AVR221: Discrete PID controller

离散的PID控制器

8-bit AVR® Microcontrollers

Application Note

Features

- **Simple discrete PID controller algorithm** 离散的PID控制器算法
- **Supported by all AVR devices**
- **PID function uses 534 bytes of code memory and 877 CPU cycles (IAR - low size optimization)**

1 Introduction

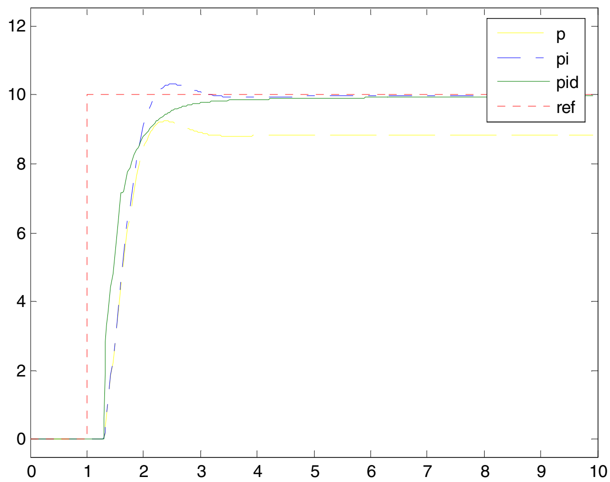
This application note describes a simple implementation of a discrete Proportional-Integral-Derivative (*PID*) controller.

When working with applications where control of the system output due to changes in the reference value or state is needed, implementation of a control algorithm may be necessary. Examples of such applications are motor control, control of temperature, pressure, flow rate, speed, force or other variables. The *PID* controller can be used to control any measurable variable, as long as this variable can be affected by manipulating some other process variables.

Many control solutions have been used over the time, but the *PID* controller has become the 'industry standard' due to its simplicity and good performance.

For further information about the PID controller and its implications the reader should consult other sources, e.g. *PID Controllers* by K. J. Astrom & T. Hagglund (1995).

Figure 1-1. Typical *PID* regulator response to step change in reference input



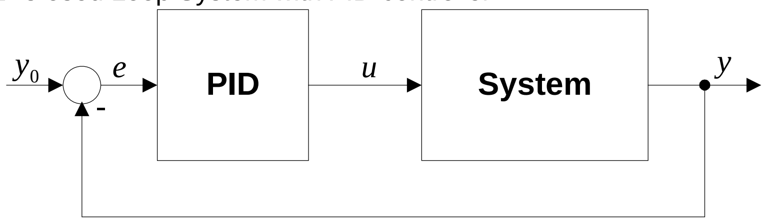
Rev. 2558A-AVR-05/06

2 PID controller

In Figure 2-1 a schematic of a system with a *PID* controller is shown. The *PID* controller compares the measured process value y with a reference setpoint value, y_0. The difference or error, e, is then processed to calculate a new process input, u. This input will try to adjust the measured process value back to the desired setpoint.

The alternative to a closed loop control scheme such as the *PID* controller is an open loop controller. Open loop control (no feedback) is in many cases not satisfactory, and is often impossible due to the system properties. By adding feedback from the system output, performance can be improved.

Figure 2-1. Closed Loop System with *PID* controller



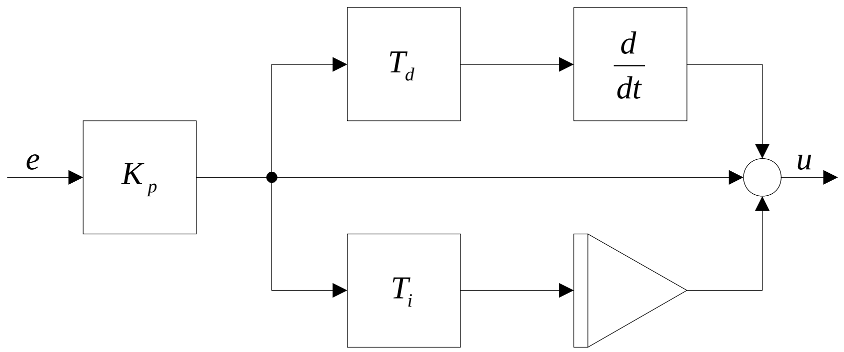
Unlike simple control algorithms, the *PID* controller is capable of manipulating the process inputs based on the history and rate of change of the signal. This gives a more accurate and stable control method.

控制现在—比例
控制过去—积分
控制将来—导数

The basic idea is that the controller reads the system state by a sensor. Then it subtracts the measurement from a desired reference to generate the error value. The error will be managed in three ways, to handle the present, through the proportional term, recover from the past, using the integral term, and to anticipate the future, through the derivate term.

Figure 2-2 shows the *PID* controller schematics, where T_p, T_i, and T_d denote the time constants of the proportional, integral, and derivative terms respectively.

Figure 2-2. *PID* controller schematic



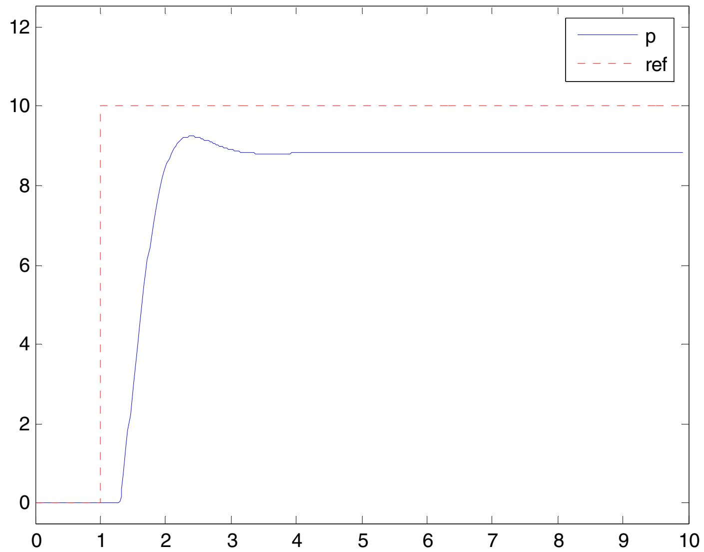
2.1 Proportional term

比例

The proportional term (*P*) gives a system control input proportional with the error. Using only *P* control gives a stationary error in all cases except when the system control input is zero and the system process value equals the desired value. In Figure 2-3 the stationary error in the system process value appears after a change in the desired value (ref). Using a too large *P* term gives an unstable system.

预期值（ref）

Figure 2-3. Step response *P* controller



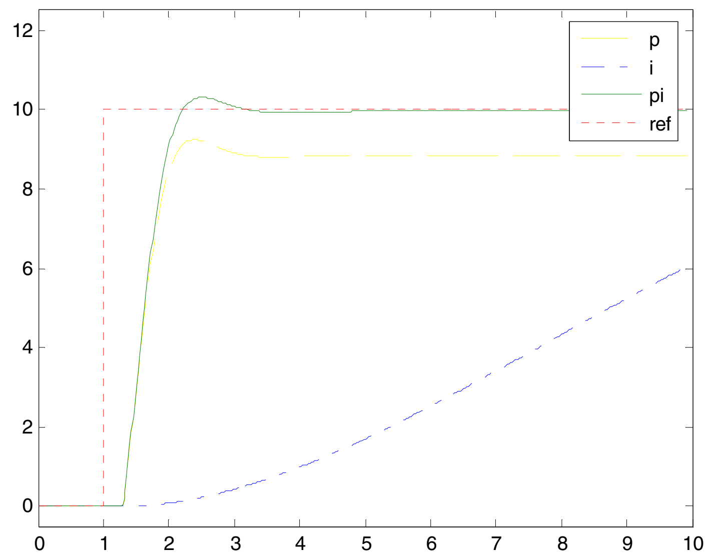
2.2 Integral term

积分

The integral term (*I*) gives an addition from the sum of the previous errors to the system control input. The summing of the error will continue until the system process value equals the desired value, and this results in no stationary error when the reference is stable. The most common use of the *I* term is normally together with the *P* term, called a *PI* controller. Using only the *I* term gives slow response and often an oscillating system. Figure 2-4 shows the step responses to a *I* and *PI* controller. As seen the *PI* controller response have no stationary error and the *I* controller response is very slow.

如果仅仅用I部分（积分部分）的话，反应会很慢并且系统会振荡。

Figure 2-4. Step response *I* and *PI* controller



2.3 Derivative term

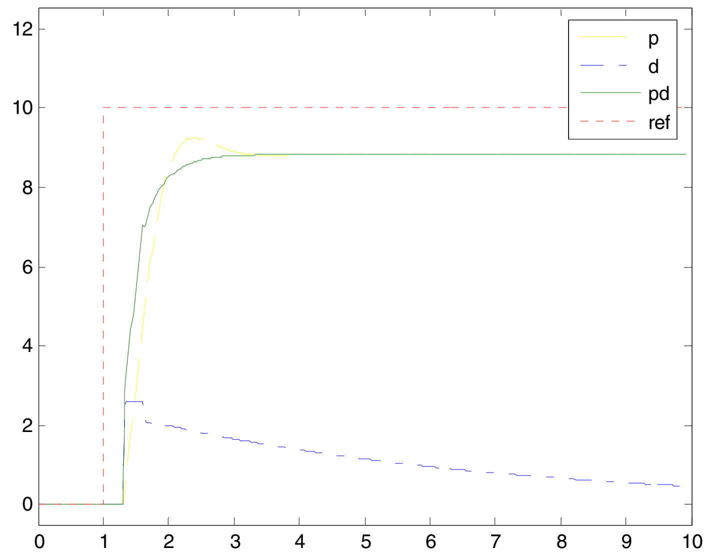
导数

The derivative term (*D*) gives an addition from the rate of change in the error to the system control input. A rapid change in the error will give an addition to the system control input. This improves the response to a sudden change in the system state or reference value. The *D* term is typically used with the *P* or *PI* as a *PD* or *PID* controller. A to large *D* term usually gives an unstable system. Figure 2-5 shows *D* and *PD* controller responses. The response of the *PD* controller gives a faster rising system process value than the *P* controller. Note that the *D* term essentially behaves as a highpass filter on the error signal and thus easily introduces instability in a system and make it more sensitive to noise.

PD控制器比P控制器有一个跟快的上升过程

D部分本质上相当于一个对于误差信号的高通滤波器，因此很容易在系统中引入不稳定因素和对噪声很敏感。

Figure 2-5. Step response *D* and *PD* controller



Using all the terms together, as a *PID* controller usually gives the best performance. Figure 2-6 compares the *P*, *PI*, and *PID* controllers. *PI* improves the *P* by removing the stationary error, and the *PID* improves the *PI* by faster response and no overshoot.

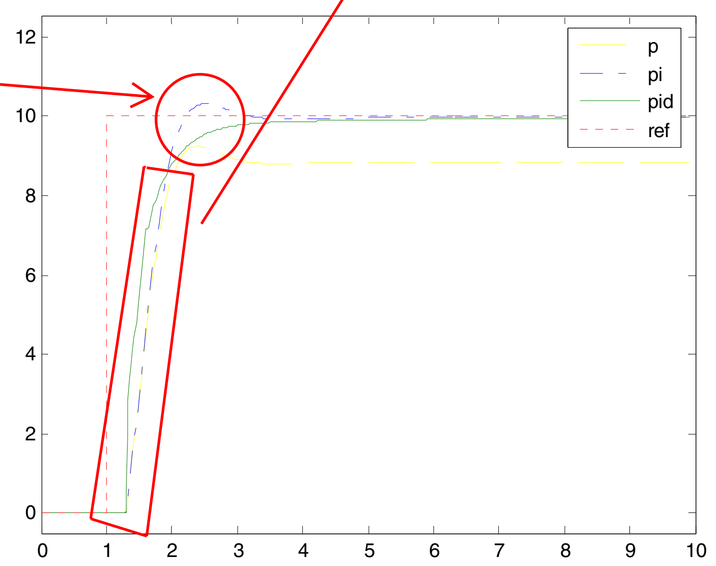
PI比P少了静态误差、PID比PI反应速度更快并且没有了过冲。

Figure 2-6. Step response *P*, *PI* and *PID* controller

什么是过冲(overshoot)和下冲(undershoot)？

过冲就是第一个峰值或谷值超过设定电压——对于上升沿是指最高电压而对于下降沿是
指最低电压。
下冲是指下一个谷值或峰值。

过分的过冲能够引起保护二极管工作，导致过早地失效。过分的下冲能够引起假的时钟或数据错误(误操作)。



2.4 Tuning the parameters

最好的寻找PID参数的办法是从系统的数学模型出发，从想要的反应来计算参数。很多时候一个详细的数学描述是不存在的，这时候就需要实际地调节PID的参数

The best way to find the needed *PID* parameters is from a mathematical model of the system, parameters can then be calculated to get the desired response. Often a detailed mathematical description of the system is unavailable, experimental tuning of the *PID* parameters has to be performed. Finding the terms for the *PID* controller can be a challenging task. Good knowledge about the systems properties and the way the different terms work is essential. The optimum behavior on a process change or setpoint change depends on the application at hand. Some processes must not allow overshoot of the process variable from the setpoint. Other processes must minimize the energy consumption in reaching the setpoint. Generally, stability is the strongest requirement. The process must not oscillate for any combinations or setpoints. Furthermore, the stabilizing effect must appear within certain time limits.

一些系统需要避免过冲
一些系统需要在选点上功耗最小
当然稳定性是最高的要求

Several methods for tuning the *PID* loop exist. The choice of method will depend largely on whether the process can be taken off-line for tuning or not. Ziegler-Nichols method is a well-known online tuning strategy. The first step in this method is setting the *I* and *D* gains to zero, increasing the *P* gain until a sustained and stable oscillation (as close as possible) is obtained on the output. Then the critical gain K_c and the oscillation period P_c is recorded and the *P*, *I* and *D* values adjusted accordingly using Table 2-1.

Ziegler-Nichols方法

Table 2-1. Ziegler-Nichols parameters

Controller	K_p	T_i	T_d
P	0.5 * K_c		
PD	0.65 * K_c		0.12 * P_c
PI	0.45 * K_c	0.85 * P_c	
PID	0.65 * K_c	0.5 * P_c	0.12 * P_c

步骤：
1. 先置Ki=Kd=0,逐渐增加Kp直到在输出得到一个持续的稳定的振荡。
2. 记录下振荡时的P部分的临界增益Kc,和振荡周期Pc，代到下表中计算出Kp,Ki(Ti),Kd(Td)。

Kp＝0.6*Km
Kd＝Kp*π/4*ω
Ki＝Kp*ω/π

上式中Kp为比例控制参数
Kd为微分控制参数
Ki为积分控制参数
Km为系统开始振荡时的比例值
ω为振荡时的频率

Further tuning of the parameters is often necessary to optimize the performance of the *PID* controller.

The reader should note there is systems where the *PID* controller will not work very well, or will only work on a small area around a given system state. Non-linear systems can be such, but generally problems often arise with *PID* control when systems are unstable and the effect of the input depends on the system state.

2.5 Discrete PID controller

A discrete *PID* controller will read the error, calculate and output the control input at a given time interval, at the sample period T. The sample time should be less than the shortest time constant in the system.

2.5.1 Algorithm background

Unlike simple control algorithms, the *PID* controller is capable of manipulating the process inputs based on the history and rate of change of the signal. This gives a more accurate and stable control method.

Figure 2-2 shows the *PID* controller schematics, where T_p, T_i, and T_d denotes the time constants of the proportional, integral, and derivative terms respectively.

The transfer function of the system in Figure 2-2:

$$\frac{u}{e}(s) = H(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right)$$

This gives u with respect to e in the time domain:

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(\sigma)d\sigma + T_d\frac{de(t)}{dt}\right)$$

Approximating the integral and the derivative terms to get the discrete form, using:

$$\int_0^t e(\sigma)d\sigma \approx T\sum_{k=0}^{n} e(k) \qquad \frac{de(t)}{dt} \approx \frac{e(n) - e(n-1)}{T} \qquad t = nT$$

Where n is the discrete step at time t.

This gives the controller:

$$u(n) = K_p e(n) + K_i\sum_{k=0}^{n} e(k) + K_d\left(e(n) - e(n-1)\right)$$

Where:

$$K_i = \frac{K_p T}{T_i} \qquad K_d = \frac{K_p T_d}{T}$$

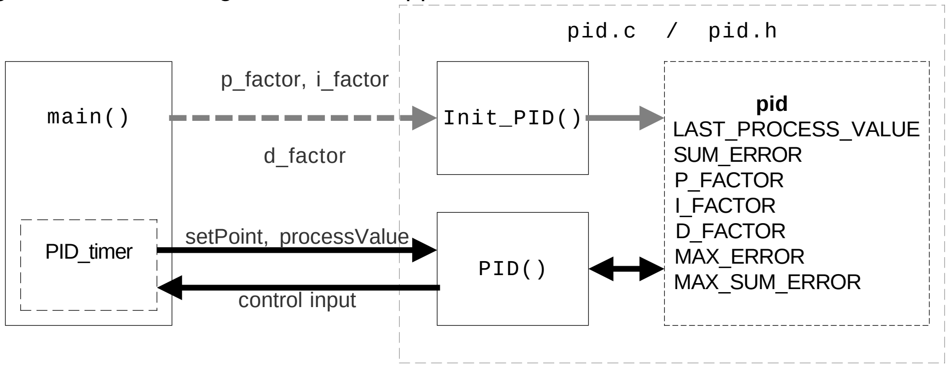
To avoid that changes in the desired process value makes any unwanted rapid changes in the control input, the controller is improved by basing the derivative term on the process value only:

$$u(n) = K_p e(n) + K_i\sum_{k=0}^{n} e(k) + K_d\left(y(n) - y(n-1)\right)$$

3 Implementation

A working implementation in C is included with this application note. Full documentation of the source code and compilation information if found by opening the 'readme.html" file included with the source code.

Figure 3-1. Block diagram of demo application



In Figure 3-1 a simplified block diagram of the demo application is shown.

The *PID* controller uses a struct to store its status and parameters. This struct is initialized in main, and only a pointer to it is passed to the `Init_PID()` and `PID()` functions.

The `PID()` function must be called for each time interval T, this is done by a timer who sets the PID_timer flag when the time interval has passed. When the PID_timer flag is set the main routine reads the desired process value (setPoint) and system process value, calls `PID()` and outputs the result to the control input.

To increase accuracy the p_factor, i_factor and d_factor are scaled with a factor 1:128. The result of the *PID* algorithm is later scaled back by dividing by 128. The value 128 is used to allow for optimizing in the compiler.

$$PFactor = 128K_p$$

Furthermore the effect of the IFactor and DFactor will depend on the sample time T.

$$IFactor = 128K_p \frac{T}{T_i}$$

$$DFactor = 128K_p \frac{T_d}{T}$$

3.1 Integral windup

When the process input, u, reaches a high enough value, it is limited in some way. Either by the numeric range internally in the PID controller, the output range of the controller or constraints in amplifiers or the process itself. This will happen if there is a large enough difference in the measured process value and the reference setpoint value, typically because the process has a larger disturbance / load than the system is capable of handling.

If the controller uses an integral term, this situation can be a problematic. The integral term will sum up as long as the situation last, and when the larger disturbance / load disappear, the PID controller will overcompensate the process input until the integral sum is back to normal.

This problem can be avoided in several ways. In this implementation the maximum integral sum is limited by not allowing it to become larger than `MAX_I_TERM`. The correct size of the `MAX_I_TERM` will depend on the system and sample time used.

4 Further development

The *PID* controller presented here is a simplified example. The controller should work fine, but it might be necessary to make the controller even more robust (limit runaway/overflow) in certain applications. Adding saturation correction on the integral term, basing the proportional term on only the system process value can be necessary.

In the calculating of IFactor and DFactor the sample time T is a part of the equation. If the sample time T used is much smaller or larger than 1 second, accuracy for either IFactor or DFactor will be poor. Consider rewriting the *PID* algorithm and scaling so accuracy for the integral and derivate term is kept.

5 Literature references

K. J. Astrom & T. Hagglund, 1995: *PID Controllers: Theory, Design, and Tuning*. International Society for Measurement and Con.

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature