PID 调节控制做电机速度控制

目录

页

1 模拟 PID 控制

将偏差的比例（Proportion）、积分（Integral）和微分（Differential）通过线性组合构成控制量，用这一控制量对被控对象进行控制，这样的控制器称 PID 控制器。

1.1 模拟 PID 控制原理

在模拟控制系统中，控制器最常用的控制规律是 PID 控制。为了说明控制器的工作原理，先看一个例子。如图 1－1 所示是一个小功率直流电机的调速原理图。给定速度 $n_0(t)$ 与实际转速进行比较 $n(t)$，其差值 $e(t)=n_0(t)-n(t)$，经过 PID 控制器调整后输出电压控制信号 $u(t)$，$u(t)$ 经过功率放大后，驱动直流电动机改变其转速。

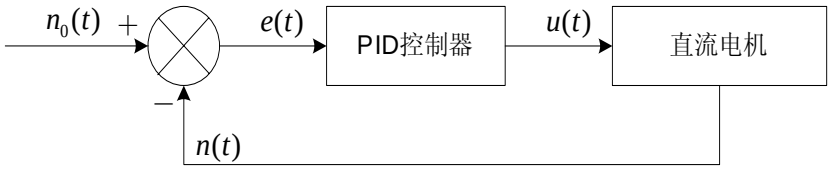

图 1－1 小功率直流电机调速系统

常规的模拟 PID 控制系统原理框图如图 1－2 所示。该系统由模拟 PID 控制器和被控对象组成。图中，$\mathrm{r}(t)$ 是给定值，$\mathrm{y}(t)$ 是系统的实际输出值，给定值与实际输出值构成控制偏差 $e(t)$

$$e(t)=\mathrm{r}(t)-\mathrm{y}(t) \qquad \text{（式 1－1）}$$

$e(t)$ 作为 PID 控制的输入，$u(t)$ 作为 PID 控制器的输出和被控对象的输入。所以模拟 PID 控制器的控制规律为

$$\mathrm{u}(t)=Kp\left[e(t)+\frac{1}{Ti}\int_0^t e(t)dt+Td\frac{de(t)}{dt}\right] \qquad \text{（式 1－2）}$$

其中： Kp －－ 控制器的比例系数

Ti －－ 控制器的积分时间，也称积分系数

Td －－ 控制器的微分时间，也称微分系数

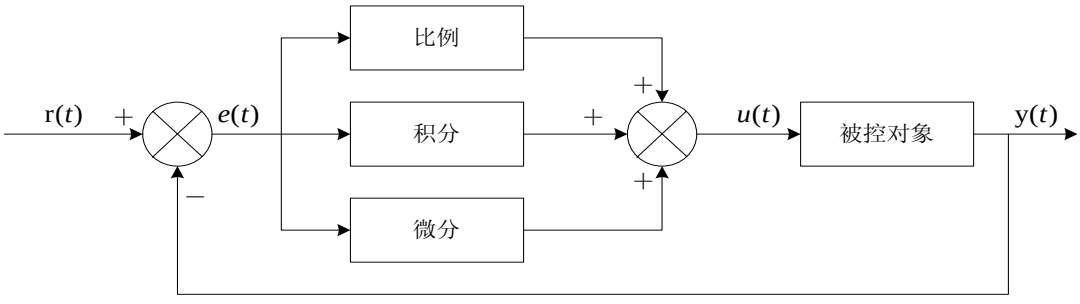

图 1－2 模拟 PID 控制系统原理图

1、 比例部分

比例部分的数学式表示是： $Kp*e(t)$

在模拟 PID 控制器中，比例环节的作用是对偏差瞬间作出反应。偏差一旦产生控制器立即产生控制作用，使控制量向减少偏差的方向变化。控制作用的强弱取决于比例系数 Kp，比例系数 Kp 越大，控制作用越强，则过渡过程越快，控制过程的静态偏差也就越小；但是 Kp 越大，也越容易产生振荡，破坏系统的稳定性。故而，比例系数 Kp 选择必须恰当，才能过渡时间少，静差小而又稳定的效果。

2、 积分部分

积分部分的数学式表示是： $\frac{Kp}{Ti}\int_0^t e(t)dt$

从积分部分的数学表达式可以知道，只要存在偏差，则它的控制作用就不断的增加；只有在偏差 $e(t)=0$ 时，它的积分才能是一个常数，控制作用才是一个不会增加的常数。可见，积分部分可以消除系统的偏差。

积分环节的调节作用虽然会消除静态误差，但也会降低系统的响应速度，增加系统的超调量。积分常数 Ti 越大，积分的积累作用越弱，这时系统在过渡时不会产生振荡；但是增大积分常数 Ti 会减慢静态误差的消除过程，消除偏差所需的时间也较长，但可以减少超调量，提高系统的稳定性。当 Ti 较小时，则积分的作用较强，这时系统过渡时间中有可能产生振荡，不过消除偏差所需的时间较短。所以必须根据实际控制的具体要求来确定 Ti 。

3、 微分部分

微分部分的数学式表示是： $Kp*Td\frac{de(t)}{dt}$

实际的控制系统除了希望消除静态误差外，还要求加快调节过程。在偏差出现的瞬间，或在偏差变化的瞬间，不但要对偏差量做出立即响应（比例环节的作用），而且要根据偏差的变化趋势预先给出适当的纠正。为了实现这一作用，可在 PI 控制器的基础上加入微分环节，形成 PID 控制器。

微分环节的作用使阻止偏差的变化。它是根据偏差的变化趋势（变化速度）进行控制。偏差变化的越快，微分控制器的输出就越大，并能在偏差值变大之前进行修正。微分作用的引入，将有助于减小超调量，克服振荡，使系统趋于稳定，特别对高阶系统非常有利，它加快了系统的跟踪速度。但微分的作用对输入信号的噪声很敏感，对那些噪声较大的系统一般不用微分，或在微分起作用之前先对输入信号进行滤波。

微分部分的作用由微分时间常数 Td 决定。Td 越大时，则它抑制偏差 $e(t)$ 变化的作用越强；Td 越小时，则它反抗偏差 $e(t)$ 变化的作用越弱。微分部分显然对系统稳定有很大的作用。

适当地选择微分常数 Td ，可以使微分作用达到最优。

由于计算机的出现，计算机进入了控制领域。人们将模拟 PID 控制规律引入到计算机中来。对（式 1－2）的 PID 控制规律进行适当的变换，就可以用软件实现 PID 控制，即数字 PID 控制。

2 数字 PID 控制

数字式 PID 控制算法可以分为位置式 PID 和增量式 PID 控制算法。

2.1 位置式 PID 算法

由于计算机控制是一种采样控制，它只能根据采样时刻的偏差计算控制量，而不能像模拟控制那样连续输出控制量量，进行连续控制。由于这一特点（式 1－2）中的积分项和微分项不能直接使用，必须进行离散化处理。离散化处理的方法为：以 T 作为采样周期，k 作为采样序号，则离散采样时间 kT 对应着连续时间 t，用矩形法数值积分近似代替积分，用一阶后向差分近似代替微分，可作如下近似变换：

$$\left.\begin{aligned} &t \approx kT \qquad (k = 0,1,2\ldots\ldots) \\ &\int_0^t e(t)dt \approx T\sum_{j=0}^{k} e(jT) = T\sum_{j=0}^{k} e_j \\ &\frac{de(t)}{dt} \approx \frac{e(kT) - e[(k-1)T]}{T} = \frac{e_k - e_{k-1}}{T} \end{aligned}\right\} \qquad \text{（式 2－1）}$$

上式中，为了表示的方便，将类似于 $e(kT)$ 简化成 e_k 等。

将（式 2－1）代入（式 1－2），就可以得到离散的 PID 表达式为

$$u_k = Kp[e_k + \frac{T}{Ti}\sum_{j=0}^{k} e_j + Td\,\frac{e_k - e_{k-1}}{T}] \qquad \text{（式 2－2）}$$

或

$$u_k = Kp * e_k + Ki\sum_{j=0}^{k} e_j + Kd\,（e_k - e_{k-1}） \qquad \text{（式 2－3）}$$

其中　　k －－　采样序号，k ＝0，1，2，……；

u_k －－ 第 k 次采样时刻的计算机输出值；

e_k －－ 第 k 次采样时刻输入的偏差值；

e_{k-1} －－ 第 k －1 次采样时刻输入的偏差值；

Ki －－积分系数，$Ki = Kp * T / Ti$；

Kd －－微分系数，$Kd = Kp * Td / T$；

如果采样周期足够小，则（式 2－2）或（式 2－3）的近似计算可以获得足够精确的结果，离散控制过程与连续过程十分接近。

（式 2－2）或（式 2－3）表示的控制算法式直接按（式 1－2）所给出的 PID 控制规律定义进行计算的，所以它给出了全部控制量的大小，因此被称为全量式或位置式 PID 控制算法。

这种算法的缺点是：由于全量输出，所以每次输出均与过去状态有关，计算时要对 e_k 进行累加，

工作量大；并且，因为计算机输出的u_k对应的是执行机构的实际位置，如果计算机出现故障，输出的u_k将大幅度变化，会引起执行机构的大幅度变化，有可能因此造成严重的生产事故，这在实生产际中是不允许的。

增量式 PID 控制算法可以避免着重现象发生。

2.2 增量式 PID 算法

所谓增量式 PID 是指数字控制器的输出只是控制量的增量Δu_k。当执行机构需要的控制量是增量，而不是位置量的绝对数值时，可以使用增量式 PID 控制算法进行控制。

增量式 PID 控制算法可以通过（式 2－2）推导出。由（式 2－2）可以得到控制器的第 k－1 个采样时刻的输出值为：

$$u_{k-1} = Kp[e_{k-1} + \frac{T}{Ti}\sum_{j=0}^{k-1} e_j + Td\frac{e_{k-1} - e_{k-2}}{T}] \qquad \text{（式 2－4）}$$

将（式 2－2）与（式 2－4）相减并整理，就可以得到增量式 PID 控制算法公式为：

$$\begin{aligned}\Delta u_k &= u_k - u_{k-1} = Kp(e_k - e_{k-1} + \frac{T}{Ti}e_k + Td\frac{e_k - 2e_{k-1} + e_{k-2}}{T}) \\ &= Kp(1 + \frac{T}{Ti} + \frac{Td}{T})e_k - Kp(1 + \frac{2Td}{T})e_{k-1} + Kp\frac{Td}{T}e_{k-2}) \\ &= Ae_k + Be_{k-1} + Ce_{k-2}\end{aligned} \qquad \text{（式 2－5）}$$

其中 A ＝ $Kp(1+\frac{T}{Ti}+\frac{Td}{T})$；

B ＝ $Kp(1+\frac{2Td}{T})$；

C ＝ $Kp\frac{Td}{T}$。

由（式 2－5）可以看出，如果计算机控制系统采用恒定的采样周期T，一旦确定 A、B、C，只要使用前后三次测量的偏差值，就可以由（式 2－5）求出控制量。

增量式 PID 控制算法与位置式 PID 算法（式 2－2）相比，计算量小的多，因此在实际中得到广泛的应用。

而位置式 PID 控制算法也可以通过增量式控制算法推出递推计算公式：

$$u_k = u_{k-1} + \Delta u_k \qquad \text{（式 2－6）}$$

（式 2－6）就是目前在计算机控制中广泛应用的数字递推 PID 控制算法。

2.3 控制器参数整定

控制器参数整定：指决定调节器的比例系数Kp、积分时间Ti、微分时间Td和采样周期Ts的

具体数值。整定的实质是通过改变调节器的参数，使其特性和过程特性相匹配，以改善系统的动态和静态指标，取得最佳的控制效果。

整定调节器参数的方法很多，归纳起来可分为两大类，即理论计算整定法和工程整定法。理论计算整定法有对数频率特性法和根轨迹法等；工程整定法有凑试法、临界比例法、经验法、衰减曲线法和响应曲线法等。工程整定法特点不需要事先知道过程的数学模型，直接在过程控制系统中进行现场整定方法简单、计算简便、易于掌握。

2.3.1 凑试法

按照先比例（P）、再积分（I）、最后微分（D）的顺序。

置调节器积分时间 Ti =∞，微分时间 Td =0，在比例系数 Kp 按经验设置的初值条件下，将系统投入运行，由小到大整定比例系数 Kp 。求得满意的 1/4 衰减度过渡过程曲线。

引入积分作用（此时应将上述比例系数 Kp 设置为 5/6 Kp ）。将 Ti 由大到小进行整定。

若需引入微分作用时，则将 Td 按经验值或按 Td =（1/3～1/4） Ti 设置，并由小到大加入。

2.3.2 临界比例法

在闭环控制系统里，将调节器置于纯比例作用下，从小到大逐渐改变调节器的比例系数，得到等幅振荡的过渡过程。此时的比例系数称为临界比例系数 Ku ，相邻两个波峰间的时间间隔，称为临界振荡周期 Tu 。

临界比例度法步骤：

1、将调节器的积分时间 Ti 置于最大（ Ti =∞），微分时间置零（ Td =0），比例系数 Kp 适当，平衡操作一段时间，把系统投入自动运行。

2、将比例系数 Kp 逐渐增大，得到等幅振荡过程，记下临界比例系数 Ku 和临界振荡周期 Tu 值。

3、根据 Ku 和 Tu 值，采用经验公式，计算出调节器各个参数，即 Kp 、 Ti 和 Td 的值。

按“先 P 再 I 最后 D”的操作程序将调节器整定参数调到计算值上。若还不够满意，可再作进一步调整。

临界比例度法整定注意事项：

有的过程控制系统，临界比例系数很大，使系统接近两式控制，调节阀不是全关就是全开，对工业生产不利。

有的过程控制系统，当调节器比例系数 Kp 调到最大刻度值时，系统仍不产生等幅振荡，对此，就把最大刻度的比例度作为临界比例度 Ku 进行调节器参数整定。

2.3.3 经验法

用凑试法确定 PID 参数需要经过多次反复的实验，为了减少凑试次数，提高工作效率，可以借鉴他人的经验，并根据一定的要求，事先作少量的实验，以得到若干基准参数，然后按照经验公式，用这些基准参数导出 PID 控制参数，这就是经验法。

临界比例法就是一种经验法。这种方法首先将控制器选为纯比例控制器，并形成闭环，改变比例系数，使系统对阶跃输入的响应达到临界状态，这时记下比例系数 Ku 、临界振荡周期为 Tu ，根

据 Z－N 提供的经验公式，就可以由这两个基准参数得到不同类型控制器的参数，如表 2－1 所示。

表 2－1 临界比例法确定的模拟控制器参数

控制器类型	Kp	Ti	Td
P	$0.5Ku$		
PI	$0.45Ku$	$0.85Tu$	
PID	$0.6Ku$	$0.5Tu$	$0.12Tu$

这种临界比例法使针对模拟 PID 控制器，对于数字 PID 控制器，只要采样周期取的较小，原则上也同样使用。在电动机的控制中，可以先采用临界比例法，然后在采用临界比例法求得结果的基础上，用凑试法进一步完善。

表 2－1 的控制参数，实际上是按衰减度为 1/4 时得到的。通常认为 1/4 的衰减度能兼顾到稳定性和快速性。如果要求更大的衰减，则必须用凑试法对参数作进一步的调整。

2.3.4 采样周期的选择

香农（Shannon）采样定律 ：为不失真地复现信号的变化，采样频率至少应大于或等于连续信号最高频率分量的二倍。根据采样定律可以确定采样周期的上限值。实际采样周期的选择还要受到多方面因素的影响，不同的系统采样周期应根据具体情况来选择。

采样周期的选择，通常按照过程特性与干扰大小适当来选取采样周期：即对于响应快、（如流量、压力）波动大、易受干扰的过程，应选取较短的采样周期；反之，当过程响应慢（如温度、成份）、滞后大时，可选取较长的采样周期。

采样周期的选取应与 PID 参数的整定进行综合考虑，采样周期应远小于过程的扰动信号的周期，在执行器的响应速度比较慢时，过小的采样周期将失去意义，因此可适当选大一点；在计算机运算速度允许的条件下，采样周期短，则控制品质好；当过程的纯滞后时间较长时，一般选取采样周期为纯滞后时间的 1/4～1/8。

2.4 参数调整规则的探索

人们通过对 PID 控制理论的认识和长期人工操作经验的总结，可知 PID 参数应依据以下几点来适应系统的动态过程。

1、在偏差比较大时，为使尽快消除偏差，提高响应速度，同时为了避免系统响应出现超调，Kp 取大值，Ki 取零；在偏差比较小时，为继续减小偏差，并防止超调过大、产生振荡、稳定性变坏，Kp 值要减小，Ki 取小值；在偏差很小时，为消除静差，克服超调，使系统尽快稳定，Kp 值继续减小，Ki 值不变或稍取大。

2、当偏差与偏差变化率同号时，被控量是朝偏离既定值方向变化。因此，当被控量接近定值时，反号的比列作用阻碍积分作用，避免积分超调及随之而来的振荡，有利于控制；而当被控量远未接近各定值并向定值变化时，则由于这两项反向，将会减慢控制过程。在偏差比较大时，偏差变化率与偏差异号时，Kp 值取零或负值，以加快控制的动态过程。

3、偏差变化率的大小表明偏差变化的速率，$e_k－e_{k-1}$ 越大，Kp 取值越小，Ki 取值越大，反之亦然。同时，要结合偏差大小来考虑。

4、微分作用可改善系统的动态特性，阻止偏差的变化，有助于减小超调量，消除振荡，缩短调节时间t_s，允许加大 Kp，使系统稳态误差减小，提高控制精度，达到满意的控制效果。所以，在e_k比较大时，Kd取零，实际为 PI 控制；在e_k比较小时，Kd取一正值，实行 PID 控制。

2.5 自校正 PID 控制器

对于一个特定的被控对象，在纯比例控制的作用下改变比例系数可以求出产生临界振荡的振荡周期Tu和临界比例系数Ku。

根据 Z－N 条件，有

$T = 0.1Tu$

$Ti = 0.5Tu$

$Td = 0.125Tu$

代入（式 2－5）则有：

$$\Delta u_k = Kp(2.45e_k - 3.5e_{k-1} + 1.25e_{k-2}) \qquad \text{（式 2－7）}$$

很显然，采用上式可以十分容易的实现常数Kp的校正。

3 软件说明

3.1 软件说明

AN_SPMC75_0012 在微处理器 SPMC75F2413A 上实现数字 PID 对 BLDC 速度的调节，重点将是对 PID 参数的整定，使系统的动静态性能达到“满意”的效果。

3.2 档案构成

文件名称	功能	类型
Main	BLDC 驱动相关参数初始化，DMC 服务	C
Chap2	BLDC 驱动相关函数	C
Initial	系统所有相关初始化程序	C
ISR	驱动中相应的中断服务	C
DigitalPID_V100.lib Spmc75_dmc_lib_V100.lib	PID 设置，初始化和 PID 计算函数库函数 DMC 通信程序	lib lib

3.3 DMC 界面

Speed1_Cmd：设置电机运转的速度

Speed1_Now：电机当前反馈速度

Speed1_Kp：2.5 节 自校正 PID 控制器中提到的 Kp 值

User_R0：当前 P_TMR3_TGRA 寄存器的值

User_R1：设置速度与电机实际转速的差值

Motor 1 Start 和 Motor 1 Stop 控制启停

3.4 子程序说明

PIDInit ( )

原　　形	void PIDInit (void)
描　　述	PID 所用到的 RAM 清零
输入参数	无
输出参数	无
头 文 件	Spmc75_PID.h
库 文 件	DigitalPID_V100
注意事项	请在设置参数前使用。
例　　子	PIDInit ();

PIDSetPoint ( )

原　　形	void PIDSetPoint(int)
描　　述	设置 PID 调节的目标值
输入参数	期望值
输出参数	无
头 文 件	Spmc75_PID.h
库 文 件	DigitalPID_V100
注意事项	
例　　子	PIDSetPoint (2000); //期望电动机的转速为 2000rpm

PIDGetSetpoint ( )

原　　形	int PIDGetSetpoint(void)
描　　述	读取 PID 调节设置的目标值
输入参数	无
输出参数	所设置的期望值
头 文 件	Spmc75_PID.h
库 文 件	DigitalPID_V100
注意事项	得到的期望值将和数值的是同一个数值
例　　子	uiSpeed ＝ PIDSetPoint (); //读取所设置的期望电动机转速

PIDSetKp ( )

原　　形	void PIDSetKp(double)
描　　述	设置 PID 的 Kp 值
输入参数	Kp 数值
输出参数	无
头 文 件	Spmc75_PID.h
库 文 件	DigitalPID_V100
注意事项	这个参数在增量 PID 和位置 PID 的计算中代表着不同的意思。 （式 2－4）所示位置式 PID 中 Kp 就是比例系数 （式 2－5）所示增量式 PID 中 Kp 相当于 e_k 的系数 $Kp(1+\frac{T}{Ti}+\frac{Td}{T})$
例　　子	PIDSetKp (0.257); //设置 Kp＝0.257

PIDGetKp ( )

原　　形	double PIDGetKp(void)
描　　述	读取 PID 中所设置的 Kp 值
输入参数	无
输出参数	Kp 数值
头 文 件	Spmc75_PID.h
库 文 件	DigitalPID_V100
注意事项	这个参数在增量 PID 和位置 PID 的计算中代表着不同的意思。 （式 2－4）所示位置式 PID 中 Kp 就是比例系数 （式 2－5）所示增量式 PID 中 Kp 相当于 e_k 的系数 $Kp(1+\frac{T}{Ti}+\frac{Td}{T})$
例　　子	dKp ＝ PIDSetKp ();

PIDSetKi( )

原　　形	void PIDSetKi(double dKii)
描　　述	设置 PID 的 Ki 值
输入参数	Ki 数值
输出参数	无
头 文 件	Spmc75_PID.h
库 文 件	DigitalPID_V100
注意事项	这个参数在增量 PID 和位置 PID 的计算中代表着不同的意思。 （式 2－4）所示位置式 PID 中 Ki 是积分系数 $\mathrm{Kp}\frac{T}{Ti}$ （式 2－5）所示增量式 PID 中 Ki 是 e_{k-1} 的系数 $Kp(1+\frac{2Td}{T})$
例　　子	PIDSetKi (0.367); //设置 Ki＝0.367

PIDGetKi( )

原　　形	double PIDGetKi(void)
描　　述	读取 PID 中所设置的 Ki 值
输入参数	无
输出参数	Ki 数值
头 文 件	Spmc75_PID.h
库 文 件	DigitalPID_V100
注意事项	这个参数在增量 PID 和位置 PID 的计算中代表着不同的意思。 （式 2－4）所示位置式 PID 中 Ki 是积分系数 $\mathrm{Kp}\frac{T}{Ti}$ （式 2－5）所示增量式 PID 中 Ki 是 e_{k-1} 的系数 $Kp(1+\frac{2Td}{T})$
例　　子	dKi ＝ PIDSetKi ();

PIDSetKd ( )

原　　形	void PIDSetKd(double dKdd)
描　　述	设置 PID 的 Kd 值
输入参数	Kd 数值
输出参数	无
头 文 件	Spmc75_PID.h
库 文 件	DigitalPID_V100
注意事项	这个参数在增量 PID 和位置 PID 的计算中代表着不同的意思。 （式 2－4）所示位置式 PID 中 Kd 是积分系数 $\mathrm{Kp}\frac{Td}{T}$ （式 2－5）所示增量式 PID 中 Kd 是 e_{k-2} 的系数 $Kp\frac{Td}{T}$
例　　子	PIDSetKd (0.157); //设置 Kd＝0.157

PIDGetKd ( )

原　　形	void PIDGetKd(double dKdd)
描　　述	读取 PID 中所设置的 Kd 值
输入参数	无
输出参数	Kd 数值
头 文 件	Spmc75_PID.h
库 文 件	DigitalPID_V100
注意事项	这个参数在增量 PID 和位置 PID 的计算中代表着不同的意思。 （式 2－4）所示位置式 PID 中 Kd 是积分系数 $\mathrm{Kp}\frac{Td}{T}$ （式 2－5）所示增量式 PID 中 Kd 是 e_{k-2} 的系数 $Kp\frac{Td}{T}$
例　　子	dKd ＝ PIDSetKd ();

IncPIDCalc ( )

原　　形	int IncPIDCalc(int)
描　　述	增量式 PID 计算
输入参数	PID 调节当前采样值
输出参数	计算增量
头 文 件	Spmc75_PID.h
库 文 件	DigitalPID_V100
注意事项	（式 2－5）增量式 PID 算法的实现。
例　　子	uiGoalvalue ＋＝ IncPIDCalc (1998); //位置式 PID 控制算法通过增量式控制算法递推实现，当前采样得到转速 1998rpm。

LocPIDCalc ( )

原　　形	unsigned int LocPIDCalc(int)
描　　述	位置式 PID 计算
输入参数	PID 调节当前采样值
输出参数	位置式 PID 计算出的绝对位置值
头 文 件	Spmc75_PID.h
库 文 件	DigitalPID_V100
注意事项	（式 2－4）位置式式 PID 算法的实现。
例　　子	uiGoalvalue ＝ LocPIDCalc (1998); //位置式 PID 控制算法，当前采样得到转速 1998rpm。

其它应用函数式对 BLDC 驱动的实现，不再一一赘述，可以参考【AN_SPMC75_0003】应用例的介绍。

4 程序范例

4.1 DEMO 程序

```
/*=========================================*/
//应用范例
/*=========================================*/
#include  "Spmc75_regs.h"
#include  "Spmc_typedef.h"
#include  "unspmacro.h"
#include  "Spmc75_BLDC.h"

main()
{
   P_IOA_SPE->W = 0x0000;
   P_IOB_SPE->W = 0x0000;
   P_IOC_SPE->W = 0x0000;

   Spmc75_System_Init();            //Spmc75 系统初始化

   while(1)
   {
       BLDC_Run_Service();          //启停监控
       NOP();
   }
}
```

```
//=================================================================
// Description: IRQ0 interrupt source is XXX,used to XXX
// Notes:错误保护
//=================================================================
void IRQ0(void) __attribute__ ((ISR));
void IRQ0(void)
{
   IPM_Fault_Protect();
}
```

```
//=================================================================
// Description: IRQ1 interrupt source is XXX,used to XXX
// Notes:BLDC 启动及正常运行服务
//=================================================================
void IRQ1(void) __attribute__ ((ISR));
```

```
void IRQ1(void)
{
   /*============================================================*/
   /*Position detection change interrupt
   /*============================================================*/
   if(P_TMR0_Status->B.PDCIF && P_TMR0_INT->B.PDCIE)
   {
       BLDC_Motor_Normalrun();
   }

   /*============================================================*/
   /*Timer Counter Overflow
   /*============================================================*/
   if(P_TMR0_Status->B.TCVIF && P_TMR0_INT->B.TCVIE)
   {
       BLDC_Motor_Startup();
   }
   P_TMR0_Status->W = P_TMR0_Status->W;
}
```

```
//=================================================================
// Description: IRQ6 interrupt source is XXX,used to XXX
// Notes:DMC 接收中断服务函数
//=================================================================
void IRQ6(void) __attribute__ ((ISR));
void IRQ6(void)
{
   if(P_INT_Status->B.UARTIF)
   {
       if(P_UART_Status->B.RXIF)  MC75_DMC_RcvStream();
       if(P_UART_Status->B.TXIF && P_UART_Ctrl->B.TXIE);
   }
}
```

```
//=================================================================
// Description: IRQ7 interrupt source is XXX,used to XXX
// Notes:512Hz 定时中断完成 PID 调节速度
//=================================================================
void IRQ7(void) __attribute__ ((ISR));
void IRQ7(void)
{
   if(P_INT_Status->B.CMTIF)
   {
       if(P_CMT_Ctrl->B.CM0IF && P_CMT_Ctrl->B.CM0IE)
```

```
        {
            BLDC_Motor_Actiyator();
        }
        P_CMT_Ctrl->W = P_CMT_Ctrl->W;
    }
}
```

PID 计算子函数:

```
//数据结构
typedef struct PID
{
   int  SetPoint;    //设定目标 Desired Value
   long SumError;             //误差累计

   double  Proportion;        //比例常数 Proportional Const
   double  Integral;          //积分常数 Integral Const
   double  Derivative;        //微分常数 Derivative Const

   int LastError;             //Error[-1]
   int PrevError;             //Error[-2]
} PID;
```

```
static PID sPID;
static PID *sptr = &sPID;

//PID 参数初始化
void IncPIDInit(void)
{
   sptr->SumError   = 0;
   sptr->LastError = 0;       //Error[-1]
   sptr->PrevError = 0;       //Error[-2]

   sptr->Proportion = 0;      //比例常数 Proportional Const
   sptr->Integral   = 0;      //积分常数 Integral Const
   sptr->Derivative = 0;      //微分常数 Derivative Const
   sptr->SetPoint   = 0;
}
```

```
//增量式 PID 控制设计
int IncPIDCalc(int NextPoint)
{
   register int iError, iIncpid;
   //当前误差
   iError = sptr->SetPoint - NextPoint;
   //增量计算
   iIncpid = sptr->Proportion * iError             //E[k]项
          - sptr->Integral   * sptr->LastError    //E[k－1]项
          + sptr->Derivative * sptr->PrevError;   //E[k－2]项
   //存储误差，用于下次计算
   sptr->PrevError = sptr->LastError;
   sptr->LastError = iError;
   //返回增量值
   return(iIncpid);
}
```

```
//位置式 PID 控制设计
unsigned int LocPIDCalc(int NextPoint)
{
    register int  iError,dError;

   iError = sptr->SetPoint - NextPoint;      //偏差
   sptr->SumError += iError;                 //积分
   dError = iError - sptr->LastError;        //微分
   sptr->LastError = iError;

   return(sptr->Proportion * iError          //比例项
         + sptr->Integral * sptr->SumError   //积分项
         + sptr->Derivative * dError);       //微分项
}
```

4.2 程序流程与说明

主程序主要完成系统必要的初始化，而对电动机的实时处理基本上是在中断中完成的，其中涉及到的中断主要有：IRQ0 的错误输入和输出中断、IRQ1 的 PDC 和 TCV 中断、IRQ6 的 UART RXD 中断及 CMT0 的定时中断。如图 5－1 所示主程序设计流程图。

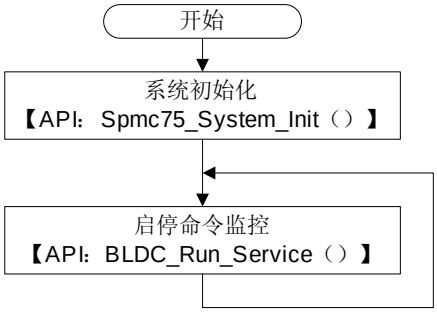

图 5-1　BLDC 主程序操作流程

4.3　中断子流程与说明

故障输入、输出短路、PDC、TCV、RXD 和 CMT0 等中断协助完成了对 BLDC 的启动、运行、速度调节和错误保护的控制。其中如果使用默认各个中断源的使用都按照初始化设置已经相应的固定下来。这里只对 PDC、TCV 中断流程示出，以便使用者参考。如图 5-2　PDC、TCV 中断操作流程

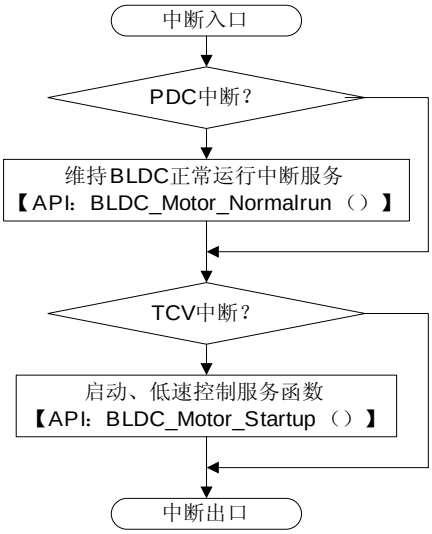

图 5-2　PDC、TCV 中断操作流程

5 MCU 使用资源

5.1 MCU 硬件使用资源说明

CPU 型号	SPMC75F2413A	封装	QFP80-0.8
振荡器	☒ crystal	频率	6MHz
	☐外部	输入频率	
WATCHDOG	☒ 有 ☐ 无	☐ 启用 ☒ 未启用	
IO 口使用情况	使用	IOB[0..6]：电机驱动接口 IOB[8..10]：位置侦测接口 IOB14：电机驱动使能 IOC[0..1]：UART	
Timer 使用情况	PDC0	位置侦测和速度测量	
	MCP3	电机驱动信号发生	
	CMT0	PID 调整服务定时器	
中断使用情况	PDC0（IRQ1）：速度测量和定时发生 MCP3（IRQ3）：电机驱动信号发生 UART（IRQ6）：主机通信服务 CMT0（IRQ7）：PID 调整服务定时器		
ROM 使用情况	6.34K Words		

6 实验测试

测试主要是针对 120 度上相 PWM 方波驱动带霍尔位置传感的直流无刷电动机并应用 PID 控制来进行对电动机的速度调节。硬件原理图及关于 BLDC 驱动请参照【AN_SPMC75_0003】。Kp 参数的调整，在源程序中是可以用预编译定义的。如果定义的话就可以在 DMC 下对 Kp 参数调整，但请注意，DMC 下的参数需要是扩大一千倍的。例如：Kp＝0.105，DMC 参数就应该是 105。这样 Kp 的范围就可以在（400，10），对于这个范围的参数只能说是能正常的工作。如果说最适合，那么还请您根据系统的需要，按照各种整定方法对其进行全面的评估。

【注意】 1、在这里 PID 参数的选择和 PWM 的载波频率也有一定的关系，载波频率越高则【P_TMRx_TGRA】所设置的范围就越小，调节就比较的快。这种种因数希望到能考虑在内，得到最适合的 PID 参数。

【注意】 2、如果选择使用 DMC 来调整 PID 的 Kp 参数，请先设置转速（Speed1_Cmd）和 Kp（Speed1_Kp）的数值再启动 Motor1；若是使用固定的 PID 的 Kp 参数（在程序中固定），请先设置转速（Speed1_Cmd）再启动 Motor1。

【注意】 3、在 DMC 下 PID 参数的时候 Speed1_Kp 与 Kp 的关系是 Speed1_Kp＝1000Kp，就是说如果要设置 Kp 为 0.125，DMC 的 Speed1_Kp 应该给出 125。

6.1 响应曲线

所有测试都是在空载，PWM 载波为 6KHz 的情况下进行：

1、 Speed1_Kp＝10，即 Kp＝0.01。在转速从 0rpm 升至 2000rpm 的响应曲线，如图 6.1。

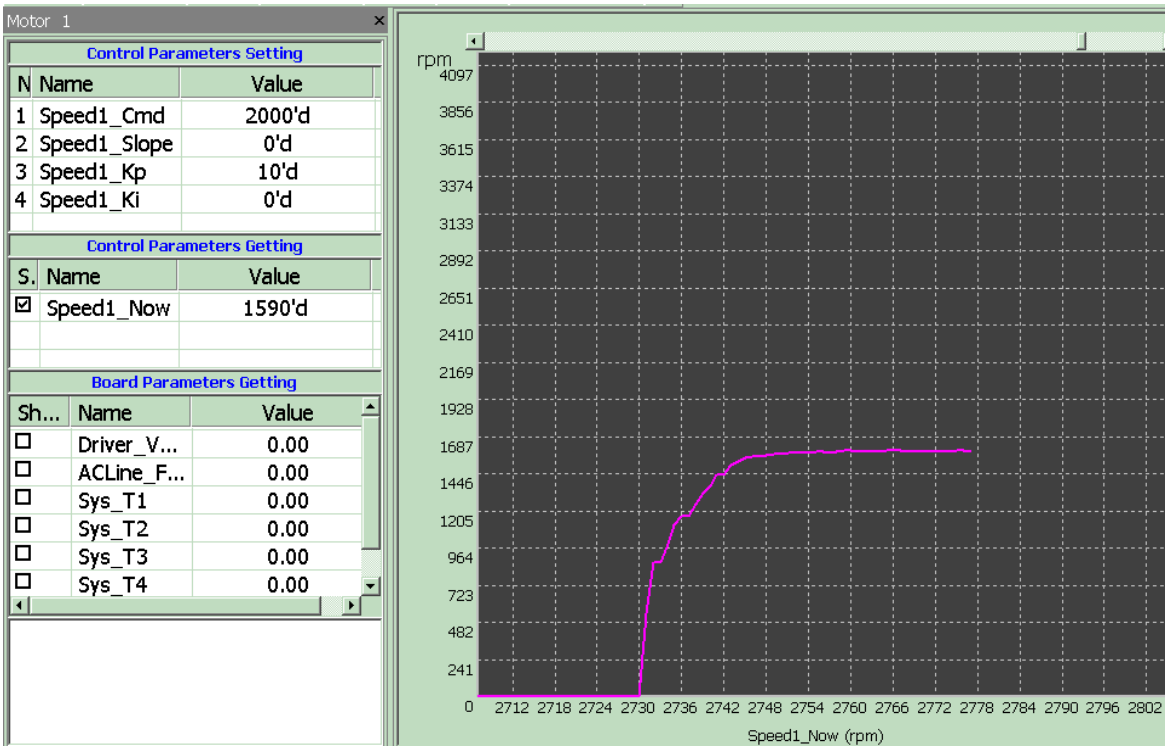

图 6.1 Kp＝0.01 响应曲线

2、Speed1_Kp＝105，即 Kp＝0.105。在转速从 0rpm 升至 2000rpm 的响应曲线，如图 6.2。

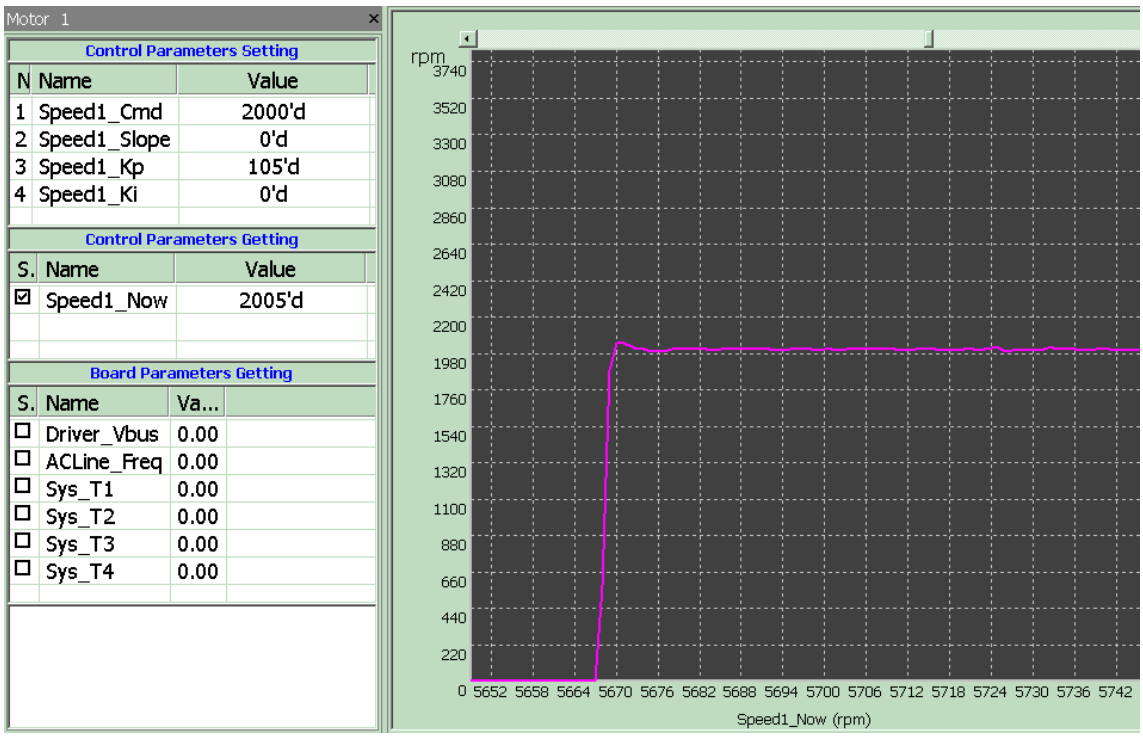

图 6.2　Kp＝0.105 响应曲线

3、 Speed1_Kp＝180，即 Kp＝0.180。在转速从 0rpm 升至 2000rpm 的响应曲线，如图 6.3。

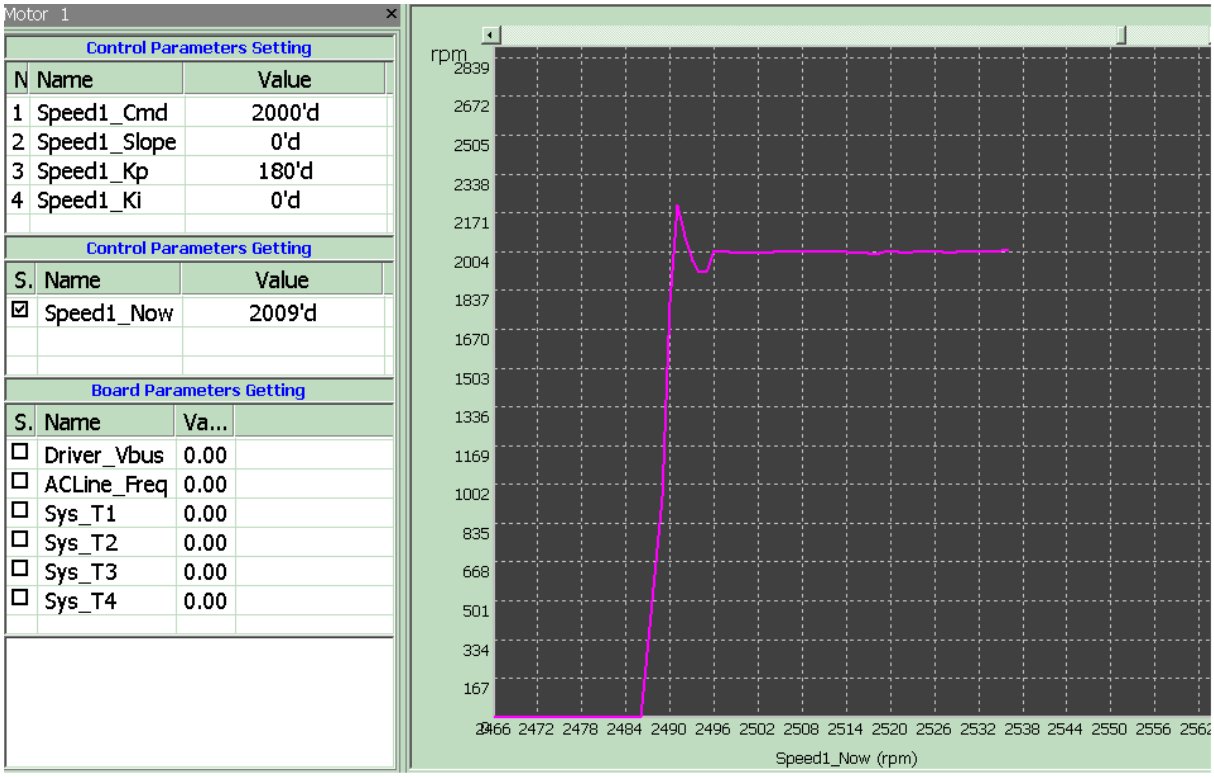

图 6.3 Kp＝0.180 响应曲线

4、 Speed1_Kp＝250，即 Kp＝0.250。在转速从 0rpm 升至 2000rpm 的响应曲线，如图 6.4。

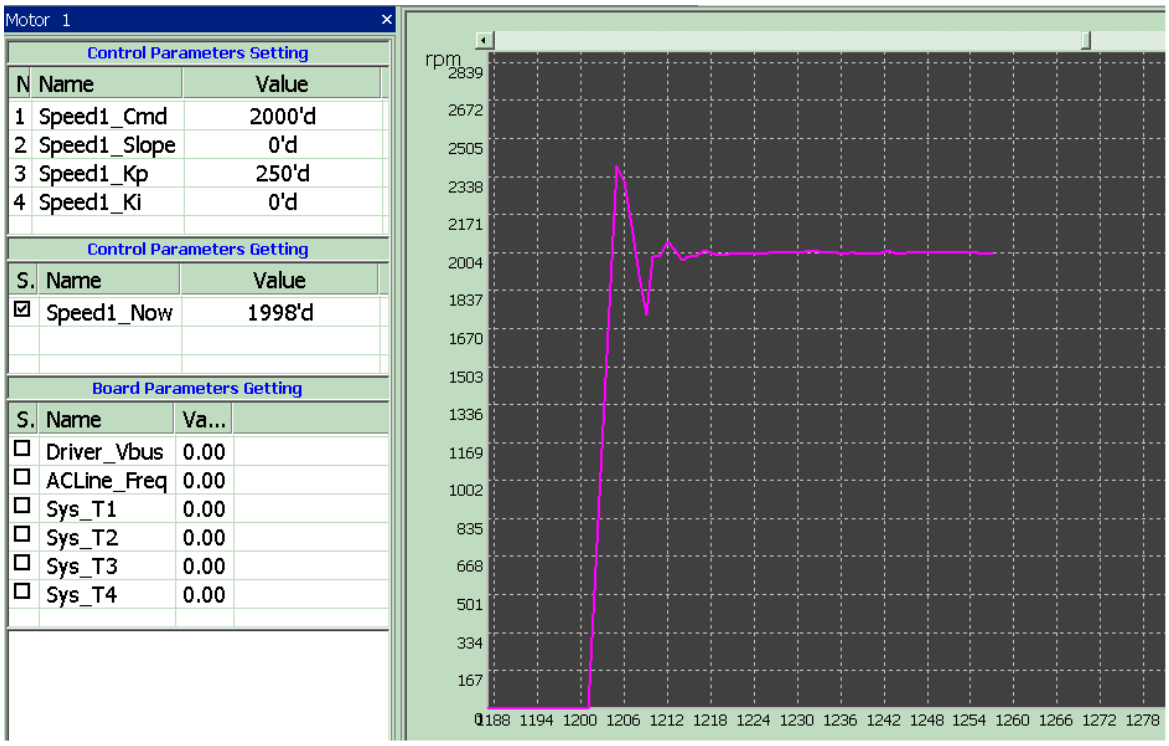

图 6.4 Kp＝0.250 响应曲线