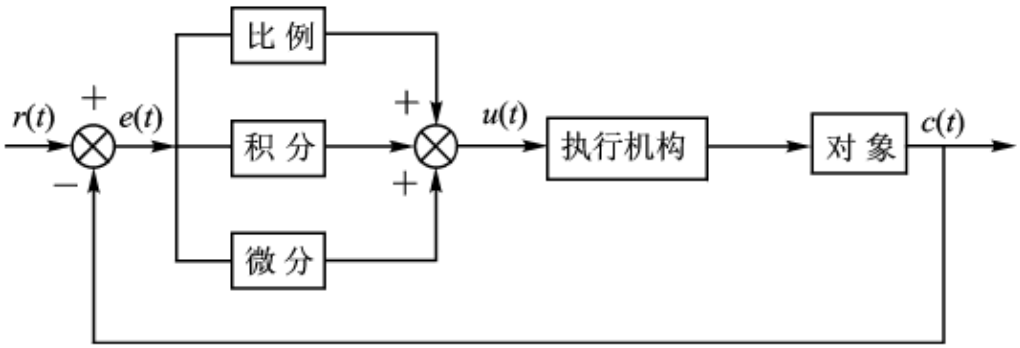
PID 的 C 语言编程

自从计算机进入控制领域以来，用数字计算机代替模拟计算机调节器组成计算机控制系统，不仅可以用软件实现 PID 控制算法，而且可以利用计算机的逻辑功能，使 PID 控制更加灵活。数字 PID 控制在生产过程中是一种最普遍采用的控制方法，在机电、冶金、机械、化工等行业中获得了广泛的应用。将偏差的比例〔P〕、积分(I)和微分(D)通过线性组合构成控制量，对被控对象进行控制，故称 PID 控制器。

PID 控制分为模拟 PID 控制器和数字 PID 控制器

模拟 PID 控制:



模拟 PID 控制系统框图

$$u(t) = K_P\left[e(t) + \frac{1}{T_I}\int_0^t e(t)dt + T_D\frac{de(t)}{dt}\right]$$

模拟 PID 控制的微分方程

因为微机只能处理数字信号不能处理模拟信号，所以要把模拟 PID 控制转换成数字 PID，这就需要把模拟 PID 的参数离散化

模拟形式	离散化形式
$e(t)=r(t)-c(t)$	$e(n)=r(n)-c(n)$
$\frac{de(t)}{dT}$	$\frac{e(n)-e(n-1)}{T}$
$\int_0^t e(t)dt$	$\sum_{i=0}^{n} e(i)T = T\sum_{i=0}^{n} e(i)$

模拟 PID 控制规律的离散化

数字 PID 控制:

数字 PID 算法有两种常用的基本类型：位置型、增量型。

增量型 PID 控制

$$\Delta u(n)=u(n)-u(n-1)$$

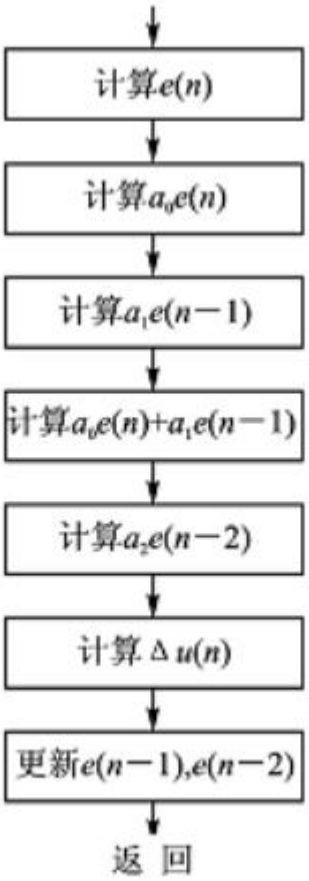
$$=K_P[e(n)-e(n-1)]+K_P\frac{T}{T_I}e(n)+K_P\frac{T_D}{T}[e(n)-2e(n-1)+e(n-2)]$$

增量式 PID 控制原理

$$\Delta u(n)=a_0e(n)+a_1e(n-1)+a_2e(n-2)$$

式中 $a_0=K_P(1+\frac{T}{T_I}+\frac{T_D}{T})$，$a_1=-K_P(1+\frac{2T_D}{T})$，$a_2=-K_P\frac{T_D}{T}$

简化后的增量式 PID 控制原理



增量型 PID 算法的程序流程

PID 程序清单

```
//定义 PID 参数
#define VV_KPVALUE 3            //比例
#define VV_KIVALUE 40          //积分
#define VV_KDVALUE 3          //微分
#define VV_MAX 10000            //返回的最大值,是 pwm 的周期值
#define VV_MIN 0
//#define VV_DEADLINE 0X08     //速度 PID，设置死区范围



//*****************************************************************
***
//PID 算法
//*****************************************************************
***
typedef struct PID            //定义数法核心数据
{
signed int vi_Ref;          //速度 PID，速度设定值       Velocity
signed int vi_FeedBack;    //速度 PID，速度反馈值      m*50

signed long vi_PreError;   //速度 PID，前一次，速度误差,,vi_Ref - vi_FeedBack
signed long vi_PreDerror; //速度 PID，前一次，速度误差之差，d_error-PreDerror;

unsigned int v_Kp;          //速度 PID，Ka = Kp
unsigned int v_Ki;          //速度 PID，Kb = Kp * ( T / Ti )
unsigned int v_Kd;          //速度 PID，

signed long vl_PreU;       //电机控制输出值
}PID;
static PID    sPID;                       //    PID Control Structure
static PID*sptr=&sPID;

void PIDInit(void)
{
sptr->vi_Ref =Velocity ;        //速度设定值
sptr->vi_FeedBack = X ;    //速度反馈值

sptr->vi_PreError = 0 ;     //前一次，速度误差,,vi_Ref - vi_FeedBack
sptr->vi_PreDerror = 0 ;     //前一次，速度误差之差，d_error-PreDerror;
```

```
sptr->v_Kp = VV_KPVALUE;
sptr->v_Ki = VV_KIVALUE;
sptr->v_Kd = VV_KDVALUE;

sptr->vl_PreU = 0 ;          //电机控制输出值
}
 //pid
void speed_pid(int v)
{
signed long    error,d_error,dd_error; //
   error = (signed long)(sptr->vi_Ref - sptr->vi_FeedBack); //  偏差计算
   d_error = error - sptr->vi_PreError;
   dd_error = d_error - sptr->vi_PreDerror;

   sptr->vi_PreError = error; //存储当前偏差
   sptr->vi_PreDerror = d_error; //储存当前误差之差

              sptr->vl_PreU += (signed long)(    sptr -> v_Kp * d_error +
                                                 sptr -> v_Ki * error    +
                                                 sptr->v_Kd*dd_error);
                                               //速度 PID 计算

         if( sptr->vl_PreU >= VV_MAX )//速度 PID，防止调节最高溢出
            sptr->vl_PreU = VV_MAX;
   else
         if( sptr->vl_PreU <= VV_MIN ) //速度 PID，防止调节最低溢出
            sptr->vl_PreU = VV_MIN;
else
         return ( sptr->vl_PreU    );   //  返回预调节占空比
}
```