**电子设计竞赛总结**

1. **问题重述**

**赛题：单相用电负载分析**

**内容：通过一定的检测装置，通过分析用电器的电参数得到用电器的相关信息，从而得知该用电器的类型。**

**并且分析装置具有一定的学习功能。**

**（详细内容请上网搜索2017年电子设计竞赛国赛）**

1. **总结**
2. **比赛之前的准备阶段假如选择检测类题目，个人建议采用已有的集成产品。比如：芯片。**

**由于个人的搭设硬件电路能力有限，一旦出现故障，没有坚实的理论基础，可能导致比赛后续内容没有进展，甚至无法达到比赛的基本要求。**

1. **基于以上分析，建议选择该类型题的同学，平时多做练习，能够学会通过查看芯片手册，完成单片机对芯片的操作，读取相应的参数。（该题中可用到的一个电能芯片cs5463，该芯片具有通过电流电压互感器就能计算电流电压（相线值）、相位、功率、温度、容性感性等等，应有尽有，并且带有滤波功能）**
2. **小贴士：一般成熟芯片大多存在以下引脚**

**片选CS引脚，通用总线IIC,SPI引脚等。**

1. **电赛最主要还是大家在平时的积累和练习。硬件上：尽可能多的调试你所选题目的可能用到的模块。不至于比赛浪费时间在调试模块上。**
2. **如果时间允许，可以多准备不同的赛题方向。**
3. **控制类赛题同学，应熟练使用PID，熟悉各个作用，理解PID口诀以及经验调试方法（已有测量曲线和输出曲线），如：等幅震荡法、临界比例度法**

**PID参数调试别无他法，只有通过多练习，多积累。**

**（具体一些本人的体会总结，可以通过查看西门子比赛总结ＰＩＤ部分）**

**７、附录给出两个程序源码，一个ｃｓ５４６３参考程序，一个参考ＰＩＤ算法Ｃ语言实现程序。**

**８、ＰＩＤ的ｃ语言实现中，是通过利用ｐｉｄ离散数字化达到ｐｉｄ效果。记住：ｐｉｄ的各个作用都是相对误差而言。时常想起ｐｉｄ的公式，有利于对ｐｉｄ的理解。**

**比例实现较好理解，直接通过向误差乘以一个比例系数。**

**积分作用的实际意义就是对误差的积累，只要存在过去的误差，就存在积分作用。如果理解积分作用意义，那么程序中对过去误差求和就不难理解了。**

**微分作用的实际意义是对未来的预测，具体预测的是误差的走势。作用大小决定于误差的斜率，因此，程序通过对过去的误差求差。**

**说明：错误之处，感谢大家指正。**

**附录：cs5463部分操作51程序（未调试）**

#include <reg52.h>

#define uchar unsigned char

#define uint unsigned char

int count1s;

unsigned int count1m;

unsigned char dis\_bitcount=0;

unsigned char display\_o[6];

unsigned char display\_q[6];

unsigned char display\_pf[6];

unsigned char display\_temp[6];

//unsigned SEG[30]=

unsigned char dis\_bitdriver=0;

static uchar idata cs5463rw[3]={0,0,0};

static uchar idata cs5463tm[3]={0,0,0};

sbit reset=P2^1;

sbit cs=P2^0;

sbit sdi=P2^5;

sbit sdo=P2^2;

sbit sclk=P2^4;

sbit int\_5463=P2^3;

sbit s1=P3^3;

unsigned char result\_h;

unsigned char result\_l;

unsigned int pf,tem;

void delay1ms(int count)

{

int j,k;

while(count--!=0)

{

for(j=0;j<9;j++)

for(k=0;k<15;k++);

}

}

void nop\_1us(void)

{

uchar i;

for(i=0;i<3;i++);

}

void send\_5463(uchar send\_data)

{

uchar i;

sclk=0;

for(i=0;i<8;i++)

{

if(send\_data & 0x80)

sdi=1;

else

sdi=0;

send\_data<<=1;

nop\_1us();

sclk=0;

}

}

void write\_5463(uchar \*pbuf,uchar command)

{

uchar j,k;

send\_5463(command);

for(j=0;j<3;j++)

{

if(\*pbuf&0x80)

sdi=1;

else

sdi=0;

sclk=1;

\*pbuf<<=1;

nop\_1us();

sclk=0;

}

pbuf+=1;

}

void read\_5463(uchar \*pbuf,uchar command)

{

uchar j,k,const\_fe;

send\_5463(command);

for(j=0;j<3;j++)

{

\*pbuf=0;

const\_fe=0xfe;

for(k=0;k<8;k++)

{

sclk=0;

\*pbuf<<=1;

if(sdo)

\*pbuf |=0x01;

if(const\_fe\*0x80)

sdi=1;

else

sdi=0;

const\_fe<<=1;

nop\_1us();

sclk=1;

}

pbuf+=1;

}

}

void init(void)

{

uchar \*temp;

sdi=0;

sclk=0;

nop\_1us();

cs=0;

reset=0;

delay1ms(6);

reset=1;

nop\_1us();

send\_5463(0xff);

send\_5463(0xff);

send\_5463(0xff);

send\_5463(0xff);

send\_5463(0xff);

send\_5463(0xff);

nop\_1us();

send\_5463(0xfe);

cs5463rw[0]=0x00;

cs5463rw[1]=0x00;

cs5463rw[2]=0x00;

temp=&cs5463rw[0];

write\_5463(temp,0x40);

delay1ms(1);

cs5463rw[0]=0x00;

cs5463rw[1]=0x0f;

cs5463rw[2]=0xa0;

temp=&cs5463rw[0];

write\_5463(temp,0x4a);

delay1ms(1);

cs5463rw[0]=0x00;

cs5463rw[1]=0x00;

cs5463rw[2]=0x00;

temp=&cs5463rw[0];

write\_5463(temp,0x74);

delay1ms(1);

cs5463rw[0]=0x00;

cs5463rw[1]=0x00;

cs5463rw[2]=0x00;

temp=&cs5463rw[0];

write\_5463(temp,0x78);

delay1ms(1);

cs5463rw[0]=0xff;

cs5463rw[1]=0xff;

cs5463rw[2]=0xff;

temp=&cs5463rw[0];

write\_5463(temp,0x53);

delay1ms(1);

send\_5463(0xe8);

}

void ceshi(void)

{

uchar \*tem0;

unsigned char bw,sw,gw,qsw;

temp=&cs5463rw[0];

read\_5463(tem0,0x1e);

cs5363tm[0]=cs5463rw[0];

cs5363tm[1]=cs5463rw[1];

cs5363tm[2]=cs5463rw[2];

if(cs5463rw[0]&0x80)

{

temp=&cs5463rw[0];

read\_5463(tem0,0x32);

send(cs5463rw[0]);

send(cs5463rw[1]);

temp0=&cs5463rw[0];

read\_5463(temp,0x26); //read wendu

send\_2asc(cs5463rw[0]);

send\_2asc(cs5463rw[1]);

temp=&cs5463rw[0];

read\_5463(temp,0x18); //read vrms

send\_2sasc(cs5463rw[0]);

send\_2sasc(cs5463rw[1]);

temp=&cs5463rw[0];

read\_5463(temp,0x16); //read lrms

send\_2sasc(cs5463rw[0]);

send\_2sasc(cs5463rw[1]);

temp=&cs5463rw[0];

read\_5463(temp,0x1a); //read frms

send\_2sasc(cs5463rw[0]);

send\_2sasc(cs5463rw[1]);

temp=&cs5463rw[0];

if(cs5463rw[0]>0x80)

{

cs5463rw[0]=~cstryerw[0]+1;

pf=cs5463rw[0];

display\_pf[3]=20;

}

else

{

pf=cs5463rw[0];

display\_pf[3]=20;

}

}

cs5463rw[0]=cs5463tm[0];

cs5463rw[1]=cs5463tm[1];

cs5463rw[2]=cs5463tm[2];

temp=&cs5463rw[0];

write\_5463(temp,0x5e);

}

main()

{

unsigned char p2code=0;

count1m=0;

count1s=0;

delay1ms(100);

init\_5463();

while(1)

{

switch(dis\_bitcount\_1)

{

case1:dis\_bitdriver=0x01; break;

case2:dis\_bitdriver=0x02; break;

case3:dis\_bitdriver=0x02; break;

default:dis\_bitdriver=0x08; break;

}

p2code=P1;

p2code=p2code&0xf0;

p2code=p2code|dis\_bitdriver;

P0=SEG[display\_1[dis\_bitcount]];

P1=p2code;

if(dis\_bitcount>=3) dis bitcount=0;

else dis\_bitcount++;

count1s++;

if(count1s>=200)

{

count1m++;

count1s=0;

ceshi(); //测试功率因数

}

else

delay1ms(4);

}

display\_q[3]=display\_pf[3];

display\_q[3]=display\_pf[3];

display\_q[3]=display\_pf[3];

display\_q[3]=display\_pf[3];

}

ＰＩＤ算法ｃ语言源码：

//PID控制，参数1：sv(设定值)，参数2：fv(反馈值)

void PID\_Control(float sv,float fy)

{

float pError=0,iError=0,dError=0;

ek=sv-fv;

pError=ek-ek1; //比例误差(等于当前误差减去前一次的误差)

iError=ek; //积分误差(等于当前误差值)

dError=ek-ek1\*2+ek2; //微分误差(等于当前误差减去前一次2倍误差再加上前两次的误差)

ek2=ek1; //储存前两次的误差值

ek1=ek; //储存前一次的误差值

u=Kp\*pError+Ki\*iError+Kd\*dError+u1; //获取PID调节的误差值

u1=u; //储存前一次的输出值

}