

```
//SIDI1.C    simple digital interferometer
```

```
//          Marko Cebokli feb 2004  GNU/GPL licence
```

```
//          www.gnu.org/licences/licences.html#GPL
```

```
//  version 1 - software sample timing
```

```
//input connection LPT port D25
```

```
//          pin 11 channel 1 I          bit 7
```

```
//          pin 10 channel 1 Q          bit 6
```

```
//          pin 12 channel 2 I          bit 5
```

```
//          pin 13 channel 2 Q          bit 4
```

```
//          pin 18..25 ground
```

```
#include<stdio.h>
```

```
#include<time.h>
```

```
#include<dos.h>    //inportb
```

```
#include<conio.h>  //kbhit,getch
```

```
#include<stdlib.h> //exit()
```

```
#include<math.h>
```

```
#include <graphics.h>
```

```
unsigned int sig[32767];
```

```
char ct0real[65535],ct0imag[65535]; //veliki arrayi zunaj, sicer crash
```

```
char dctab[65535];
```

```
//-----
```

```
//mkcortab prepares lookup tables for the correlator
```

```
//data input format: AaBbAaBbAaBbAaBb (chans A and B, In phase X quadrature x)
```

```
//can prepare the table for 4 possible phases - faza=0,1,2,3
```

```
//0=0deg, 1=90deg, 2=180 deg, 3=270deg (for fringe rotation)
```

```
void mkcortab(char faza,char real[],char imag[])
```

```
{
```

```
char faza0r[]={1,0,0,-1,0,1,-1,0,0,-1,1,0,-1,0,0,1};
```

```
char faza0i[]={0,1,-1,0,-1,0,0,1,1,0,0,-1,0,-1,1,0};
```

```
char faza90r[]={0,1,-1,0,-1,0,0,1,1,0,0,-1,0,-1,1,0};
```

```
char faza90i[]={-1,0,0,1,0,-1,1,0,0,1,-1,0,1,0,0,-1};
```

```
char faza180r[]={-1,0,0,1,0,-1,1,0,0,1,-1,0,1,0,0,-1};
```

```
char faza180i[]={0,-1,1,0,1,0,0,-1,-1,0,0,1,0,1,-1,0};
```

```

char faza270r[]={0,-1,1,0,1,0,0,-1,-1,0,0,1,0,1,-1,0};
char faza270i[]={1,0,0,-1,0,1,-1,0,0,-1,1,0,-1,0,0,1};
long j;
int b1,b2,b3,b4;

```

```

for (j=0;j<65536;j++)
{
  b1=(j&0x0000000F);
  b2=(j&0x000000F0)>>4;
  b3=(j&0x00000F00)>>8;
  b4=(j&0x0000F000)>>12;
  switch (faza)
  {
    case 0:{ real[j]=faza0r[b1]+faza0r[b2]+faza0r[b3]+faza0r[b4];
            imag[j]=faza0i[b1]+faza0i[b2]+faza0i[b3]+faza0i[b4];break;}
    case 1:{ real[j]=faza90r[b1]+faza90r[b2]+faza90r[b3]+faza90r[b4];
            imag[j]=faza90i[b1]+faza90i[b2]+faza90i[b3]+faza90i[b4];break;}
    case 2:{ real[j]=faza180r[b1]+faza180r[b2]+faza180r[b3]+faza180r[b4];
            imag[j]=faza180i[b1]+faza180i[b2]+faza180i[b3]+faza180i[b4];break;}
    case 3:{ real[j]=faza270r[b1]+faza270r[b2]+faza270r[b3]+faza270r[b4];
            imag[j]=faza270i[b1]+faza270i[b2]+faza270i[b3]+faza270i[b4];break;}
    default:{ printf("\n Napaka v mkcortab: faza= %d",faza);exit(1);}
  }
}

```

```

//-----
//mkdctab prepares a lookup table for fast DC offset determination
//tests bits 0,4,8,12  others are done by shifting

```

```

void mkdctab(char dctab[])
{
  long j;
  for (j=0;j<65536;j++)
  {
    dctab[j]=-2;
    if ((j&0x1)==0x1) dctab[j]=dctab[j]+1;
    if ((j&0x10)==0x10) dctab[j]=dctab[j]+1;
    if ((j&0x100)==0x100) dctab[j]=dctab[j]+1;
    if ((j&0x1000)==0x1000) dctab[j]=dctab[j]+1;
  }
}
//-----

```

```

void vzorcenje() //this one does the sampling
{
    //array is written each time to equalize timing
int i;
long j;
unsigned char a;

for (j=0;j<131072;j++) //131072
{
    i=(int)(j>>2);
    a=inportb(0x379)>>4; //D25: *pin11 pin10 pin12 pin13 (pin15)
    sig[i]=(sig[i]<<4)+a;
}
}

```

```
//-----
```

```

void checkdc() //calculate the DC offsets for adjustment
{
long j,dc1,dc2,dc3,dc4;
float dc1p,dc2p,dc3p,dc4p;
unsigned int i;

dc1=0;dc2=0;dc3=0;dc4=0;
for (j=0;j<32768;j++)
{
    i=sig[j]^0x8888;
    if ((i&0x1)==0x1) dc1=dc1+1; //Q2
    if ((i&0x2)==0x2) dc2=dc2+1; //I2
    if ((i&0x4)==0x4) dc3=dc3+1; //Q1
    if ((i&0x8)==0x8) dc4=dc4+1; //I1
    if ((i&0x10)==0x10) dc1=dc1+1;
    if ((i&0x20)==0x20) dc2=dc2+1;
    if ((i&0x40)==0x40) dc3=dc3+1;
    if ((i&0x80)==0x80) dc4=dc4+1;
    if ((i&0x100)==0x100) dc1=dc1+1;
    if ((i&0x200)==0x200) dc2=dc2+1;
    if ((i&0x400)==0x400) dc3=dc3+1;
    if ((i&0x800)==0x800) dc4=dc4+1;
    if ((i&0x1000)==0x1000) dc1=dc1+1;
    if ((i&0x2000)==0x2000) dc2=dc2+1;
    if ((i&0x4000)==0x4000) dc3=dc3+1;
    if ((i&0x8000)==0x8000) dc4=dc4+1;
}
}

```

}

dc1p=(dc1-65536)/655.36;dc2p=(dc2-65536)/655.36;

dc3p=(dc3-65536)/655.36;dc4p=(dc4-65536)/655.36;

//printf("\nDC: I1=%lu ",dc4);printf("%lu Q1=",dc3);

//printf("%lu I2=",dc3);printf("%lu Q2=",dc1);

printf("\n DC offsets: ");

printf(" I1=%5.2f%% ",dc4p);printf("Q1=%5.2f%% ",dc3p);

printf("I2=%5.2f%% ",dc2p);printf("Q2=%5.2f%% ",dc1p);

}

//-----

void checkquad() //check quadrature of I/Q channels

{ //with noise input, perfect quadrature

long j,dc1,dc2,dc3,dc4,k1,k2; //means no correlation between I and Q

unsigned int i,k;

float q1,q2;

k1=0;k2=0;

for (j=0;j<32768;j++)

{

i=sig[j]^0x8888;k=i;

k=k^(k>>1);

if((k&0x1)==0x1) k1=k1+1;

if((k&0x4)==0x4) k2=k2+1;

if((k&0x10)==0x10) k1=k1+1;

if((k&0x40)==0x40) k2=k2+1;

if((k&0x100)==0x100) k1=k1+1;

if((k&0x400)==0x400) k2=k2+1;

if((k&0x1000)==0x1000) k1=k1+1;

if((k&0x4000)==0x4000) k2=k2+1;

dc1=dc1+(long)dctab[i]; //Q2

i=i>>1;dc2=dc2+(long)dctab[i]; //I2

i=i>>1;dc3=dc3+(long)dctab[i]; //Q1

i=i>>1;dc4=dc4+(long)dctab[i]; //I1

}

q1=(k1-65536.0)/65536; //tukaj se DC korekcija!

q2=(k2-65536.0)/65536;

printf("\n Quadrature %1.5f %1.5f",q1,q2);printf(" %ld %ld",k1,k2);

}

//-----

void korelator(float *korr,float *kori) //here correlation is done using lookup tables

```
{
long j,rkor,ikor,dc1,dc2,dc3,dc4;
float dckorr,dckori,dckr,dcki;
float pdc1,pdc2,pdc3,pdc4;
float p0,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14,p15;
unsigned int i;
```

rkor=0;ikor=0;dc1=0;dc2=0;dc3=0;dc4=0;

for (j=0;j<32768;j++)

```
{
i=sig[j]^0x8888; //correction for LPT inverted status bits
rkor=rkor+(long)ct0real[i];
ikor=ikor+(long)ct0imag[i];
dc1=dc1+(long)dctab[i]; //Q2
i=i>>1;dc2=dc2+(long)dctab[i]; //I2
i=i>>1;dc3=dc3+(long)dctab[i]; //Q1
i=i>>1;dc4=dc4+(long)dctab[i]; //I1
}
```

*korr=(float)rkor/131072;

*kori=(float)ikor/131072;

dckorr=((float)dc1*dc3+(float)dc2*dc4)/8589934592.0; //false correlation

dckori=((float)dc3*dc2-(float)dc1*dc4)/8589934592.0; //caused by DC bias

//858.... = 2^33

pdc1=(float)dc1/131072+0.5;pdc2=(float)dc2/131072+0.5;

pdc3=(float)dc3/131072+0.5;pdc4=(float)dc4/131072+0.5;

p0=(1-pdc1)*(1-pdc2)*(1-pdc3)*(1-pdc4);p1=(1-pdc1)*(1-pdc2)*(1-pdc3)*pdc4;

p2=(1-pdc1)*(1-pdc2)*pdc3*(1-pdc4);p3=(1-pdc1)*(1-pdc2)*pdc3*pdc4;

p4=(1-pdc1)*pdc2*(1-pdc3)*(1-pdc4);p5=(1-pdc1)*pdc2*(1-pdc3)*pdc4;

p6=(1-pdc1)*pdc2*pdc3*(1-pdc4);p7=(1-pdc1)*pdc2*pdc3*pdc4;

p8=pdc1*(1-pdc2)*(1-pdc3)*(1-pdc4);p9=pdc1*(1-pdc2)*(1-pdc3)*pdc4;

p10=pdc1*(1-pdc2)*pdc3*(1-pdc4);p11=pdc1*(1-pdc2)*pdc3*pdc4;

p12=pdc1*pdc2*(1-pdc3)*(1-pdc4);p13=pdc1*pdc2*(1-pdc3)*pdc4;

p14=pdc1*pdc2*pdc3*(1-pdc4);p15=pdc1*pdc2*pdc3*pdc4;

dckr=p0+p5+p10+p15-p3-p6-p9-p12;dcki=p1+p7+p8+p14-p2-p4-p11-p13;

//printf("\n DC: %15d",dc1);printf(" %15d",dc2);printf(" %15d",dc3);printf(" %15d",dc4);

//printf("\n PDC: %1.5f %1.5f",pdc1,pdc2);printf(" %1.5f %1.5f",pdc3,pdc4);

//printf("\n Surova: Re %1.5f Im %1.5f",korr,kori);

```

//printf("  DC= %ld",dc4);printf("  %ld",dc3);
//printf("  %ld",dc2);printf("  %ld",dc1);
//printf("\n DCKold=   Re %1.5f Im %1.5f",dckorr,dckori);
//printf("\n DCKnew= %1.5f %1.5f",dckr,dcki);
*korr=*korr-dckorr;
*kori=*kori-dckori;
}

//-----

float checkspeed()    //measure speed of computer (sample rate etc)
{
clock_t sta,mid,sto;
float srate,brate,korr,kori;

//printf("\n clktck=%8.2f \n",CLK_TCK);
sta=clock();
vzorcenje(); //do the sampling
mid=clock();
korelator(&korr,&kori);
sto=clock();
printf("\n cas vzorčenja= %d",mid-sta);
srate=131072.0/(mid-sta)*CLK_TCK;
printf("\n sampling rate= %8.0f /sec",srate);
printf("\n cas korelacije = %d",sto-mid);
brate=1.0/(sto-sta)*CLK_TCK;
printf("\n block rate= %3.2f /sec \n",brate);
return brate;
}

//*****

int main()
{
int i,x,y,ys,yi,yis,avg,xkor;
long j;
float korr,kori,kora,kork,mer,kavgr,kavgi,brate;
int gdriver = DETECT, gmode, errorcode;
time_t t;
struct tm *utc;
char filn[30],irqmask;
FILE* ff;

```

```

mkcortab(0,ct0real,ct0imag); //make lookup tables for the correlator
mkdctab(dctab);

printf("\n\n ----- SIDI1 starting ----- \n");

brate=checkspeed();checkquad();getch();

for (j=0;j<10;j++) printf("\n %6X",sig[j]); //print out a few samples

do //allow the user to set the DC offsets
{
vzorcenje();
checkdc();
}
while (kbhit()==0);
getch();

irqmask=inportb(0x21);
outportb(0x21,irqmask+1); //stop timer interrupt

do //main loop: sampling and correlation
{
vzorcenje();
korelator(&korr,&kori);
kora=sqrt(korr*korr+kori*kori);
if (kori==0)
kork=0;
else
kork=atan2(kori,korr)*180/3.14159;
printf("\n Korelacija: Re= %2.5f Im= %2.5f",korr,kori);
printf(" Abs= %2.5f Kot= %2.5f",kora,kork);
}
while (kbhit()==0);
getch();

initgraph(&gdriver, &gmode, "c:\\bc\\bgi");

mer=5.0; //scaling factor
do //circular graph
{
setviewport(50,500,10,460,1);
clearviewport();

```

```
setcolor(GREEN);
circle(275,235,225);
line(50,235,500,235);line(275,10,275,460);
setcolor(WHITE);
printf("Rim=%1.3f%c",1/mer,13);
    for(i=1;i<10;i++)
    {
    vzorcenje();
    korelator(&korr,&kori);
    kora=sqrt(korr*korr+kori*kori);
    x=275+(int)225*kora*cos(kork)*mer;
    y=235-(int)225*kora*sin(kork)*mer;
    line(x-5,y,x+5,y);line(x,y-5,x,y+5);
    }
}
while (kbhit()==0);
getch();

clearviewport();
printf("\n Enter file name for output data: ");
scanf("%30s",filn);fflush(stdin);clrscr();

avg=100;mer=10;xkor=2;                //fringe drawing

t = time(NULL);
utc = gmtime(&t);

ff=fopen(filn,"w");
fprintf(ff," SIDI1 ver. A");
fprintf(ff,"\n File: %s",filn);
fprintf(ff,"\n Source:");
fprintf(ff,"\n Center frequency: 1276.88 MHz");
fprintf(ff,"\n %s",asctime(utc));
fprintf(ff,"\n Averaging: %d",avg);
fprintf(ff,"\n Data rate: %2.2f /sec",brate/avg);
fprintf(ff,"\n\n Correlation values");
fprintf(ff,"\n\n Real   Imag");

setviewport(50,600,100,470,1);
clearviewport();
setcolor(GREEN);
for (x=50;x<601;x=x+55) line(x,100,x,470);
for (y=100;y<471;y=y+37) line(50,y,600,y);
```

```
setcolor(WHITE);

printf("                SIDI1 by s57uuu \n\n");
printf("                Vert. scale= %1.3f/div, ",.2/mer);
printf("Hor. scale= %4.1f sec/div",55*avg/brate);
printf("\n\n                %s", asctime(utc));

x=50;y=285;yi=285;
do
{
kavgr=0.0;kavgi=0.0;
for (i=0;i<avg;i++)
{
vzorcenje();
korelator(&korr,&kori);
kavgr=kavgr+korr;kavgi=kavgi+kori;
}
kavgr=kavgr/avg;kavgi=kavgi/avg;
fprintf(ff,"\n %2.5f %2.5f",kavgr,kavgi);
x=x+xkor;ys=y;yi=yi;
y=285-(int)185*kavgr*mer;
yi=285-(int)185*kavgi*mer;
setcolor(WHITE);line(x-xkor,ys,x,y);
setcolor(RED);line(x-xkor,yis,x,yi);
}
while ((kbhit()==0)&&(x<601));
getch();

outportb(0x21,irqmask); //resume timer interrupt

closegraph();
fclose(ff);

}
```