## 读飞控源码：PID控制原型



APM中用PID控制roll，pitch，yaw三个轴，

从apm的void loop()函数进入主要循环的static void fast\_loop()，再进入run\_rate\_controllers()函数，该函数主要进行速度的控制，采用PID控制，函数原型如下:

 

get\_heli\_rate\_roll(roll\_rate\_target\_bf);

get\_heli\_rate\_pitch(pitch\_rate\_target\_bf);

get\_heli\_rate\_yaw(yaw\_rate\_target\_bf);

该三个函数主要控制飞控的roll，pitch，yaw，本文主要介绍get\_heli\_rate\_roll(roll\_rate\_target\_bf);函数的控制，其他两个控制一样，再不重复介绍。get\_heli\_rate\_roll(roll\_rate\_target\_bf);先贴上PID控制原型：

 

再贴上get\_heli\_rate\_roll(roll\_rate\_target\_bf);函数原型：





两张截图

前面定义了:相关需要用到的变量，

int32\_t p,i,d,ff;     //PID需要用到的参数保存变量

int32\_t current\_rate;  //用来保存当前获取的速度

int32\_t rate\_error;   //用来保存当前速度与目标速度的差值

int32\_t output;  //最后计算的结果进行保存参数输出，传递给电机库函数

current\_rate    = (omega.x \* DEGX100);// omega.x对应x轴的角速度roll，

DEGX100常量值等于5729.57795f，

current\_rate = rate\_roll\_filter.apply(current\_rate);调用滤波器进行数据过滤

apply(current\_rate)滤波函数原型如下：



rate\_error  = target\_rate - current\_rate;//目标速度减去当前速度得到误差，目标速度来自函数参数的传递，

p   = g.pid\_rate\_roll.get\_p(rate\_error);//此处进行的参数的传递，调用PID里面的p（比例）进行计算。

get\_p(rate\_error);函数原型如下：

 

计算后进行参数返回传递

if (motors.flybar\_mode == 1) { //进行飞行模式的判断

if (target\_rate > -50 && target\_rate < 50){  //判断目标速度的范围

i= g.pid\_rate\_roll.get\_i(rate\_error, G\_Dt);  //进行积分0.02

get\_i(rate\_error, G\_Dt);函数原型如下：



} else {

i= g.pid\_rate\_roll.get\_integrator();//获取积分器

}} else {

i= g.pid\_rate\_roll.get\_leaky\_i(rate\_error, G\_Dt, RATE\_INTEGRATOR\_LEAK\_RATE);

get\_leaky\_i(rate\_error, G\_Dt, RATE\_INTEGRATOR\_LEAK\_RATE);函数原型如下：



G\_Dt等于0.02，RATE\_INTEGRATOR\_LEAK\_RATE=0.02f

d = g.pid\_rate\_roll.get\_d(rate\_error, G\_Dt);//进行微分，

原型如下：



ff = g.heli\_roll\_ff \* target\_rate;//前馈系数

output = p + i + d + ff;//进行各个计算后参数相加，同上最开始的公式，

output = constrain\_int32(output, -4500, 4500);//限制输出，将控制在45°范围之内，

#if LOGGING\_ENABLED == ENABLED  //判断是否启用日志记录数据

    // log output if PID logging is on and we are tuning the rate P, I or D gains    if( g.log\_bitmask & MASK\_LOG\_PID && (g.radio\_tuning == CH6\_RATE\_KP || g.radio\_tuning == CH6\_RATE\_KI || g.radio\_tuning == CH6\_RATE\_KD) ) {        pid\_log\_counter++;        if( pid\_log\_counter >= 10 ) {               // (update rate / desired output rate) = (100hz / 10hz) = 10            pid\_log\_counter = 0;            Log\_Write\_PID(CH6\_RATE\_KP, rate\_error, p, i, d, output, tuning\_value);        }    }#endif   //此处是进行数据的保存，方便分析飞行数据。

return output;//输出控制   此值计算出来返回给 电机