PID 算法（c 语言)（来自老外）

(2010-02-17 00:18:24)

转载

```
#include <stdio.h>
#include<math.h>
//定义 PID 的结构体
struct _pid
{
int pv; //integer that contains the process value 过程量
int sp; //＊integer that contains the set point   设定值
float integral; // 积分值 —— 偏差累计值
float pgain;
float igain;
float dgain;
int deadband;    //死区
int last_error;
};

struct _pid warm,*pid;
int process_point, set_point,dead_band;
```

```
float p_gain， i_gain， d_gain，
integral_val,new_integ;;

//------------------------
----------------------

pid_init DESCRIPTION This function initializes the
pointers in the _pid structure to the process variable
and the setpoint. *pv and *sp are integer pointers.
//------------------------
----------------------


void pid_init(struct _pid *warm， int process_point,
int set_point)
{
struct _pid *pid;
pid = warm;
pid->pv = process_point;
pid->sp = set_point;
}

//------------------------
----------------------
```

pid_tune DESCRIPTION Sets the proportional gain (p_gain), integral gain (i_gain), derivitive gain (d_gain), and the dead band (dead_band) of a pid control structure _pid.

设定 PID 参数 ———— P,I,D,死区

```
//————————————————————————
——————————————————————


void pid_tune(struct _pid *pid, float p_gain, float
i_gain, float d_gain, int dead_band)
{
pid->pgain = p_gain;
pid->igain = i_gain;
pid->dgain = d_gain;
pid->deadband = dead_band;
pid->integral= integral_val;
pid->last_error=0;
}

//————————————————————————
——————————————————————
```

pid_setinteg DESCRIPTION Set a new value for the integral term of the pid equation.

This is useful for setting the initial output of the pid controller at start up.

设定输出初始值

//————————————————————————

——————————————————————

```
void pid_setinteg(struct _pid *pid,float new_integ)
{
pid->integral = new_integ;
pid->last_error = 0;
}
```

//————————————————————————

——————————————————————

pid_bumpless DESCRIPTION Bumpless transfer algorithim.

When suddenly changing setpoints, or when restarting the PID equation after an extended pause,

the derivative of the equation can cause a bump in the controller output. This function will help smooth out that bump.

The process value in *pv should be the updated just before this function is used.

pid_bumpless 实现无扰切换

当突然改变设定值时，或重新启动后，将引起扰动输出。这个函数将能实现平顺扰动，在调用该函数之前需要先更新 PV 值

//——

```
void pid_bumpless(struct _pid *pid)
{
pid->last_error = (pid->sp)-(pid->pv);  //设定值与反馈值偏差
}
```

//——

pid_calc DESCRIPTION Performs PID calculations for the _pid structure *a.

This function uses the positional form of the pid equation, and incorporates an integral windup prevention algorithim.

Rectangular integration is used, so this function must be repeated on a consistent time basis for accurate control.

RETURN VALUE The new output value for the pid loop.

USAGE #include "control.h"

本函数使用位置式PID计算方式，并且采取了积分饱和限制运算

PID计算

```
//－－－－－－－－－－－－－－－－－－－－－－－－
－－－－－－－－－－－－－－－－－－－－－－


float pid_calc(struct _pid *pid)
{•
int err;
float pterm, dterm, result, ferror;

// 计算偏差
err = (pid->sp) - (pid->pv);
```

```
// 判断是否大于死区
if (abs(err) > pid->deadband)
{
ferror = (float) err;   //do integer to float conversion only once 数据类型转换

// 比例项
pterm = pid->pgain * ferror;

if (pterm > 100 || pterm < -100)
{
pid->integral = 0.0;
}
else
{
// 积分项
pid->integral += pid->igain * ferror;

// 输出为0——100%
// 如果计算结果大于100，则等于100
if (pid->integral > 100.0)
{
```

```
pid->integral = 100.0;
}
// 如果计算结果小于0.0，则等于0
else if (pid->integral < 0.0)
pid->integral = 0.0;

}

// 微分项
dterm = ((float)(err - pid->last_error)) * pid->dgain;

result = pterm + pid->integral + dterm;
}
else
result = pid->integral; // 在死区范围内，保持现有输
出

// 保存上次偏差
pid->last_error = err;

// 输出 PID 值(0-100)
return (result);
```

```
}

//------------------------
----------------------
void main(void)
{
float display_value;
int count=0;
pid = &warm;

// printf(″Enter the values of Process point, Set
point, P gain, I gain, D gain \n″);
// scanf(″%d%d%f%f%f″, &process_point, &set_point,
&p_gain, &i_gain, &d_gain);

// 初始化参数
process_point = 30;
set_point = 40;
p_gain = (float)(5.2);
i_gain = (float)(0.77);
d_gain = (float)(0.18);
dead_band = 2;
```

```
integral_val =(float)(0.01);

printf("The values of Process point， Set point， P gain，
I gain， D gain \n");
printf(" %6d %6d %4f %4f %4f\n"， process_point，
set_point， p_gain， i_gain， d_gain);
printf("Enter the values of Process point\n");
while(count<=20)
{
scanf("%d",&process_point);

// 设定PV,SP值
pid_init(&warm， process_point， set_point);

// 初始化PID参数值
pid_tune(&warm， p_gain,i_gain,d_gain,dead_band);

// 初始化PID输出值
pid_setinteg(&warm,0.0);
//pid_setinteg(&warm,30.0);
//Get input value for process point
pid_bumpless(&warm);

// how to display output
display_value = pid_calc(&warm);

printf(″%f\n″, display_value);
//printf(″\n%f%f%f%f″,warm.pv,warm.sp,warm.igain,warm.dgain);

count++;
}
}
```