http://www.51hei.com/bbs/dpj-156070-1.html

用整型变量来实现PID算法，由于是用整型数来做的,所以也不是很精确,但是对于很多  
的使用场合,这个精度也够了，关于系数和采样电压全部是放大10倍处理的.所以精度  
不是很高. 但是也不是那么低,大部分的场合都够了. 实在觉得精度不够, 可以再放大  
10倍或者100倍处理,但是要注意不超出整个数据类型的范围就可以了.本程序包括PID计  
算和输出两部分.当偏差>10度全速加热,偏差在10度以内为PID计算输出. 具体的参考代  
码参见下面：\*/  
//================================================================

//================================================================

// pid.H

// Operation about PID algorithm procedure

// C51编译器 Keil 7.08

//================================================================

// 作者:zhoufeng

// Date :2007-08-06

// All rights reserved.

//================================================================

#include <reg52.h>

#include <intrins.h>

typedef unsigned char uint8;

typedef unsigned int uint16;

typedef unsigned long int uint32;

/\*\*\*\*\*\*\*\*\*\*函数声明\*\*\*\*\*\*\*\*\*\*\*\*/

void PIDOutput ();

void PIDOperation ();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef struct PIDValue {

uint32 Ek\_Uint32[3]; //差值保存，给定和反馈的差值

uint8 EkFlag\_Uint8[3]; //符号，1则对应的为负数，0为对应的为正数

uint8 KP\_Uint8;

uint8 KI\_Uint8;

uint8 KD\_Uint8;

uint16 Uk\_Uint16; //上一时刻的控制电压

uint16 RK\_Uint16; //设定值

uint16 CK\_Uint16; //实际值

} PIDValueStr;

PIDValueStr PID;

uint8 out ; // 加热输出

uint8 count; // 输出时间单位计数器

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PID = Uk + KP\*[E(k)-E(k-1)]+KI\*E(k)+KD\*[E(k)-2E(k-1)+E(k-2)];(增量型PID算式)

函数入口: RK(设定值),CK(实际值),KP,KI,KD

函数出口: U(K)

//PID运算函数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void PIDOperation (void) {

uint32 Temp[3]; //中间临时变量

uint32 PostSum; //正数和

uint32 NegSum; //负数和

Temp[0] = 0;

Temp[1] = 0;

Temp[2] = 0;

PostSum = 0;

NegSum = 0;

if( PID.RK\_Uint16 > PID.RK\_Uint16 ) { //设定值大于实际值否？

if( PID.RK\_Uint16 - PID.RK\_Uint16 >10 ) { //偏差大于10否？

PID.Uk\_Uint16 = 100;

} //偏差大于10为上限幅值输出(全速加热)

else {

Temp[0] = PID.RK\_Uint16 - PID.CK\_Uint16; //偏差<=10,计算E(k)

PID.EkFlag\_Uint8[1]=0; //E(k)为正数

//数值移位

PID.Ek\_Uint32[2] = PID.Ek\_Uint32[1];

PID.Ek\_Uint32[1] = PID.Ek\_Uint32[0];

PID.Ek\_Uint32[0] = Temp[0];

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if( PID.Ek\_Uint32[0] >PID.Ek\_Uint32[1] ) { //E(k)>E(k-1)否？

Temp[0]=PID.Ek\_Uint32[0] - PID.Ek\_Uint32[1]; //E(k)>E(k-1)

PID.EkFlag\_Uint8[0]=0;

} //E(k)-E(k-1)为正数

else {

Temp[0]=PID.Ek\_Uint32[0] - PID.Ek\_Uint32[1]; //E(k)<E(k-1)

PID.EkFlag\_Uint8[0]=1;

} //E(k)-E(k-1)为负数

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Temp[2]=PID.Ek\_Uint32[1]\*2 ; // 2E(k-1)

if( (PID.Ek\_Uint32[0]+ PID.Ek\_Uint32[2])>Temp[2] ) { //E(k-2)+E(k)>2E(k-1)否？

Temp[2]=(PID.Ek\_Uint32[0]+ PID.Ek\_Uint32[2])-Temp[2]; //E(k-2)+E(k)>2E(k-1)

PID.EkFlag\_Uint8[2]=0;

} //E(k-2)+E(k)-2E(k-1)为正数

else {

Temp[2]=Temp[2]-(PID.Ek\_Uint32[0]+ PID.Ek\_Uint32[2]); //E(k-2)+E(k)<2E(k-1)

PID.EkFlag\_Uint8[2]=1;

} //E(k-2)+E(k)-2E(k-1)为负数

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Temp[0] = (uint32)PID.KP\_Uint8 \* Temp[0]; // KP\*[E(k)-E(k-1)]

Temp[1] = (uint32)PID.KI\_Uint8 \* PID.Ek\_Uint32[0]; // KI\*E(k)

Temp[2] = (uint32)PID.KD\_Uint8 \* Temp[2]; // KD\*[E(k-2)+E(k)-2E(k-1)]

/\*以下部分代码是讲所有的正数项叠加，负数项叠加\*/

/\*\*\*\*\*\*\*\*\*\*KP\*[E(k)-E(k-1)]\*\*\*\*\*\*\*\*\*\*/

if(PID.EkFlag\_Uint8[0]==0)

PostSum += Temp[0]; //正数和

else

NegSum += Temp[0]; //负数和

/\*\*\*\*\*\*\*\*\* KI\*E(k)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(PID.EkFlag\_Uint8[1]==0)

PostSum += Temp[1]; //正数和

else

; //空操作，E(K)>0

/\*\*\*\*KD\*[E(k-2)+E(k)-2E(k-1)]\*\*\*\*/

if(PID.EkFlag\_Uint8[2]==0)

PostSum += Temp[2]; //正数和

else

NegSum += Temp[2]; //负数和

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*U(K)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PostSum += (uint32)PID.Uk\_Uint16;

if(PostSum > NegSum ) { // 是否控制量为正数

Temp[0] = PostSum - NegSum;

if( Temp[0] < 100 ) //小于上限幅值则为计算值输出

PID.Uk\_Uint16 = (uint16)Temp[0];

else

PID.Uk\_Uint16 = 100; //否则为上限幅值输出

} else //控制量输出为负数，则输出0(下限幅值输出)

PID.Uk\_Uint16 = 0;

}

} else {

PID.Uk\_Uint16 = 0;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数入口: U(K)

函数出口: out(加热输出)

//PID运算植输出函数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void PIDOutput (void) {

static int i;

i=PID.Uk\_Uint16;

if(i==0)

out=1;

else out=0;

if((count++)==5) { //如定时中断为40MS,40MS\*5=0.2S(输出时间单位),加热周期20S(100等份)

//每20S PID运算一次

count=0;

i--;

}

}