decrease the user task limit.
Table-driven setup	Simplifies setup process. No need to code PID initialization steps.
Multiple properties	Easily set up and customize PID loops for a wide variety of applications.
Properties changeable on-the-fly	Allows the QuickBuilder program to adjust loop behavior based on external events.
Multiple status and alarm parameters	Eliminates the need to code these items separately, saving time and resources. Also provides faster notification.
Programmable deadband	Eliminates excessive dithering around a setpoint.
Multiple modes	Easily set up manual, automatic, or cascaded control loops.

2.2 PID Loop Algorithm

The actual loop algorithm used by the PID object is shown in the diagrams below. In the next section we will explain the process of adding a PID loop to a controller. Following the setup section we will define all of the parameters and variables shown in the PID diagram.
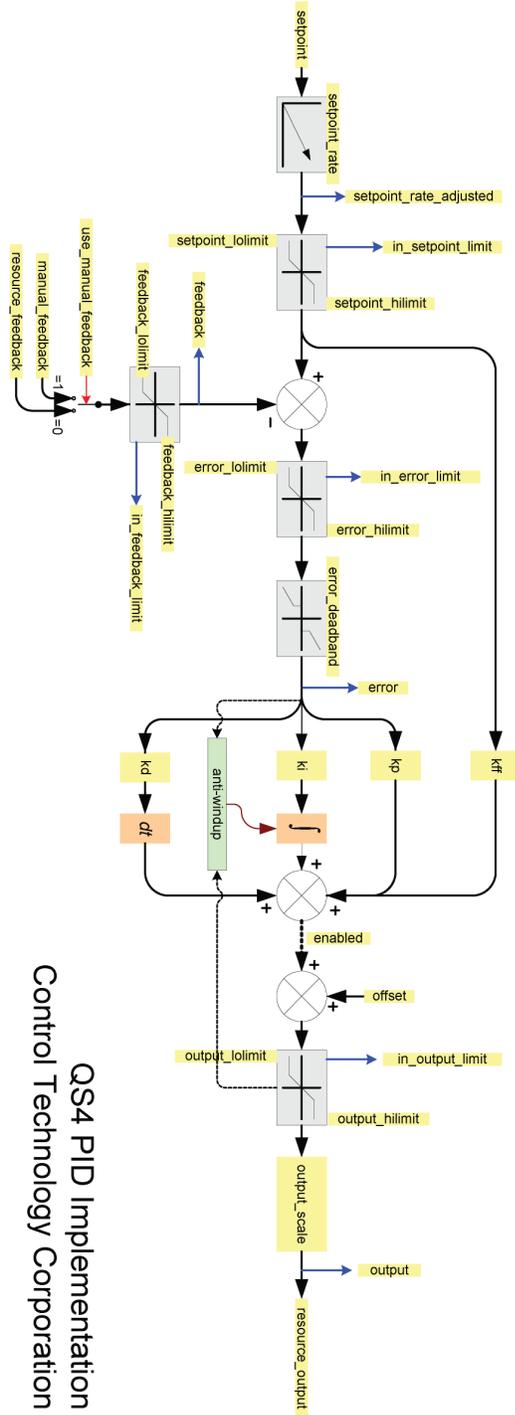
QS4 PID Implementation
Control Technology Corporation

2.3 PID Object Setup

Adding a PID object to your QuickBuilder project is easy. PID objects are associated and linked to Model 5300 CPU processors. To add a PID loop, simply click on the controller icon in the Resources window to select the destination controller for the PID loop.
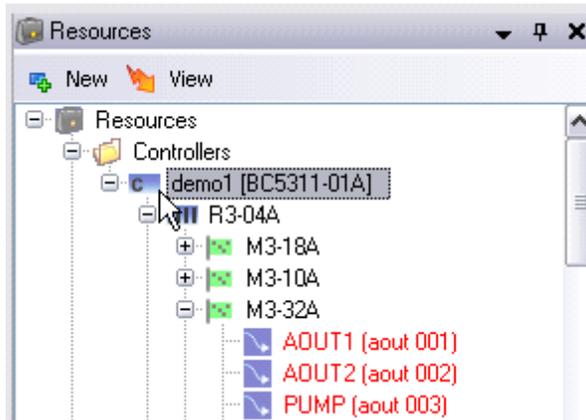
Then go to the **New** menu item and add a PID Object.
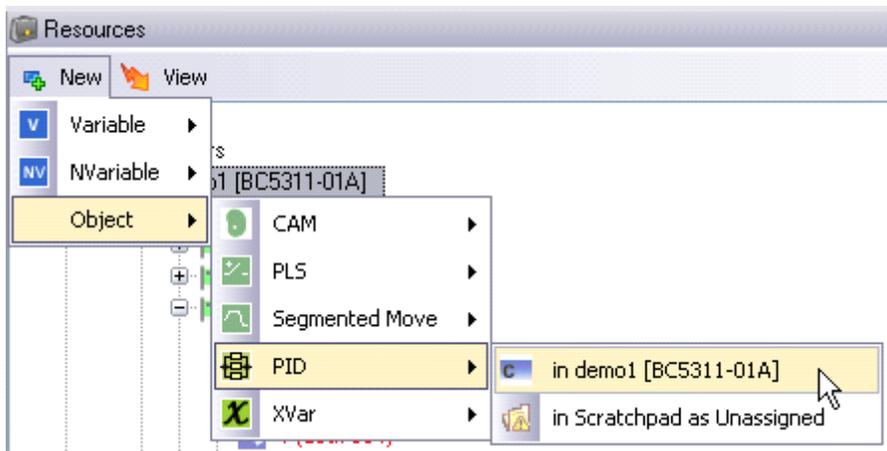
At this point a new PID loop object will be added to the selected controller. The PID object should now be given a meaningful name and then you will be ready to set up its parameters.
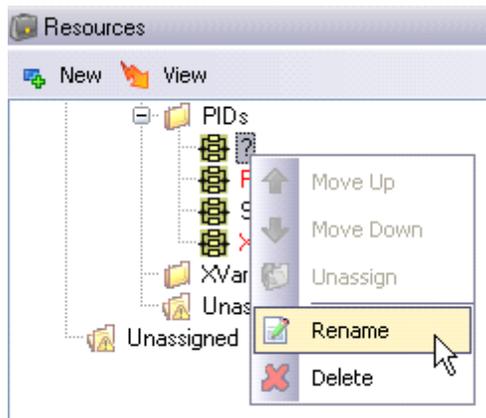
For this example, we will name the PID object "New_PID." Just like the other resources in the Resource window, when you highlight a PID Object, its properties are automatically brought up in the Properties window. The screen capture on the following page shows the Properties window for our New_PID.

You will notice that all of the Object Properties are preloaded with default values except for *resource_feedback* and *resource_output*. These properties must be associated with actual controller resources, and a pop-up selection window is provided for this purpose.
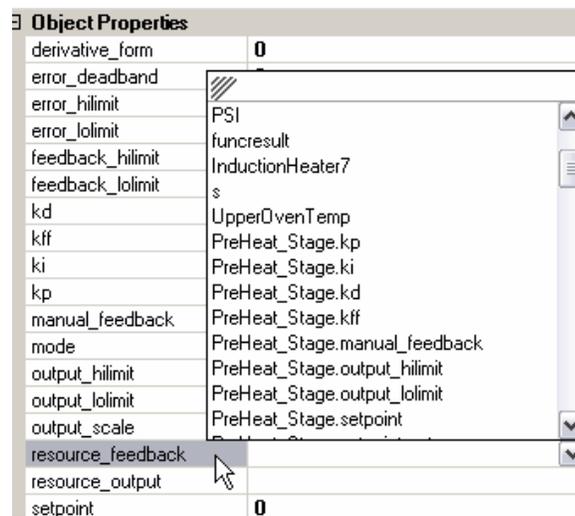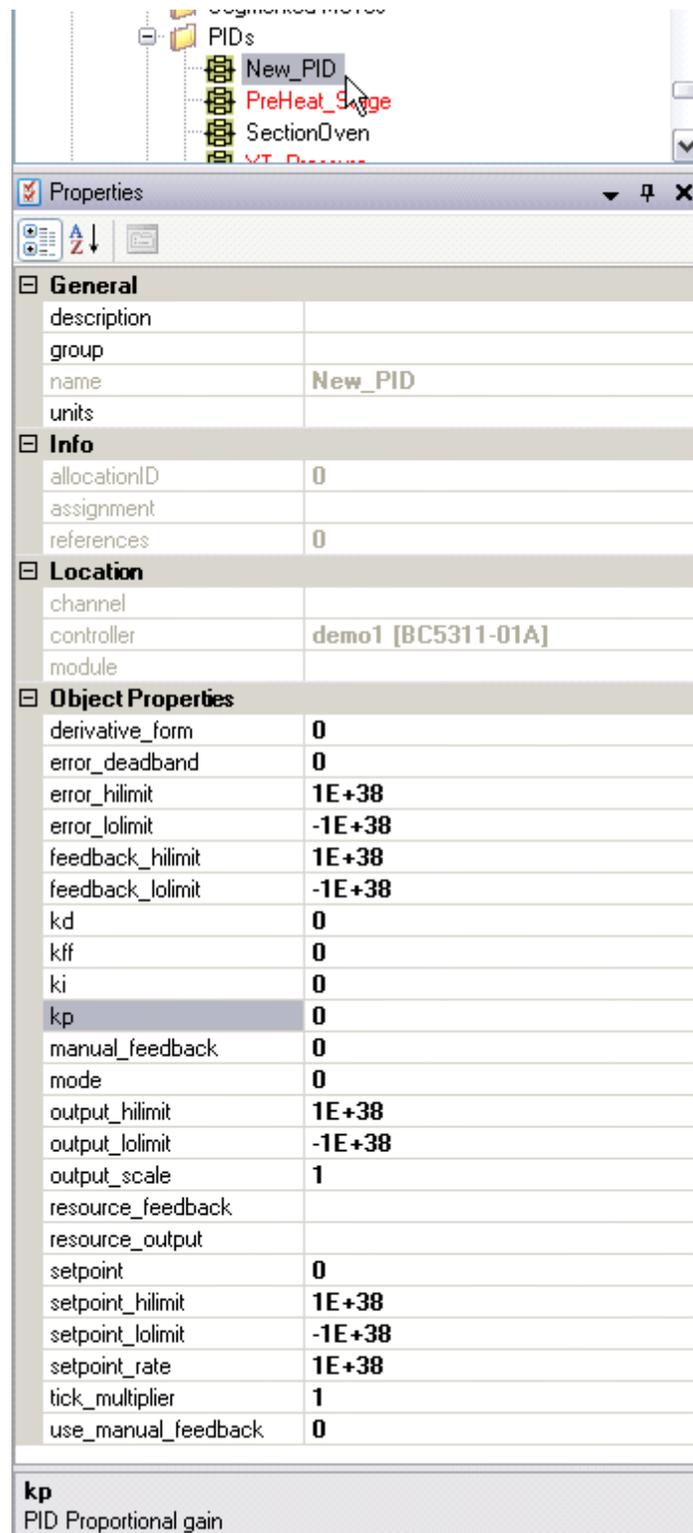
The Properties window for New_PID

2.4   PID Object Properties

The following properties can be set in the Properties window of QuickBuilder. Most can also be altered in QS4 code as well through *dot property notation*:

```
pidname.property
```

Only items marked [REQUIRED] need be filled in. All other parameters are optional and need only be applied where they improve or are required for the process.

- **derivative_form**: When this parameter is set to a non-zero value, the PID algorithm is followed by an additional derivative. This is used when the process being controlled is self-integrating.

- **error_deadband**: This value controls when the loop ignores small values of *error*. The absolute-value of the *error* is compared to the value specified. If the absolute error value is less than or equal to this value, the *error* for this update is set to be zero.

- **enabled**: Controls whether or not the PID loop is active. When set to a zero value, the loop is effectively disabled, since only the *offset* is routed to the output limiter.

- **error_hilimit**: Limits the maximum value of the error fed into the PID equation.

- **error_lolimit**: Limits the minimum value of the error fed into the PID equation.

- **feedback_hilimit**: Limits the maximum value of the feedback signal fed into the error calculation.

- **feedback_lolimit**: Limits the minimum value of the feedback signal fed into the error calculation.

- **integrator_unwind_constant**: A factor that determines how fast the integrator should self-discharge when either the *output* is in limit or the value for ki is set to 0.

  A value of 1.00 (the maximum) means that the integrator should hold its last value and not discharge in those two cases. A value >0 but <1 discharges the integrator by multiplying the integrator by that value each update.

- **kd**: Constant that determines the derivative gain.

- **kff**: Constant that determines the feed-forward gain. Provides improved response when the setpoint value is changed.

- **ki**: Constant that determines the integral gain.

- **kp**: Constant that determines the proportional gain.

- **manual_feedback**:   The value that is used in place of *resource_feedback* when the *use_manual_feedback* parameter is set equal to a non-zero value.

- **mode**: Reserved for future use.

- **offset**: Offsets the generated output value. Can be used in conjunction with *enabled* to force the output of the PID to a specific value (by setting *enabled* to 0 and the value to force the PID output to into the property *offset*).

- **output_hilimit**: Limits the maximum value of the PID loop output (before output scaling).

- **output_lolimit**: Limits the minimum value of the PID loop output (before output scaling).

- **output_scale**: The PID output is multiplied by this value to get the *resource_output* value. Setting this equal to -1 effectively negates the output value when required.

- **resource_feedback**: This is the controller resource used for feedback to the PID loop. **[REQUIRED]**

- **resource_output**: This is the controller resource connected to the scaled output of the PID loop. **[REQUIRED]**

- **setpoint**: The desired initial value for the setpoint.

- **setpoint_hilimit**: Limits the maximum allowable value for the setpoint.

- **setpoint_lolimit**: Limits the minimum allowable value for the setpoint.

- **setpoint_rate**: Limits the rate at which a change in setpoint is presented to the system (/sec).

- **tick_multiplier**: All Model 5300 CPU modules have a settable tick rate (default tick = 50ms) that is used to limit how fast it performs certain operations such as filtering analog I/O points. The PID update rate = (controller tick) * (tick_multiplier).

- **use_manual_feedback**: If set = 0, then *resource_feedback* is used. If set = 1, then *manual_feedback* is used.

2.5   Accessing Properties in QS4 code

The screen shot below shows *dot properties* for a PID object named "pid1" being accessed in the Code window of QuickBuilder. This is a very powerful feature of the PID object that lets the application designer manipulate most aspects of the PID loop under program control. The selection box shown below pops up automatically as soon as the period key is pressed after typing a PID object name. Dot properties enable object properties to be accessed directly in code. For example, pidOne.rate refers to the loop rate of the PID object named pidOne.

Note that the bubble help also tells what type of variable is used for the property: in this case *setpoint* is a read/write floating point value.

The properties listed below can only be accessed in QuickBuilder code via the dot properties. They cannot be set in the Property Window, as they are read-only values, or they are computed on-the-fly by the PID loop.

- **error**: Current (nth value) calculated error value after limiting and deadband.

- **error0**: Previously (n-1) calculated error value after limiting and deadband.

- **error1**: Previously (n-2) calculated error value after limiting and deadband.

- **feedback**: Current value of feedback (manual or resource) after limiting.

- **in_error_limit**: True when the error value is currently being limited.

- **in_feedback_limit**: True when the feedback value is currently being limited.

- **in_output_limit**: True when the output value is currently being limited.

- **in_setpoint_limit**: True when the setpoint value is currently being limited.

- **integrator**: The current value of the integrator (derivate-form=0 only).

- **iperiod**: An internal PID value used to determine the PID period.

- **output**: Value of the PID output after scaling and limiting.

- **pcount**: PID processed counter – holds the number of times the PID loop has run.

- **period**: the actual PID period (sec) – a computed read-only value.

- **rate**: the actual PID rate (Hz) – a computed read-only value.

- **setpoint_rate_adjusted**: The rate-adjusted setpoint value.

- **subtick**: PID sub tick – counts up to tick_multiplier.

Here is an example of dot properties in QS4 code:

3   Appendix A: PID Loop Tuning

The following table gives general guidelines as to the effect of changing the PID tuning gains.

This is only a general guideline, because there are interdependencies between these variables and changing one will impact the other two.

PID Factor	Rise Time	Overshoot	Settling Time	Steady State Error
larger kp	Decreases	Increases	Small Effect	Decreases
larger ki	Decreases	Increases	Increases	Eliminates
larger kd	Small Effect	Decreases	Decreases	Small Effect
larger kff	Decreases	May Increase	Generally Decreases	No Effect

Tuning Response Curves
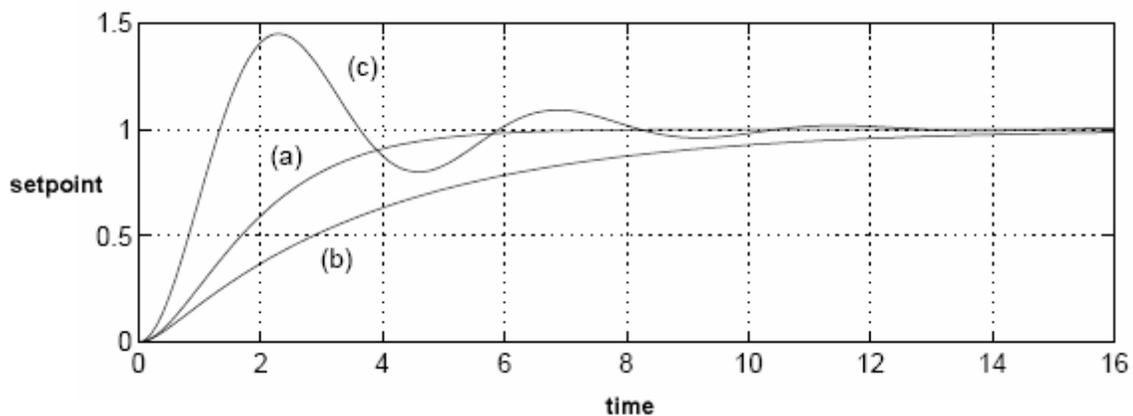
The plot above shows the system response based on three different tuning setups based on a setpoint change from 0 to 1.

a) **Critically Damped:** The optimally tuned system is shown in curve (a). This system is said to be critically damped. It does not overshoot the setpoint value and settles quickly (6 seconds) at the new setpoint value**.**

b) **Over Damped:**   Curve (b) shows a system that is over damped. It does not overshoot the setpoint; however, it takes too long to reach the desired setpoint**.**

c) Under Damped: Curve (c) shows a system that is under damped. It overshoots the setpoint and then oscillates around the setpoint**.**

Index

- P -

- Q -