# How PID Control Works and Implemented on the MSP430

Mark Barnhill
ECE480 Design Team 4

Application Note
November 19, 2010

## EXECUTIVE SUMMARY

A PID controller is the most common instrument used when control of a variable is needed such as, speed, temperature, pressure, and other variables. A PID controller is used to continuously vary a regulator which adjusts the variable being controlled. A PID controller can be implemented in code using simple but precise calculations.

**Keywords:** PID controller, MSP430

# INTRODUCTION

Control of a system is a very important component in many applications today. A proportional-integral-derivative (PID) controller is a control loop feedback mechanism used in many industrial control applications. An error signal is calculated by evaluating the difference between some desired value and the actual output of the system. The controller performs the PID math functions on this calculated error and the sum is applied to some machine. Each part of the PID controller has different effects on the response of the system. The process of calculating the error is repeated continuously each time the output changes and the PID controller math functions are applied to this new error. This application note will instruct the user how a PID controller works and implemented on the MSP430.

# OBJECTIVE

The objective of this application note is to provide the reader with the information needed to understand how PID controllers work and implemented on the MSP430. It is assumed the user has a basic understanding of the MSP430 microcontroller and basic C coding.

# BACKGROUND

The PID controller is made up of three separate terms: the proportional term, the integral term, and the derivative term (Figure 1). Each term in the controller affects a different aspect of the output of the system. The input to the controller is the calculated error. The derivative and integral of this calculated error is taken so it can be used in the controller. The proportional term is calculated by taking the proportion gain, $K_p$, times the magnitude of the calculated error. The proportional term will decrease the steady state error and rise time, but increase the overshoot in the system. The integral term is calculated by taking the integral gain, $K_i$, multiplied by the integral of the calculated error. The integral term decreases the rise time, increases the overshoot and settling time, and eliminates the steady state error. The derivative term is calculated by taking the derivative gain, $K_d$, multiplied by the derivative of the calculated error. The derivative term decreases both the overshoot and settling times.

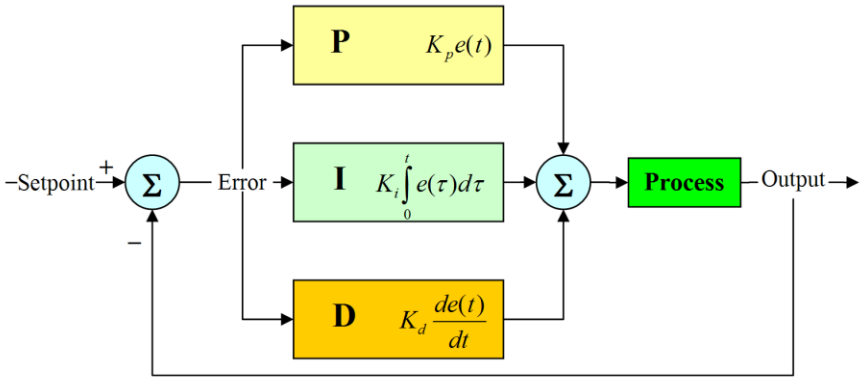


Figure 1: System with a PID Controller

# IMPLEMENTATION

The PID controller is implemented on the MSP430 using a macro. A macro is used when a complex or repetitive task needs to be implemented. The input values of the macro are defined in the main portion of the program while the macro is implemented in a header file referenced by the main program.

```
#define PID_MACRO(v)                                                    \
      v.Err = v.Ref - v.Fdb;  /* Compute the error */                   \
      v.Up = v.Kp * v.Err;    /* Compute Up */                          \
      v.Ui = v.Ui + v.Ki*v.Up + v.Kc*v.SatErr;  /* Compute Ui */        \
      v.OutPreSat = v.Up + v.Ui;    /* Compute pre-saturated output */  \
      if (v.OutPreSat > v.OutMax)   /* Saturate output */               \
      {v.Out = v.OutMax;}                                               \
      else if (v.OutPreSat < v.OutMin)                                  \
      {v.Out = v.OutMin;}                                               \
      else                                                              \
      {v.Out = v.OutPreSat;}                                            \
      v.SatErr = v.Out - v.OutPreSat; /* Compute saturate difference */ \
      v.Up1 = v.Up;
// Add the lines below if derivative output is needed following the integral
update
// v.Ud = v.Kd * (v.Up - v.Up1);
// v.OutPreSat = v.Up + v.Ui + v.Ud;
```

If the previous code, v.Ref is the reference value which is set in the main program and v.Fdb is the feedback sensed from the motor. These two values are used to calculate the error of the system. The output from the proportional term is, v.Up, while the output from the integral term is, v.Ui. The integral output is calculated by taking the sum of three terms: the previous integral output, the product of the integral gain and proportional output, and the product of an integral gain factor and the saturated error. The proportional and integral terms are sufficient enough for the motor control, so the derivative term is not needed. If for some reason the derivative term is needed, the code is commented out at the end to the code above. The pre-saturated value, v.OutPreSat, is the sum of the proportional and integral outputs. If the v.OutPreSat is above the maximum output, v.outMax, the output is set to v.outMax, and if the v.OutPreSet is below the minimum output, v.outMin, the output is set to v.outMin. The saturated difference, v.SatErr, is calculated by taking the difference between the output after saturation and the output before saturation. This saturated difference is used when calculating the integral output.

# CONCLUSION

Some sort of controller is an essential component when controlling the speed of the motor. Since the PID controller is most common controller used by industries, it makes sense to use this controller opposed to other controllers. In motor control we are most interested in minimizing the rise time and eliminating the steady state error. For this reason, the derivate term of the controller is not needed for our application of motor control.

# REFERENCES

"CTM: PID Tutorial." *Control Tutorials for MATLAB*. University of Michigan, 26 Aug. 1997. Web. 16 Nov. 2010. <http://www.engin.umich.edu/group/ctm/PID/PID.html>.

Nise, Norman S. *Control Systems Engineering*. 5th ed. [Hoboken, NJ]: Wiley, 2008. Print.

"The PID Controller." *ECircuit Center*. 2002. Web. 19 Nov. 2010. <http://www.ecircuitcenter.com/circuits/pid1/pid1.htm>.

"PID Controller Simplified." *My Weblog*. 11 May 2008. Web. 16 Nov. 2010. <http://radhesh.wordpress.com/2008/05/11/pid-controller-simplified/>.