`ibisc.exe`

J.-M Friedt, August 7, 2003

1 Hardware setup

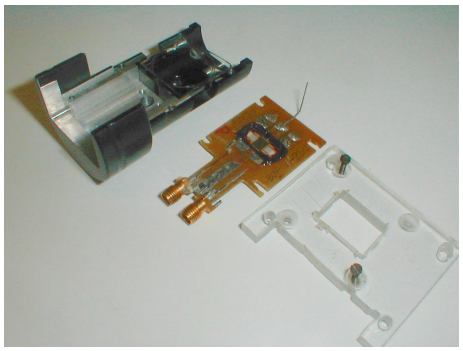
1.1 Preliminary hardware setup

The cooling bath is assumed to have been started at least one hour before starting the experiment for the temperature of the SPR instrument and of the room to stabilize. The solutions to be used are stored in the cooling liquid of the cooling bath and are thus at the right temperature.

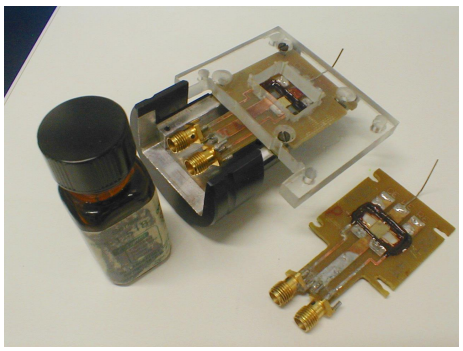
Keep the cover of the Ibis instrument removed: the fan cooling the power supply of the SPR instrument otherwise blows air over the sensing area and leads to unstable signals. However, make sure that the yellow and green grounding cable is connected to the frame of the instrument, otherwise communication problems between the control software and the instrument will occur.

First the SAW mounted on a PCB must be properly fixated to the prism. "Properly" here means without breaking the SAW device, limiting the chances of leak during the experiment, and providing a good optical coupling between the prism and the quartz slide.

1. bend the metallic rigid wire connected to the center electrode upward in order to be able to ground the sensing area once the experiment is started.

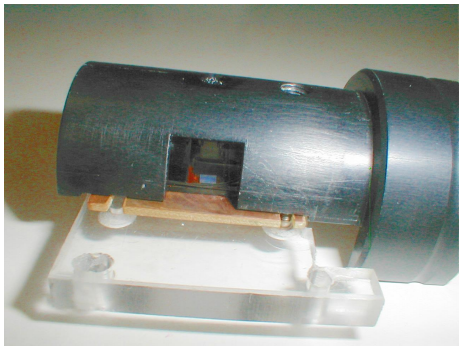


2. put *one drop* of optical index matching oil over the backside of the SAW, on the area opposite to the sensing area. The oil will make the roughened side of the quartz wafer look transparent. Put the PCB on the prism, align the holes in the PCB over the screw holes in the black prism holder, and fasten the two diagonally opposite screws (no need to use all four screws as initially planned).



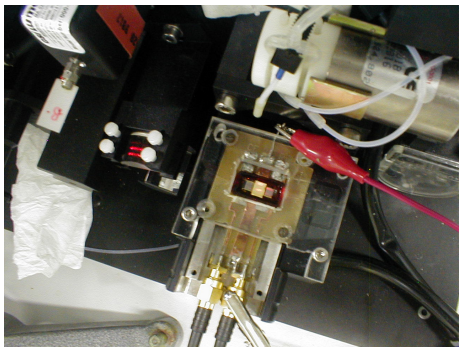
3. make sure that the optical index matching oil is well distributed between the prism and the quartz slide. By looking through the prism, you should be able to see the IDTs and the

sensing area. Any circular mirror-looking areas are bubble in the optical index matching oil: either there is not enough oil, or the screws are not evenly fastened. Restart from step 2.



4. position the prism holder on the SPR instrument and fasten the two silver-colored screws closer to the user with the appropriate Allen wrench.

5. Finally, connect the two SMA cable *and be careful not to twist the coax cable with respect to the connector*. The SMA connector is only attached to the coax cable through the outer shield, and twisting the cable will lead to breaking the ground connexion (and thus high insertion loss and fluctuating signals).



6. connect the red cable with two crocodile clips between the shield of the SMA connector and the metallic pin connected to the sensing electrode.

1.2 Starting the data acquisition

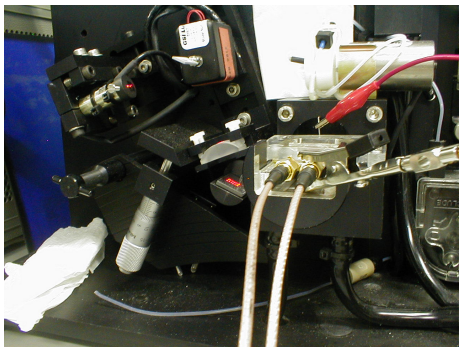
The following sequence of actions was developed for using the instrument. It results from multiple trials for understanding how the Ibis hardware and software work: it might no be the best sequence, but it has allowed me to perform multiple experiments with minimal waste of time in troubleshooting. The order of the actions to be performed is important:

1. Start the commercial Ibis data acquisition software (Start -> Ibis2 -> Data Acquisition)

2. switch on the Ibis instrument. Wait for the red laser to switch on, at which time the software should detect that the hardware had been switched on. You will know the software is initializing the instrument by watching both syringes move. The laser operation is *independent* from the software communicating with the instrument. I have observed that the software will *never* successfully start communicating with the hardware if the laser does not switch on. *Don't* switch on the instrument first and then start the software: communication initialization will not be performed properly.

3. Answer "OK" to the comment "Sampler not responding"

4. In the Ibis software: Tools -> Scope mode
   will display the SPR dip v.s angle. Make sure the dip is visible on both channels, and that the minimum of the dip is the lowest point of the curve. For that to happen, the curve must be shifted to the right ($\simeq$ 1000 m^o) so that the interference fringe does not prevent the data processing software from identifying the position of the peak (only important for real time peak position monitoring – we can eliminate the interference fringe during post-processing). The position of the peak is roughly tuned by turning the micrometric screw on the bottom left of the following photograph:



5. Stop (red square), double click on the curves and Data -> Copy! and save as a text document in Wordpad (Paste). You will need the first column (pixel number to angle conversion) during the post-processing step. I save both curves to remember what they looked like, although only one of them will actually be needed.

6. now that we have checked that the SPR dip is within the measurement range, kill the Ibis software and switch the instrument off. Wait a few seconds and switch the instrument on. Wait for the laser to switch on. The Ibis instrument is again ready for use, but with our custom software this time.

7. Setup the network analyzer for measuring the properties of the SAW device.
   Meas -> B/R
   Display -> Dual Chan ON
   Ch2 -> Format -> PHASE
   Scale Ref -> Auto Scale
   Ch1 -> Scale Ref -> Auto Scale
   Center=123.5
   Span=2 MHz (default point num: 201, keep this val)
   Marker = 123.2 MHz
   Bw/Avg -> Averaging Factor=4, Averaging=ON

8. In a DOS/Command window, run dumpmrker once, which will display an GPIB communication error on the screen of the network analyzer: kill the software by CTRL-C. I cannot identify where this initialization error comes from, but it only occurs the 1st time !

9. start in another DOS/Command window `ibisc`. The lower syringe should move, then the top one. *When the top syringe reaches the top position*, start `dumpmrker 350` in the DOS/Command window which was opened in the previous step. Starting the second program when the top syringe reaches the top position guarantees that the two datasets from the SPR and the SAW measurements are synchronous (within 1 s). The parameter 350 after `dumpmrker` is a delay in ms, and was calculated by considering that sweeping 201 points on a 2 MHz range around 125 MHz takes 70 ms. We take four averages so 4×70=280 ms, and we wait a little bit longer to make sure we have a full new curve.

10. Now that the acquisition is started, we can use Wgnuplot (gnuplot for Windows) to monitor the data while they are recorded. Start Wgnuplot, then `cd 'd:/saw_spr'` to go to the right directory, `plot "phasemrk.asc" using 1:2 with dots` to plot the SAW phase and `replot "min.asc" using 1:2 with points` to plot the position of the dip of SPR channel 1, `replot "min.asc" using 1:4 with points` for channel 2.

11. periodically type `replot` in Wgnuplot to update the graph (under Unix the reload command allows periodic update of the graph plotted by gnuplot, but this function does not seem to work under Windows).

12. Wait 10-15 min to get a proper baseline: the SPR curve should look flat (poor accuracy since we are looking at raw pixel number) and the phase fluctuations of the SAW curve should be less than 0.5 ^{o} (ideally around 0.2 ^{o}).

2 The SPR software: `ibisc`

2.1 Introduction

The Holland Biomaterials Ibis II instrument provides a 670 nm collimated laser source, the mechanical setup for holding the cylindrical glass prism and the sample on which the SPR is to be generated, and a 200-pixel CCD array providing reflected intensity *v.s* angle curves. The laser beam sweeps a $\pm 3^o$ range after being reflected by a rotating mirror.

2.2 Objective

The objective of this software is to provide an opensource program for controlling the Holland Biomaterials Ibis II SPR instrument. The proprietary software provided with the instrument is flawed by the following limitation:
- a maximum of 8000 points can be acquired, limiting a single acquisition run to 2.2 hours.
- automatic peak fitting for extracting the SPR dip position which requires a parabola-like intensity *v.s* angle curve and is thus not suitable for samples in which interference fringes overlap with the SPR signal.

2.3 Getting started

`ibisc` will try to retrieve reflected intensity *v.s* angle curves as quickly as possible, which means obtaining about 1 point every 2 seconds. There is thus no user defined delay. The files names in which the data are stored cannot be modified by the user: they are `min.asc` for a rough estimate of the minimum (raw data: the 1st column is a timestamp in seconds since the beginning of the experiment, then follow two sets of two columns which contain respectively the position of the minimum and the intensity value at this minimum for channel 1 and channel 2) and `log.asc` which contains an array of $n \times 200$ points (ASCII), where n is the number of curves read during the experiment. This last file can become very large for last experiments. Thus, the program is simply started by running `ibisc` in a Command Prompt window under Windows NT a few seconds after switching on the instrument. The data acquisition only starts when the top syringe reaches its top position. **The program assumes that the Ibis II is connected to serial port COM1**. If that is not the case, modify the source code accordingly (see the `idComDev=` line). At that time another data acquisition program can be started, such as `dumpmrker` for recording the SAW device phase (since the latter program only uses the GPIB bus for communication, the data acquisition rate of `ibisc` will not be modified).

2.4 How it works

The Ibis II hosts a full embedded DOS PC which communicates with the control software via RS232. Such a bidirectional communication can easily be monitored and once the command

sequences are recorded, writing a new RS232 communication program for controlling the SPR instrument is a trivial task. The initialization routine has been recorded and duplicated without any attempt at understanding the meaning of the codes. This work was performed by Bart Vanlandeghem during his summer 2002 stay in our group. We then identified the command requesting the instrument to send the full 200 points representing reflected intensity *v.s* angle. These 200 points are recorded, and a rough raw minimum intensity pixel index is saved for low resolution real time monitoring of the evolution of the reaction under investigation.

The program first initializes the instrument by sending a first series of bytes as stored in `initia2a.txt`. We then wait for the Ibis instrument to answer with `I`, after which the second part of the initialization step stored in `initia2b.txt` and `initia2c.txt` are sent. At that time the two syringes start moving.

Once the initialization step is completed, the Ibis is ready for use: we then continuously send the command which requests the full 200 pixels to be sent back (similar to the 'plot raw curve' function in the original Ibis software). The 2×200 pixel intensities, in the 0 to 65535 range, is sent back as series of 4 bytes in ASCII hexadecimal format, MSB first. The full curve is recorded in the file `log.asc` while the index of the pixel of minimum intensity as well as the pixel value of both channels are recorded in `min.asc`, whose first column also holds a timestamp (time in seconds since the beginning of the experiment). The content of `min.asc` can be plotted during the experiment using `gnuplot` for example for real time monitoring of the evolution of the SPR dip position while the reaction is occurring.

Once the experiment is completed, the raw curves can be accurately fitted with any appropriate function and the dip position recorded with high accuracy using the following Matlab script:

```
% marker at 123.2 MHz, device 38/8
% span=2 MHz, 201 points, power=0 dBm

% load log.asc
load ch1salt.txt
x=ch1salt(:,1);
xx=interp(x,100);
ally=log;
cpt=1
for k=1:2:length(ally)
 y=ally(k,:)';
 yy=interp(y,100);
 [a,b]=polyfit(x,y,5);
 yy=polyval(a,xx);
 [m,n]=min(yy(3000:12000));
 minim1(cpt)=xx(n+3001);
 cpt=cpt+1
end

cpt=1
for k=2:2:length(ally)
 y=ally(k,:)';
 yy=interp(y,100);
 [a,b]=polyfit(x,y,5);
 yy=polyval(a,xx);
 [m,n]=min(yy(3000:12000));
 minim2(cpt)=xx(n+3001);
 cpt=cpt+1
end

load min.asc
```

```
f=[minim1' minim2']
plot(min(:,1),f(:,1),'.');hold on; plot(min(:,1),f(:,2),'r.')
save -ascii fit.asc f
```

This script assumes that the file `log.asc` was loaded in memory (`load log.asc`) and that the variable `min` was cleared (in order to use the `min()` function of Matlab rather than the variable with the same name). We also assume that a variable named `cuch1` was loaded in memory: this variable contains as a first column the pixel index to angle relationship, as provided when saving a raw curve under the Ibis software (request a raw curve, double click on the curve, Data, Copy and save the resulting values in Wordpad in a file named cuch1.txt. Then `load cuch1.txt` under Matlab). Each curve stored in the file `log.asc` is fitted with an 11th degree polynomial (the degree was chosen randomly and appeared to provide good results, but might not be an optimum value in terms of accuracy and speed), the X-range resolution increased 100-fold (variable `x`) and the position of the minimum of the polynomial identified with such an accuracy. The resulting dip position and value are stored in the variables `mm_pos` and `mm_val` respectively. Since channels 1 and 2 are interlaced in the log file `log.asc`, having the counter `k` start at 1 or 2 will define whether the fit is done on the curves of channel 1 or 2. This script must thus be run twice, once with `k` starting at 1 and once with `k` starting at 2 in order to obtain all relevant dip positions for both channels. Take care after the first run to save the variables `mm_pos` and `mm_val` in variables with different names.

3 Data processing

`ibisc` generates two files: `log.asc` and `min.asc`. The latter contains the time at which the point was measured (with respect to the start of the experiment), the position of the first dip (in pixel), the height of the first dip, the position of the dip of channel 2 and its height. The former contains all spectra (reflected intensity v.s angle) and can be quite large for longer experiments.

We have seen in the previous section how to convert `log.asc` to an accurate identification of the position of the dip of the SPR, and stored the result in a file called `fit.asc`.

The SAW data are recorded in `phasemrk.asc` and contains in the first column the date with respect to the beginning of the experiment, and the phase at the position of the marker. The graph summarizing all results is thus generated with:

```
load min.asc
load fit.asc
load phasemrk.asc
subplot(211)
plot(min(:,1),fit(:,1),'.');hold on;plot(min(:,1),fit(:,2),'r.');
subplot(212)
plot(phasemrk(:,1),phasemrk(:,2),'.')
```

One can eventually convert the data contained in `temp.asc` in hexadecimal format (4 bytes) to decimal and plot the evolution of the temperature during the experiment, but the conversion factor to kelvin is not known (so actually only the relative variation of the temperature can be extracted from these data). The program `tconvert.c` load `temp.asc` and displays the decimal values on the standard output:

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

void main()
{FILE *f;
 unsigned int a,b;
 int res;
```

```
 f=fopen("temp.asc","r");
 do {
   res=fscanf(f,"%d %x",&a,&b);
   printf("%u %u \n",a,b);
 } while (res!=EOF);
   fflush(stdout);
}
```

4 Future work

`ibisc.exe` was written in ANSI-C under Windows NT4, but being a console (text mode) application communicating by RS232, it should be easily ported to any other operating system supporting RS232 communication. While DOS will not enable simultaneous monitoring of the dip position (due to lack of multitasking), porting to linux should be a trivial task. An example of non-blocking RS232 port initialization can be found for example in the H8 programming software at `http://friedtj.free.fr/h8.tgz` (accompanying document at `http://friedtj.free.fr/h8eng.pdf`).

The full source code can be found in `\\nt4\friedtj\visualc\ibisc` and the binary version can be found on the computer between the network analyzer and the Ibis instrument in `D:\saw_spr`.

```
// IBIS control software for WinNT (04/09/02)
// modified for recording whole SPR curves 03/03
#include "stdafx.h"
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>

#include <time.h>

#undef debug

HANDLE  idComDev ;
DWORD dwBytesRecd,dwBytesWritten;

void send_cmd(char *s)
{int n=0;
 char aa[5000];                    // max command length !
 while (s[n]!=0)
   {
#ifdef debug
 if ((s[n]==13)||(s[n]==10)) printf(" "); else printf("%c",s[n]);
#endif
aa[n]=s[n];
    if (s[n]=='_') aa[n]=13;
    if (s[n]==10) aa[n]=13; // unix/DOS files: chr10 for end of line
if (s[n]==13) n--;      // if it's a DOS file, rm chr(13)
    n++;
   }
 aa[n]=13;                   // trailing character
 WriteFile(idComDev,aa,n,&dwBytesWritten,0);fflush(stdout);
#ifdef debug
 printf("\n\n");
#endif
```

```
}

void send_file(char *s)
{int n=0,res=EOF+1;FILE *f;
 char aa[5000],cc;                    // max command length !
 f=fopen(s,"r");
 while (res!=EOF)
 //  {res=fscanf(f,"%d",&c);aa[n]=c;n++;
 //   if (aa[n-1]==13) printf("\n"); else printf("%c",(char)c&0xff);
   {res=fscanf(f,"%c",&cc);
    if ((cc==13)||(cc==10)) printf(" "); else printf("%c",cc);
aa[n]=cc;
    if (cc==32) aa[n]=13;
    if (cc==10) aa[n]=13; // unix/DOS files: chr10 for end of line
if (cc==13) n--;      // if it's a DOS file, rm chr(13)
    n++;
   }
 WriteFile(idComDev,aa,n-1,&dwBytesWritten,0);fflush(stdout);
 printf("\n\n");fclose(f);
}

int main(int argc, char* argv[])
{DCB dcb;COMMTIMEOUTS cmm;
 BYTE abIn[2]; int cnt;unsigned int res1[200],res2[200],min,minp;
 FILE *flog,*fmin;
 char c1,c2,c3,c4;
 time_t clock1,clock2;

 printf("CHECK THAT IBIS IS CONNECTED TO COM2\n");
 idComDev=CreateFile("COM2", GENERIC_READ | GENERIC_WRITE,1,NULL,
            OPEN_EXISTING,0,NULL ); // 0=NO overlapped IO
 dcb.DCBlength = sizeof( DCB ) ;
 GetCommState(idComDev, &dcb ) ;
 dcb.BaudRate=9600 ;
 dcb.ByteSize=8;
 dcb.Parity=NOPARITY;
 dcb.StopBits=ONESTOPBIT;
 SetCommState( idComDev, &dcb ) ;

 GetCommTimeouts(idComDev, &cmm ) ;
 cmm.ReadIntervalTimeout=100;
 cmm.ReadTotalTimeoutConstant=100;
 cmm.ReadTotalTimeoutMultiplier=100;
 cmm.WriteTotalTimeoutConstant=100;
 cmm.WriteTotalTimeoutMultiplier=100;
 SetCommTimeouts(idComDev, &cmm) ;

/*
 ReadFile(idComDev,abIn,1,&dwBytesRecd,0);    // blocking read ... until timeout
 if (dwBytesRecd==1) printf("%d=%c \n",(int)abIn[0],(char)abIn[0]);
    else printf("Timeout ... \n");           // dwBytesRecd=0 if timeout else =1
*/

 do ReadFile(idComDev,abIn,1,&dwBytesRecd,0); // empty UART buffer
```

```
 while (dwBytesRecd!=0);
 // send_file("initia1.asc");

 /*****************************************************************/
 // INITIALIZATION STEPS
 send_file("initia2a.txt");

 printf("\n Here the instrument must answer 'I': ");
 ReadFile(idComDev,abIn,1,&dwBytesRecd,0);printf("%c",(char)abIn[0]); // read 'I'
 printf("\n");
 // send_file("initia2.asc");
 send_file("initia2b.txt");

 printf("Watch the seringes move ...\n");
 do ReadFile(idComDev,abIn,1,&dwBytesRecd,0); // read 'i'+'Y...'
 while((char)abIn[0]!='Y'); printf("%c",(char)abIn[0]);
 for (cnt=0;cnt<6;cnt++) // we get the 6 chars after the 'Y' & print them
{ReadFile(idComDev,abIn,1,&dwBytesRecd,0);printf("%c",(char)abIn[0]);}
 printf("\n");
 // send_file("initia3.asc");
 send_file("initia2c.txt");
 /*****************************************************************/

  flog=fopen("log.asc","w");fmin=fopen("min.asc","w");
 /*****************************************************************/
 // SEND COMMAND REQUESTING CURVE (G3) AND STORE / PROCESS
 time(&clock1);
 send_cmd("O_2_G_6_");
 printf("\n Scope mode commandb (O2G6) sent -- 20/05/03\n");
 while (1) {
//  do {ReadFile(idComDev,abIn,1,&dwBytesRecd,0); // read 'TA????'
//#ifdef debug
//      printf("%c",abIn[0]);
//#endif
// }                  // empty buffer
//   while (dwBytesRecd!=0);
#ifdef debug
   printf("\n");
#endif
//   send_cmd("G_3_");
   do ReadFile(idComDev,abIn,1,&dwBytesRecd,0); // read until beginning of curve
while ((char)abIn[0]!='S'); // && ((char)abIn[0]!='T'));
   printf("reading curve ... ");fflush(stdout);
   time(&clock2);
   // if ((char)abIn[0]=='T') // temperature is NOT sent with O2G6 command
   // {printf("\nT=");
   // for (cnt=0;cnt<4;cnt++)
   //  {ReadFile(idComDev,abIn,1,&dwBytesRecd,0);c1=(char)abIn[0];printf("%c",c1);}
   // }
   // else { // 'S' => we got a curve
   for (cnt=0;cnt<200;cnt++)
{ReadFile(idComDev,abIn,1,&dwBytesRecd,0);c1=(char)abIn[0];
ReadFile(idComDev,abIn,1,&dwBytesRecd,0);c2=(char)abIn[0];
ReadFile(idComDev,abIn,1,&dwBytesRecd,0);c3=(char)abIn[0];
```

```
ReadFile(idComDev,abIn,1,&dwBytesRecd,0);c4=(char)abIn[0];
if (c1>64) c1-=55;else c1-=48;if (c2>64) c2-=55;else c2-=48;
if (c3>64) c3-=55;else c3-=48;if (c4>64) c4-=55;else c4-=48;
res1[cnt]=(((c1*16)+c2)*16+c3)*16+c4;
}
printf("c1 read ");fflush(stdout);
ReadFile(idComDev,abIn,1,&dwBytesRecd,0); // remove 'V'
for (cnt=0;cnt<200;cnt++)
{ReadFile(idComDev,abIn,1,&dwBytesRecd,0);c1=(char)abIn[0];
 ReadFile(idComDev,abIn,1,&dwBytesRecd,0);c2=(char)abIn[0];
 ReadFile(idComDev,abIn,1,&dwBytesRecd,0);c3=(char)abIn[0];
 ReadFile(idComDev,abIn,1,&dwBytesRecd,0);c4=(char)abIn[0];
 if (c1>64) c1-=55;else c1-=48;if (c2>64) c2-=55;else c2-=48;
 if (c3>64) c3-=55;else c3-=48;if (c4>64) c4-=55;else c4-=48;
res2[cnt]=(((c1*16)+c2)*16+c3)*16+c4;
}
min=65535;minp=0;
for (cnt=0;cnt<200;cnt++)
{fprintf(flog,"%u ",res1[cnt]);if (res1[cnt]<min) {min=res1[cnt];minp=cnt;}}
fprintf(flog,"\n");printf("min1=[%u,%u] ",minp,min);
fprintf(fmin,"%ld %u %u ",clock2-clock1,minp,min);
min=65535;minp=0;
for (cnt=0;cnt<200;cnt++)
{fprintf(flog,"%u ",res2[cnt]);if (res2[cnt]<min) {min=res2[cnt];minp=cnt;}}
fprintf(flog,"\n");printf("min2=[%u,%u] ",minp,min);
fprintf(fmin,"%u %u\n",minp,min);
fflush(flog); fflush(fmin); printf("DONE\n");
}
// }
/*
 while (1)
 {printf ("> ");scanf("%s",cmd);send_cmd(cmd);
  fprintf(log,"CMD: %s\n",cmd); // cmd is separated and terminated by '_'
  fflush(log);
  do {ReadFile(idComDev,abIn,1,&dwBytesRecd,0); // empty UART buffer
      printf("%c",abIn[0]);fprintf(log,"%c",abIn[0]);fflush(log);}
  while (dwBytesRecd!=0);
  printf("\n");
 }
 */
 return(0);
}
// if (argc==1) f=fopen("connect.txt","r"); else f=fopen(argv[1],"r");
```