HELP FOR THE NEW SPECTROMETER UNDER DELPHI 4.0 (2001)

Julien Bobroff, Safia Ouazi, Véronique Brouet
LPS, Bat.510, Université Paris XI, 91405 Orsay Cedex, France
bobroff@lps.u-psud.fr

How to start the program

start Delphi Borland 4.0 (the yellow greek temple icon). Then press Ctrl+F11 and choose in the directory c:\newspectro\pilotageManip3 the project named PilotageManip. Then press F9 (being sure that the spectrometer is open).

How to work with the different windows :

All the acquisition is done through the **acquisition RMN** window. The measured data (sum of all the shots) is in ECHO which appears in A/B of this window. The last single shot data is in C/D in OSCILLO. One uses the Edit box of this window (in A/B) to save the data or recall another one (in C/D) for comparison purpose.
There is an automatic soustraction of background (taken between 75% and 100% of your data) on ECHO at the end of the run or at each rafraichissement. (no soustraction like that on OSCILLO).

All the Fourier Transforms are done in **TFRMN** window. Everything works as in the acquisition window, with the TF parameters appearing in front pannel. The TF can be done on data acquired in the acquisition window, or on previously saved data.

There is an **interpreteur** window aimed at more sophisticated commands or list of commands. It can be used for automatic runs, or for automatic analysis of the data. The syntax to be used is not easy, so it is easier to use the preexisting examples.

There is a **reception** window where to find and change the spectrometer reception amplifier parameters.

There is a **sequence** window which contains the pulse sequence. Do not be confused with the interpreteur window : the sequence window is specifically dealing with the radiofrequency pulses sent to the sample. It has a list of parameters on its left, and a box where to write the sequence on its right. The sequence uses the parameters of left part.

There is a **gestgage** window wich contains the acquisition card (GAGE card) parameters, like digitilization, voltage scale, number of points, etc.

Where is the data saved and how does it look like ?

All data are saved in the current directory, which appears in the acquisition RMN window. Typical saved data have the form : blablaECHO1.spe, blablaECHO2.spe, blablaTF3.spe, blablaECHO4.spe, etc… The number of the last saved data appears near the directory name. You can also save data in specific names as bloubli.spe. These .spe files are the program format files and can be read only by this program. You can also save your data in additional files in ascii format to export them elsewhere (in .dat extension).

An important constraint : at beginning of the program, this directory is found by the software in an ascii file named current.txt located in c:\newspectro\data\ containing just the full directory name (like c:\newspectro\data\blabla). The software also needs a file named

info.txt located in the current directory (here c:\newspectro\data\blabla) which contains in the first line again the directory name (here c:\newspectro\data\blabla) and in the second line the number of the last saved file (here, it would be 4). If one of these files does not exist, you have to create it.

What to do if a bug occurs and stops the program ?
Try to switch to the Delphi window (the yellow greek temple icon) with Alt+Tab. Then, press the F9 key to make it run again. It may bug again, say always OK and F9 again. Switch back to the program window with Alt+Tab when it works. If it always fails and does not start again, then press Ctrl+F2 to stop the program. Start it again with F9.

Some bugs you may (will) meet :

When starting, the program does not find info.txt or current.txt files or c:\newspectro\Parametres RMN.txt (see paragraph Where is the data saved) : it will crush.

The rf pulses or frequencies don't seem to work : try to change the value of the working frequency in the reception window (at 14000000) to 14000001 and put it back to 14000000.

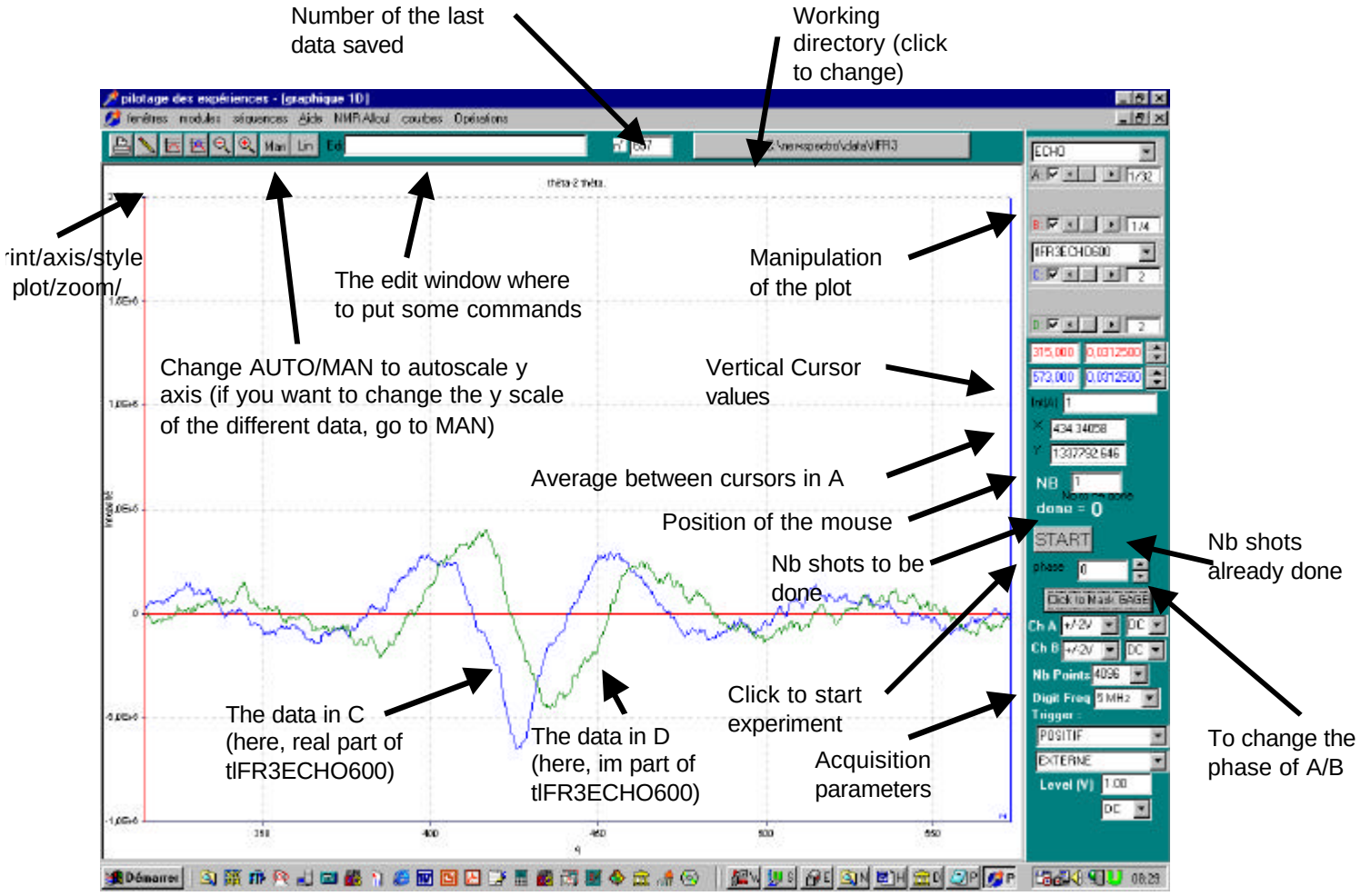
When starting, there is a division by zero mistake : no idea where it comes from, doesn't matter.

Problems when trying to call back a spectrum : check in the current directory if it really exists

Problems with the interpreteur commands : did you really use brackets, "" for characters, and ';' at the end of the line (like command["aa",1,2];) ?

Problems with the sequence window : did you really put a dot '.' at the end of the program ? Do you really put time in msec ?

The acquisition RMN window



in the Edit box, you can type the following commands :

s : **standard saving of the data.** It saves the data currently in A/B channel in the program internal format (which is then readable again by this program but no other). Each data is saved at c:\NewSpectro\data\example\exampleECHO4.spe if “example” is the current directory, and if already 3 files were saved before (either TF or ECHO). In the box of data A, only exampleECHO4 will appear.

sblabla : saving with a specific name (here blabla). It saves the data in A/B at c:\NewSpectro\data\example\blabla.spe (instead of the preformated name exampleECHO4)

asci : saving of the data in ascii format. It saves A/B in an ASCII format readable with other programs like Exel or Origin. The data consists of 3 columns : x, y of channel A, y of channel B. If the present name in the box appears for exemple as exampleECHO4, it will save it as c:\NewSpectro\data\example\exampleECHO4.dat
Caution : the ascii data cannot be recalled then in this program and is usually bigger in size than the .spe files.

asciblabla : saving of the data in ascii format with a specific name (here blabla). It acts like sasc but saves it as c:\NewSpectro\data\example\blabla.dat

r4 : recalls data (here n°4) which was saved previously with s and puts it in C and D.

rblabla : recalls data with specific name (here blabla) which was saved previously with s and puts it in C and D.

add5,7,8 or add5-8 : adds the files 5,7 and 8 or 5 to 8 in SOMME

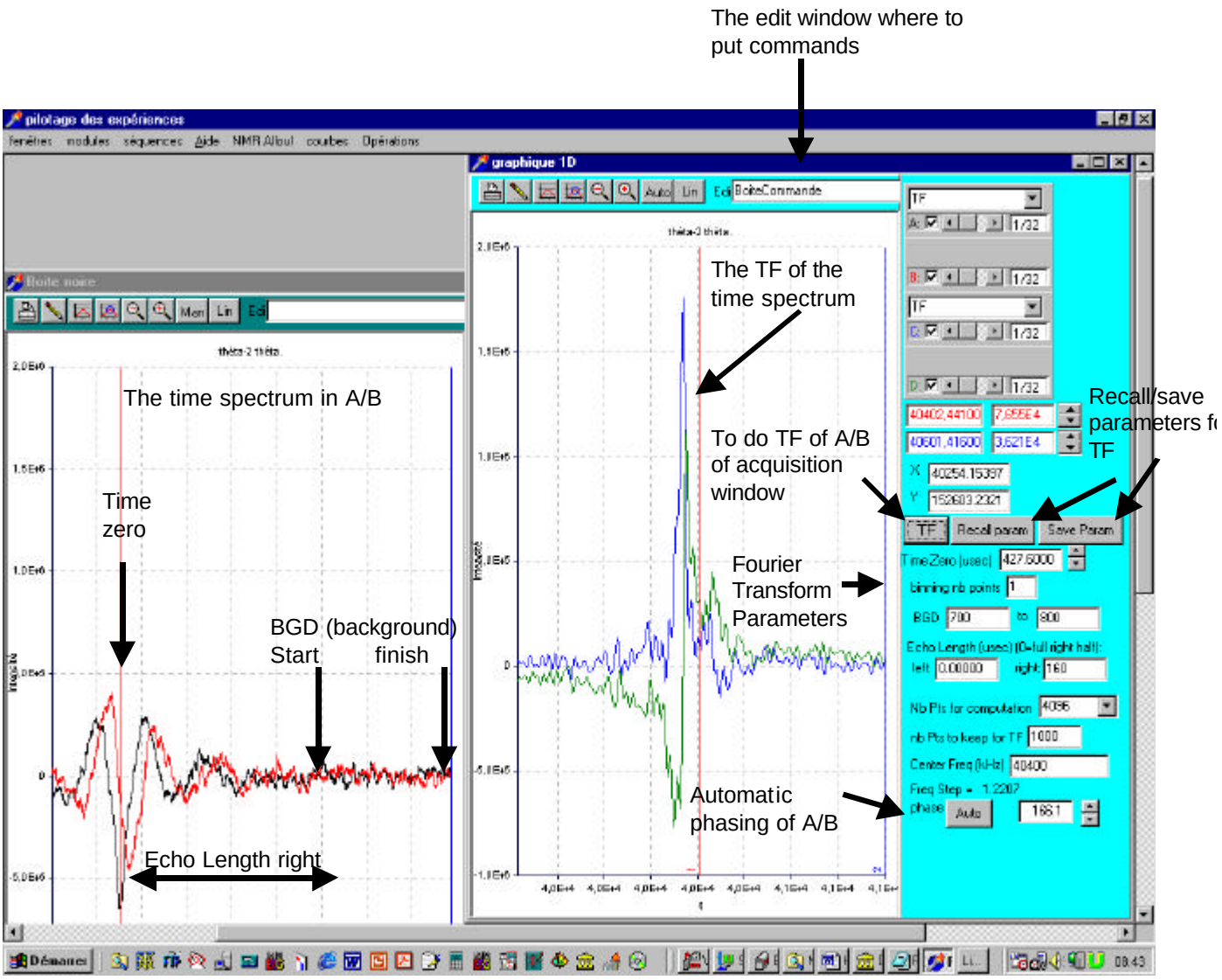
bg or bg3 or bgnom : substracts background on A/B or 3 or nom (taken between 3/4 and end of the data)

int(A,5,24.3) : integral of channel A between 5 and 24.3
intn(A,5,24.3) : integral of channel A between 5 and 24.3 normalized = average value per channel

intecho(A,5,24.3,50,100.3) : echo intensity = square root of [square of (integral of channel A after substraction of bgd taken between 50 and 100.3) + square of (idem for B)] **.** Note that you don't need to take the same interval width for background and echo max. If you put C, it will take C and D. If the two background values are equal, it does not perform the bgd soustraction.

intechon(A,5,24.3,50,100.3) : echo intensity normalized = average value of the echo per channel

The TFRMN window



in the Edit box, you can type the following commands :

Like in Acquisition RMN window, you can use **s**, **sasc**, **r**. Here, the data will be saved as exampleTF4.spe instead of exampleECHO4.spe.

tf : does the Fourier Transform of A/B of Acquisition RMN window. It uses the parameters of Fourier Transform displayed in the window and puts the result in A/B with name TF.

tfblabla : does the Fourier Transform of the file blabla.spe.

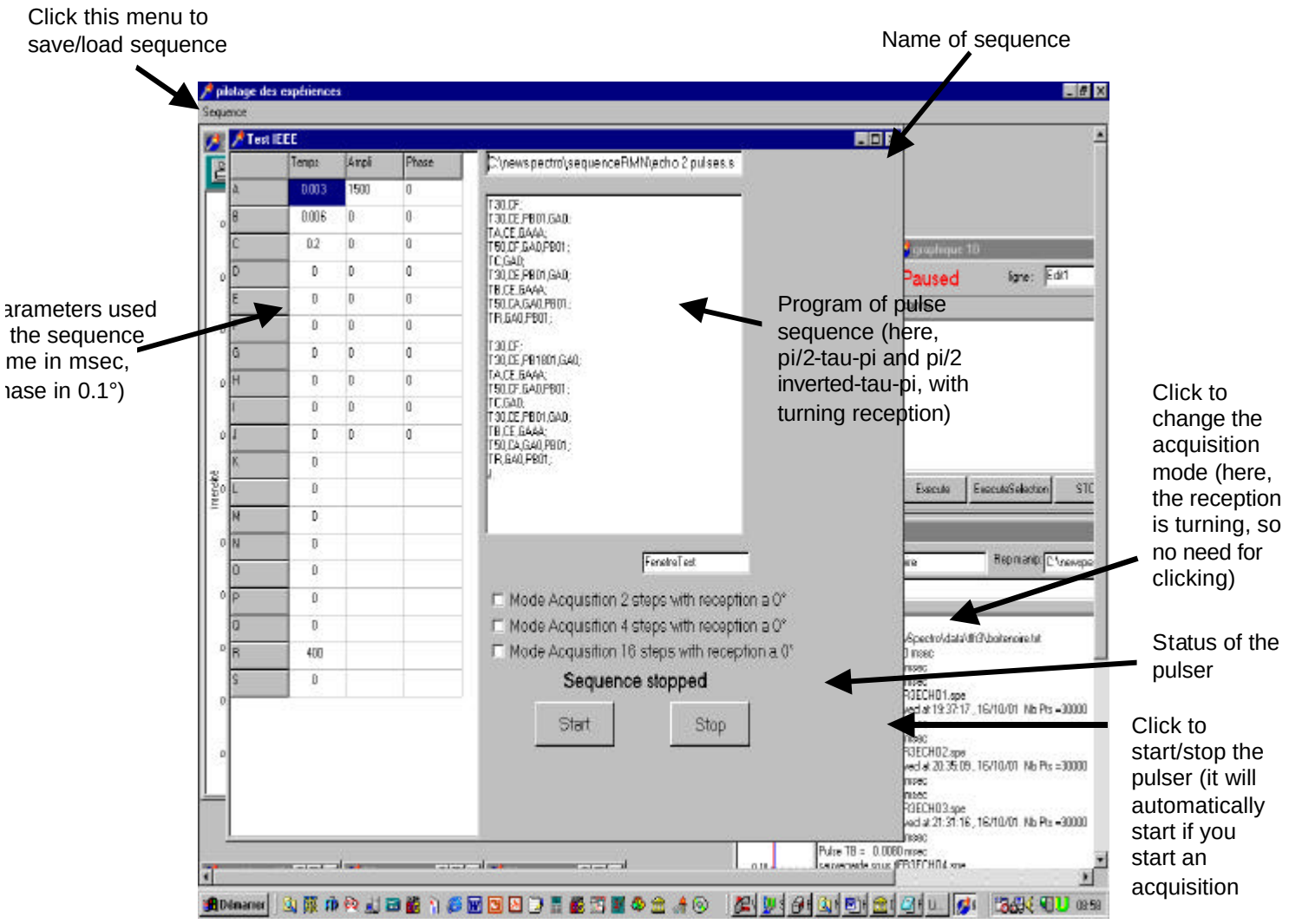
tf5 : does the Fourier Transform of the file with number 5 (in our example a file of the form c:\NewSpectro\data\example\exampleECHO5.dat)

tf5-9 : does the Fourier Transform of the file with number 5 to number 9 and saves them each time (with same parameters for all TF)

The Fourier Transform Parameters : they can be saved and recalled by the button. They consist of ascii file with name like blabla.par and with explicit form (you can read them with word for exemple).
Time Zero : the origin of time of the signal in microseconds
Binning : it averages the time data by replacing a group of binning points by its average. Put 1 doesn't do any binning. You can then put 2, 3, etc… to see the effect. It smoothes the time signal. If you bin too much, you loose some informations contained in the signal.
BGD : the interval (in microseconds) where to take the background which will be substracted
Echo Length : the length of the echo in microseconds to be used (if zero, it uses all the data)
Nb Points for computation : you can use more than the time signal you have to do a more smooth TF by adding some fake points with zero intensity after your signal (Filling Zero). You do so by choosing in Nb of Points for computation a higher number of points than the ones you really recorded. The maximum is a priori the better (16384).
Nb Points to keep for TF : usually, you are interested only in the center of the TF, not in all the points (like the 16384). You can then restrict your TF to this number of points (the program just destroys the other points), it creates a much lighter file easier to work with. Use typically 1000.
Frequency Step : the frequency step between two TF points. It is equal to 1/(Nb of Points for computation*Time step*binning) and is given in kHz.

Example of a time signal and the parameters to be chosen, and the corresponding TF :

The Pulse Sequence window



you can recall an old sequence. But take care : if you modify it and then start the sequence, it saves the new version. So keep somewhere a backup with another name of your standard sequences. You can enter by hand the parameters on left side. It consists of 3 columns :

Phase variation : if standard echo with all phases at 0° gives A,B, then when varying the phases of the different pulses, here is the result :

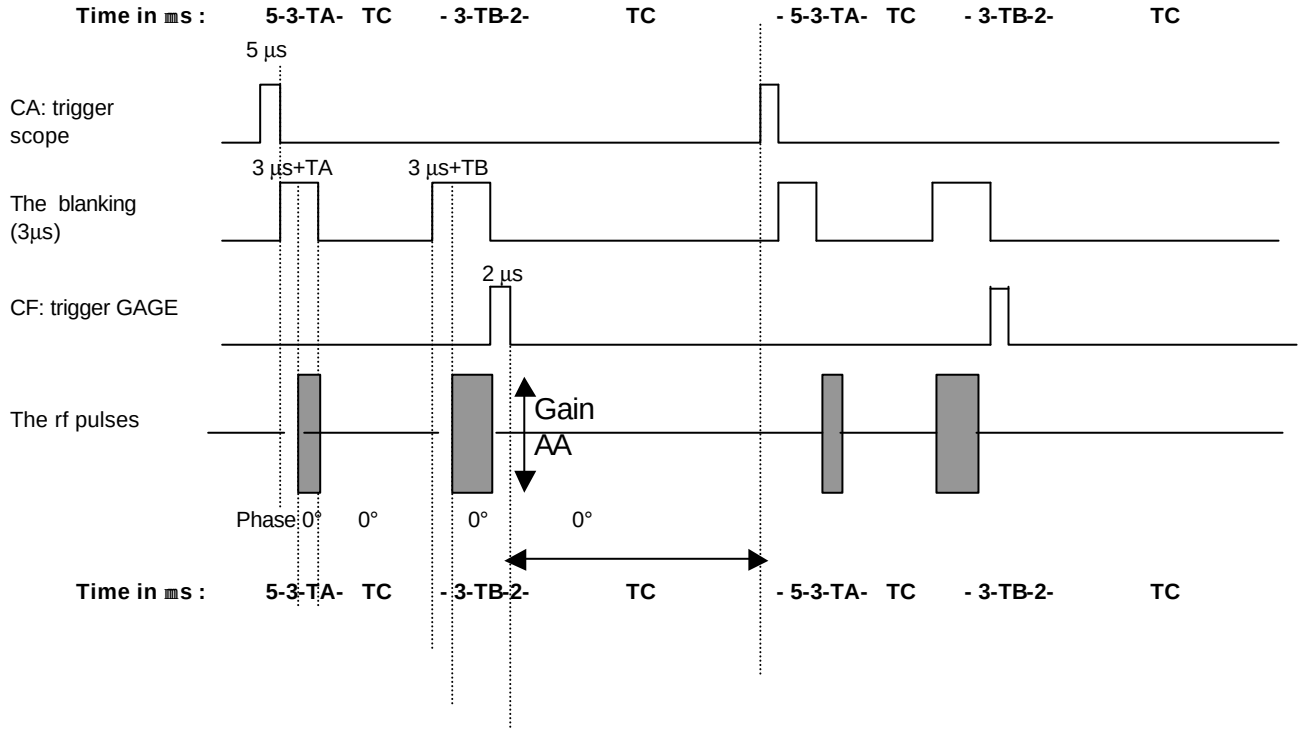
Phase pi/2	Phase Pi	Phase Reception	Channel 1	Channel 2
0°	0°	0°	+A	+B
180°	0°	0°	-A	-B
0°	0°	90°	+B	-A
0°	0°	180°	-A	-B
0°	0°	270°	-B	+A

A simple pi/2 - tau - pi :

pi/2 length = TA with gain AA, 0°
Tau = TC + 3 µs (due to blanking)
Pi length = TB with gain AA, 0°
Rep time = TR (in fact, = TR + 3 + TA + TC + TB + TC + 2)
Blanking on Channel Out E
Trigger for Gage on Channel F
Trigger for scope on Channel Out A
Pulses on channel Out Pulse

T50,CA; starts a trigger (for scope use) on CA during 5 µs
T30,CE,PB01,GA0; starts a blanking pulse on CE during 3 µs
TA,CE,GAAA; starts an rf pulse with amplitude AA during time TA (and keeps on pulse CE) **TC,GA0;** waits during TC with no pulse
T30,CE,PB01,GA0; starts a blanking pulse on CE during 3 µs
TB,CE,GAAA; starts an rf pulse with amplitude AA during time TB (and keeps on pulse CE)
T20,CF,GA0; gives a 2 µs trigger on channel F
TC,GA0,PB01; waits during TC with no pulse
TR,PB01,GA0; waits during TR with no pulse
J. end of sequence

hereis the corresponding histogram:



A pi/2 - tau - pi alternated with pi/2 inverted- tau - pi :
Two ways : either make reception turn as well in phase to have the recombination of pulses, or make recombination in the program.
Advantage and inconvinients : if recombination by turning reception, allows a fast repetition but does not cancel well the saturation after pi pulse (because this saturation has not been turned in phase). If recombination by soft, obliges a slow repetition not to miss any shot, but cancels well the saturation.

To activate the program mode recombination (if your reception phase is left 0°), you have to click the mode in the sequence window. It will do it automatically if in the name of the sequence, you put somewhere 'mode 2' or 'mode 4'.

For example, a sequence named 'piSur2 pi mode 2.seq' when loaded will automatically activate the mode 2 in the window.

If you forget to activate the mode when needed, your resulting data will always be zero !

2 Steps Method with recombination by turning reception :

```
T30,CF;
T30,CE,PB01,GA0;
TA,CE,GAAA;
T50,CF,GA0,PB01;
TC,GA0;
T30,CE,PB01,GA0;
TB,CE,GAAA;
T50,CA,GA0,PB01;
TR,GA0,PB01;

T30,CF;
T30,CE,PB1801,GA0;  here pulse pi/2 is inverted
TA,CE,GAAA;
T50,CF,GA0,PB01;
TC,GA0;
T30,CE,PB01,GA0;
TB,CE,GAAA;
T50,CA,GA0,PB1801;   here reception is turned
TR,GA0,PB1801;
J.
```

2 Steps Method with recombination with reception always at 0°:
One must click then on Mode Acquisition 0° 2 pulses option and choose a slow repetition time

```
T30,CF;
T30,CE,PB01,GA0;
TA,CE,GAAA;
T50,CF,GA0,PB01;
TC,GA0;
T30,CE,PB01,GA0;
TB,CE,GAAA;
T50,CA,GA0,PB01;
TR,GA0,PB01;

T30,CF;
T30,CE,PB1801,GA0; pi/2 inverted
TA,CE,GAAA;
T50,CF,GA0,PB01;
TC,GA0;
T30,CE,PB01,GA0;
TB,CE,GAAA;
T50,CA,GA0,PB01;  reception is not turned
TR,GA0,PB01;
J.
```

4 Steps Method with recombination by turning reception :

```
T30,CF;
T30,CE,PB01,GA0;
TA,CE,GAAA;
T50,CF,GA0,PB01;
```

TC,GA0;
T30,CE,PB01,GA0;
TB,CE,GAAA;
T50,CA,GA0;
TR,GA0,PB01;

T30,CE,PB901,GA0;
TA,CE,GAAA;
T50,GA0,PB01;
TC,GA0;
T30,CE,PB01,GA0;
TB,CE,GAAA;
T50,CA,GA0,PB2701;
TR,GA0;

T30,CE,PB1801,GA0;
TA,CE,GAAA;
T50,GA0,PB01;
TC,GA0;
T30,CE,PB01,GA0;
TB,CE,GAAA;
T50,CA,GA0,PB1801;
TR,GA0;

T30,CE,PB2701,GA0;
TA,CE,GAAA;
T50,GA0,PB01;
TC,GA0;
T30,CE,PB901,GA0;
TB,CE,GAAA;
T50,CA,GA0,PB901;
TR,GA0;
J.

4 Steps Method with recombination with reception always at 0°: One must click then on Mode Acquisition 0° 4 pulses option and choose a slow repetition time

T30,CF;
T30,CE,PB01,GA0;
TA,CE,GAAA;
T50,CF,GA0,PB01;
TC,GA0;
T30,CE,PB01,GA0;
TB,CE,GAAA;
T50,CA,GA0;
TR,GA0,PB01;

T30,CE,PB901,GA0;
TA,CE,GAAA;
T50,GA0,PB01;
TC,GA0;
T30,CE,PB01,GA0;
TB,CE,GAAA;
T50,CA,GA0,PB01;
TR,GA0;

T30,CE,PB1801,GA0;
TA,CE,GAAA;
T50,GA0,PB01;
TC,GA0;
T30,CE,PB01,GA0;

```
TB,CE,GAAA;
T50,CA,GA0,PB01;
TR,GA0;

T30,CE,PB2701,GA0;
TA,CE,GAAA;
T50,GA0,PB01;
TC,GA0;
T30,CE,PB01,GA0;
TB,CE,GAAA;
T50,CA,GA0,PB01;
TR,GA0;
J.
```

Structure of the file c:\newspectro\Parametres RMN.txt

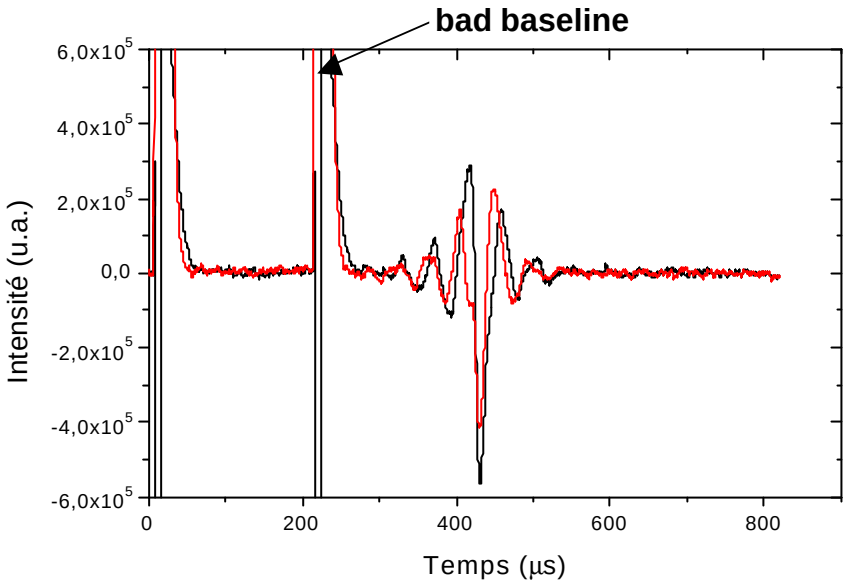
```
Parametres RMN :

Reception Amplifiers
GainHF1
140
GainHF2
140
GainLo0
130
GainLo90
130
GainBF0
200
GainBF90
200
Phase F+Lo1
200
Phase F+Lo2
200
Frequence Lo = 15*F
14000000
**************************
Acquisition GAGE Parameters
Chanel A (+/-10V +/-5V +/-2V +/-1V +/-0.5V  +/-0.2V  +/-0.1V)
+/-1V
Chanel B (Volts)
+/-1V
Chanel A (Mode AC ou DC)
DC
Chanel B (Mode AC ou DC)
DC
Nb Points (16 32 64 128 256 512 1024 2048 4096 8192)
2048
Digit Freq (10 MHz 5 MHz 4 MHz 2 MHz 1 MHz 500 KHz 200 KHz 100 KHz 50 KHz  20 KHz  10 KHz 5 KHz
2 KHz  1 KHz EXTERNE)
2 MHz
Trigger (POSITIF/NEGATIF, VOIE A/VOIE B/VOIES A et B/EXTERNE/SOFT, level in V, DC/AC)
POSITIF
EXTERNE
1.00
DC
******************************
Last Pulse Sequence (exple : nom.seq (found in sequenceRMN) )
C:\NewSpectro\sequenceRMN\echo 2 pulses mode vary.seq
```

2 Steps Method

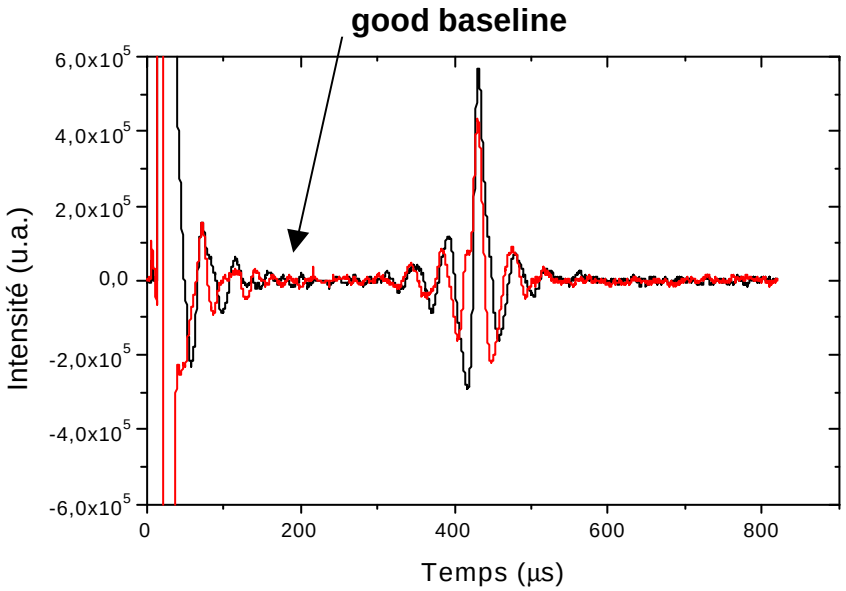
Comparison of the two possible recombination methods: by turning reception or by the soft

"echo 2 pulses mode vary"

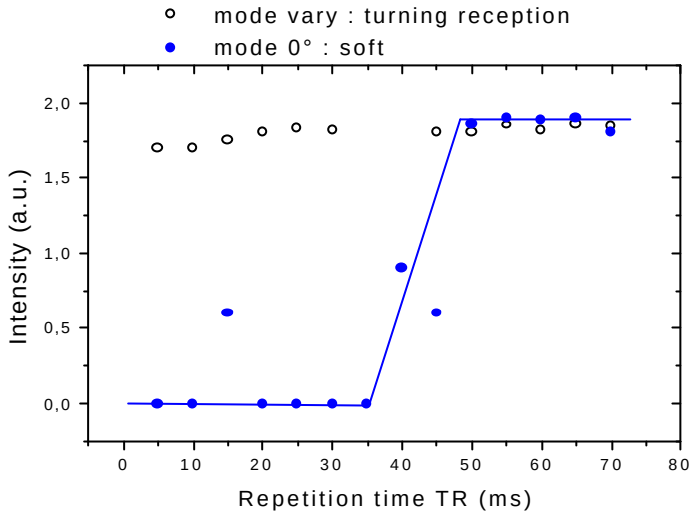


advantage: you can choose a fast repetition time
inconvenient: the bad baseline

"echo 2 pulses mode 0"



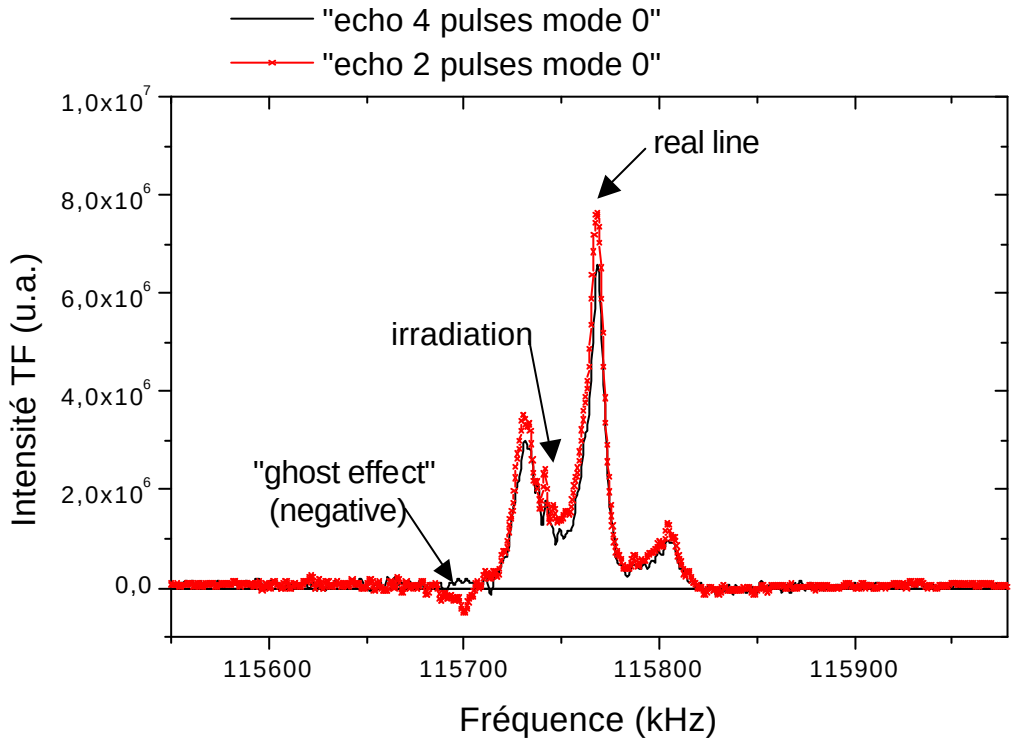
advantage: the good baseline
inconvenients: 1) you can't choose a repetition time faster than 50 ms (see next figure).
2) the sign of the echo is randomly positive or negative (it depends of the starting point in the program). So you can't add several files.



For the recombination by turning phase, the TF intensity is always the same. But for the recombination by the program, the intensity vanishes for repetition time faster than 50ms.

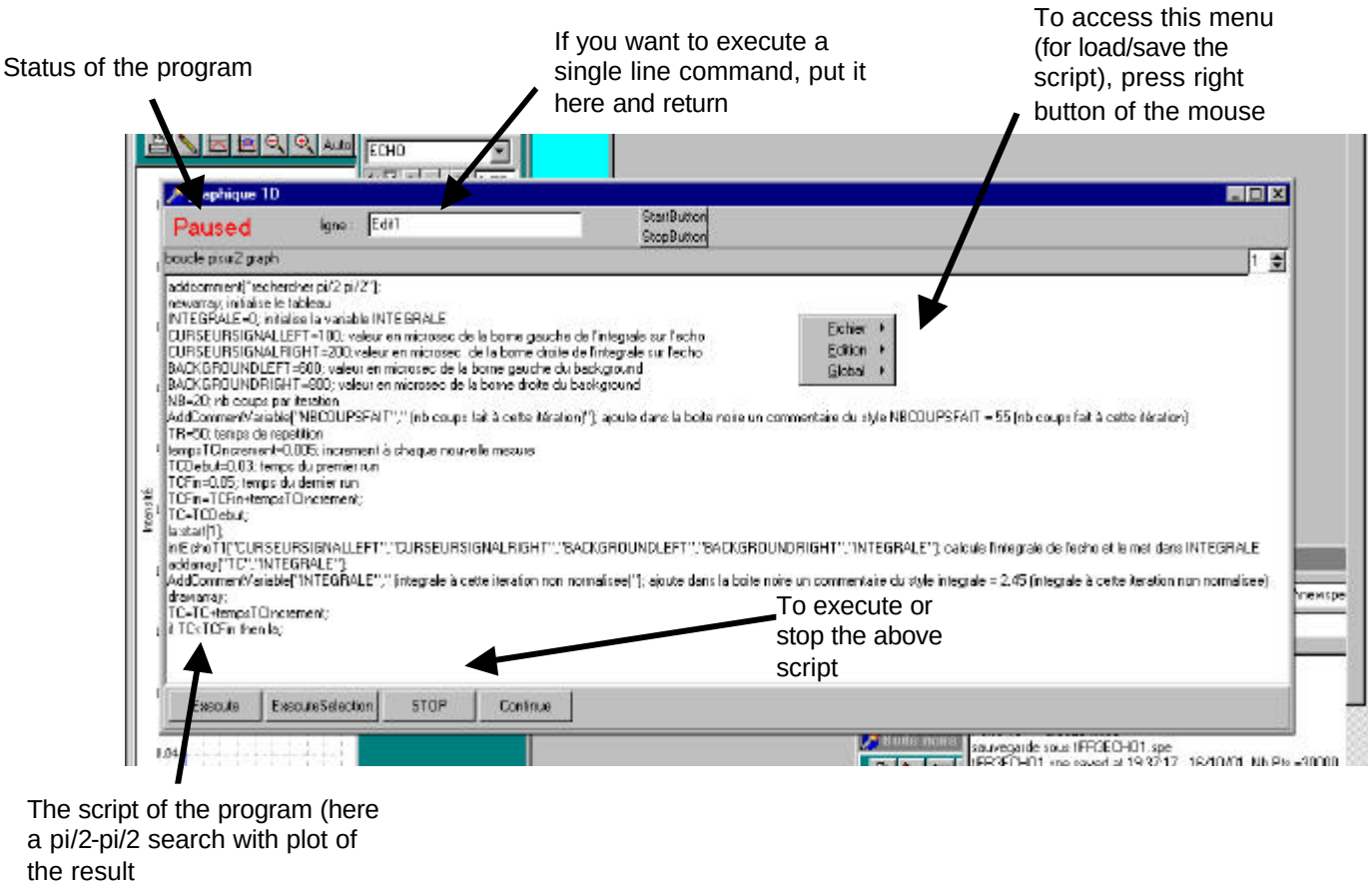
4 Steps Method

The 4 Steps Method has two modes (mode vary and mode 0) like the 2 steps method which have the same advantages and inconvenients. Turning the $\pi/2$-phase in 4 directions cancel furthermore the **ghost effect**: when there is a fantom line symetric of a real one on the other side of the irradiation frequence (line with symbols).



With the 4 steps method (continuous line) is the negative fantom line no more present.

interpréteur window:



It allows to program automatically a lot of instructions, either for doing automatic experiments or also automatic analysis of the data.

Syntax for programming

The syntax for programs is very simple :

- each line must end with ';'
- you can use variables. They must be in capital letters. For example :

  RESULTAT=5;

This will affect the value 5 to the variable RESULTAT. All variables are real numbers (no string no boolean). There are some specific names of variables you cannot use for everything (which are listed below). You can follow the value and name of all variables by clicking on the menu 'variables' when you are in the interpreteur window.

- you can use labels following the syntax :

  instruction 1;
  instruction 2;
  la: instruction 3;
  instruction 4;
  goto la;

here, instruction 3 and 4 will be in an endless loop. Note that there must be an instruction on the line 'la:…'. You can use for labels the name la, lb, lc, ld, le …

- you can use conditions if - then following for example:

NUMBERRUN=1;
La:instruction 1;
Instruction 2;
NUMBERRUN=NUMBERRUN+1;
If NUMBERRUN<= 10 then la;
Instruction 3;

Here, the program will do instruction 1 and 2 ten times, and then go to instruction 3

- you can add comments after ';' like (very usefull next time you look at your program)

NUMBERRUN=1; this variable specifies the run number

- you can use arithmetics functions (all are listed in the menu 'fonctions' when you are in the interpreteur window) for example :

X=5;
RESULTAT=ABS(-COS(X)); this will put in resultat absolute value of -cos(5)

The variables which cannot be used for everything:

Some names of variables are booked for specific tasks. For example, when you write 'TEMPERATURE=200.5;', It will not just put 200.5 in a variable TEMPERATURE, but It will also program the real temperature through ITC4 to go to 200.5. So take care not to use the following names for any purpose, you would have big big surprises !!!

Variable (and an exemple of instruction)	**What it does**
NB=5000;	Puts the number of points to do for acquisition to 5000
TEMPERATURE=300.2;	Puts the temperature in ITC4 to 300.2, putting at the same time heater and gaz flow automatic
TA=0.002;	Puts the pulse TA to 0.002 msec (idem for TB,TC,TD,…)
AA=2000;	Put the amplitude AA of pulse to 2000 (idem for AB,AC,AD…)
PA=1801;	Put the phase of pulse PA to 180° (idem for PB,PC,…)
TZERO=500.54; **BIN=3;** **BGDINF=1254.3;** **BGDSUP=1354.6;** **LEFTLENGTH=150.65;** **RIGHTLENGTH=150.65;** **NBCOMPUTE=4096;** **NBTF=1000;** **FREQ=43252.25**	Changes all these parameters for Fourier TRansform

The instructions

A faire : instructions pour temperature, integrales sur une seule voie, integrale en specifiant le nom du data (par exemple ECHO ou TF) et avec variables possibles pour curseurs comme dans intechoT1

Note that there are already many instructions listed above in the array for specific variables.

All instructions have the form :
Instruction; here, instruction with no input or output variable
Instruction[24,35.5]; here, instruction with two real number variables
Instruction["RESULTAT","ZZ",24.5]; here, instruction with two string variables and one real number variable.

Here are some instructions :

Instructions for acquisition parameters

Startnosave; starts an experiment but does not save it at the end. It will appear as ECHO in the acquisition window.

Start[1]; starts and saves at the end the run in the usual syntax

Start[5]; starts and saves at the end the run in the usual syntax five times. Note that the user wants to interrupt this instruction, if he presses stop from the acquisition window, it will stop the run but do all the others. You need to interrupt delphi itself !

Start["blabla"]; : idem but saves the run under the name blabla

Save[“blabla”]; saves the data named blabla in c:\newspectro\data\Echantillon\Echantillonblabla5.spe**.**

Save[“blabla”,"blablBis2"]; saves the data named blabla (for exple, ECHO at the end of a run) in c:\newspectro\data\Echantillon\blablaBis.spe**.**

saveascii["nom","nom2"]; saves nom in ASCII 3 columns in nom2.dat in the working directory

ChangePulse[« TA » ,0.01] ; changes a pulse value. Here, it changes TA to the value 0.01 in SeqPar window. Caution : it does not mean that the pulse has been physically changed; you have to type then startSequence to validate this change.

CompteNbCoupsDejaFait["NBCOUPSDEJAFAIT"]; puts in the variable NBCOUPSDEJAFAIT the actual nb of shots already done, not the one to be done (usefull if the user stops the experiment from the acquisition window during the unfinished run to interrupt before NB has been done)

AddAndReplace["ECHO","NOM"]; : it will recall NOM.spe, will add to it ECHO and will save back NOM.spe. If NOM.spe does not exist, it will create it. If ECHO does not exist, it will do nothing.

AddAndReplaceT1["T1300K","TEMPS"]; if for exple the value in TEMPS is 0.5,and the name of the working directory is c:\NewSpectro\data\direc, it will recall direcT1300K0_5.spe, and will add to it the last data in ECHO, then saves it back. If direcT1300K0_5.spe does not exist, it will create it

Instructions for the boite noire

Addcomment["mesure en cours"] : adds the comment mesure en cours to the black box
AddCommentVariable["ZZ"]; adds the comment 'ZZ=50' if the variable ZZ exists (50 being its value here)
AddCommentVariable["ZZ","commentaire quelconque"]; idem and adds at the end of the line 'commentaire quelconque' which is a free comment
ParamGAGE : puts in the boite noire the parameters used by GAGE card
ParamReception : puts in the boite noire the parameters used for reception (amplis)

Instructions for analysis of the data

Integ[“blabla”,1,15.5,25.2] : integrates channel 1 of data blabla between 15.5 and 25.2. Result is zero if there was a mistake.
IntegEcho[nom,12.2,32.2,500,800,1]: integrates the two channels of the data nom where signal is taken between 12.2 and 32.2, after substractionof background between 500 and 800. The last parameter 1 gives the sign of the area of channel A to the integral (1 for channel A, 2 for channel B)
IntegEcho[nom, "LEFT","RIGHT","BGLEFT","BGRIGHT",1]; exactly the same but takes the value of the variables LEFT, RIGHT,… for the min and max of cursors.
IntEcho[505,12.2,32.2,500,800,1]: do exactly the same as **IntegEcho** but the data to be integrated can either be "nom" or a number file "505"
IntEchoB[12.2,32.2,500,800,1]: do exactly the same as **IntegEcho** but the data to be integrated is the last ECHO unsaved
intEchoT1[10.3,14.3,25.2,26.2,1,"INTEGRALE"] : puts in INTEGRALE the value of the integrale of the two channels of last data saved where signal is taken between 10.3 and 14.3, after substraction of background between 25.2 and 26.2. The parameter 1 gives the sign of the area of the channel A to the integral (parameter 1 for channel A, 2 for channel B).
intEchoT1["LEFT","RIGHT","BGLEFT","BGRIGHT",1,"INTEGRALE"]; exactly the same but takes the value of the variables LEFT, RIGHT,… for the min and max of cursors.
Int1Voie[12.2,24.5,"INTEGRALE",4]: puts in INTEGRALE the value of the integrale of only one channel of the data 4 in spectlist, without substraction of the background: 4 for TF, 2 for ECHO, 1 for the last data recalled
TF[5] : Fourier Transform of file 5 using the parameters of the TF window
TF["blabla"] : Fourier Transform of file blabla.spe using the parameters of the TF window
TF["blabla","parametre"] or TF[12,"parametre"] : Fourier Transform of file blabla.spe or file n°12 using the parameters of the contained in parametre.tfp saved previously in the TF window
TFT1: Fourier Transform of the last ECHO unsaved using the parameters of the TF window
TFT1["parametre"]: Fourier Transform of the last ECHO unsaved using the parameters of the contained in parametre.ftp saved previously in the TF window

Instructions for manipulation of an array

Tableau : on peut sauver ou recuperer des datas dans un tableau qu'on peut egalement tracer. Il contient maxi 10 colonnes et 1000 lignes :

Newarray : reinitialise le tableau à0
AddArray[1.22,4.12,6.34] ou AddArray["ZZ",4.12,6.34]: ajoute au tableau GraphArray existant 1.22, 4.12 et 6.34 en 1ere 2eme et 3eme colonne (ou la valeur de ZZ, et 4.12, 6.34 idem)
ReadArrayFromFile["nom.zzz","NBLIGNES"] : remplace le tableau GraphArray existant par les valeurs dans nom.zzz qui est un tableau de 2 colonnes ascii (nom.zzz est dans le directory de travail) et met le nb de lignes total dans la variable NBLIGNES
SaveArray["nom.zzz"] : sauve tout le tableau GraphArray dans nom.zzz dans le directory de travail (10 colonnes)
SaveArray["nom.zzz",3] : idem mais seulement les 3 premieres colonnes
PutArray[6.34,3,2] ou PutArray["ZZ",3,2]: met dans le tableau GraphArray existant la valeur 6.34 ou celle de la variable ZZ en ligne 3 colonne 2 (ne peut pas ajouter au dela de la derniere ligne de valeur)
GetFromArray["ZZ",2,3] : met la valeur en ligne 2 colonne 3 dans la variable ZZ
DrawArray : trace dans GraphAlloulForm le data x,y contenu dans les deux premieres colonnes de GraphArray
DrawArray[5,7] : trace dans GraphAlloulForm le data x,y contenu dans le colonnes 5 et 7de GraphArray

Some examples of programs

A simple experiment

```
Addcomment["experiment at 300K"]; adds the comment in the black box
NB=20000;  specifies nb of points for each acquisition
TR=50; repetition time in the sequence (msec)
Start[1]; starts and saves one acquisition
NB=30000; specifies another nb of points for each acquisition
Start[5]; starts and saves five acquisitions
```

A search for pi/2-pi optimal

```
Addcomment["loop for pi/2-pi search"]; adds the comment in the black box
NB=20000;  specifies nb of points for each acquisition
TR=50; repetition time in the sequence (msec)
PulseIncrement=0.001; increment in msec between two steps for the search
TADebut=0.001; starting pulse length for pi/2
TAFin=0.01; ending pulse
TAFin=TAFin+PulseIncrement; in order to do the last step also
TA=TADebut; initialize the value for pi/2
TB=2*TA; idem for pi
la:start[1];  begining of the loop : starts and saves the acquisition
TA=TA+PulseIncrement; at each step, we increase TA
TB=2*TA; we increase TB
if TA<TAFin then la; end of the loop : if we reach the last length, we stop the loop
TA=TADebut; reinitialization of the pulse length for pi/2
TB=2*TADebut; reinitialization of the pulse length for pi
```

A search for pi/2-pi/2 optimal with plot of the intensity of the echo versus pi/2

```
addcomment["rechercher pi/2 pi/2"];
```

newarray; initialise le tableau
INTEGRALE=0; initialise la variable INTEGRALE
CURSEURSIGNALLEFT=100; valeur en microsec de la borne gauche de l'integrale sur l'echo
CURSEURSIGNALRIGHT=200;valeur en microsec de la borne droite de l'integrale sur l'echo
BACKGROUNDLEFT=600; valeur en microsec de la borne gauche du background
BACKGROUNDRIGHT=800; valeur en microsec de la borne droite du background
NB=20; nb coups par iteration
AddCommentVariable["NBCOUPSFAIT"," (nb coups fait àcette itération)"]; ajoute dans la boite noire un commentaire du style NBCOUPSFAIT = 55 (nb coups fait àcette itération)
TR=50; temps de repetition
tempsTCIncrement=0.005; increment àchaque nouvelle mesure
TCDebut=0.03; temps du premier run
TCFin=0.05; temps du dernier run
TCFin=TCFin+tempsTCIncrement;
TC=TCDebut;
la:start[1];
intEchoT1["CURSEURSIGNALLEFT","CURSEURSIGNALRIGHT","BACKGROUNDLEFT", "BACKGROUNDRIGHT","INTEGRALE"]; calcule l'integrale de l'echo (racine de somme des integrales au carre) et le met dans INTEGRALE
addarray["TC","INTEGRALE"];
AddCommentVariable["INTEGRALE"," (integrale àcette iteration non normalisee)"]; ajoute dans la boite noire un commentaire du style integrale = 2.45 (integrale àcette iteration non normalisee)
drawarray;
TC=TC+tempsTCIncrement;
if TC<TCFin then la;

T1 RUN

; En entree : tableau de 2 colonnes ascii contenant le temps àchanger en msec et le nb de points àfaire pour ce temps
; En sortie : un tableau de 6 colonnes avec successivement : temps / nb points àfaire par iteration/ integrale totale normalisee par nb coups fait / integrale totale non normalisee / nb coups total deja fait
; fichiers sauves sous un format du genre directoryT1MANIP300K_0,05.spe (pour le temps 0.05 ici)
; pour sortir du programme : si on appuye sur STOP de cette fenetre interpreteur. Il va arreter sans tenir compte du run en cours. Si STOP de la fenetre d'acquisition, il arrete en tenant compte du run en cours.

CURSEURSIGNALLEFT=5.1; valeur en microsec de la borne gauche de l'integrale sur l'echo
CURSEURSIGNALRIGHT =5.1;valeur en microsec de la borne droite de l'integrale sur l'echo
BGDLEFT=12.85; valeur en microsec de la borne gauche du background
BGDLEFT =18.15; valeur en microsec de la borne droite du background

ChargeParamTF["parametres TF"]; charge dans le directory de travail le fichier parametres TF.tfp qui contient les parametres de la TF (facultatif)
ParamTF; affiche dans boite noire les parametres de la TF

addcomment["RUN DE T1"]; commentaire dans boite noire

ReadArrayFromFile["TableauT1.dat","NbIterations"]; lecture du fichier ascii 2 colonnes temps / nb de points àfaire
RunNumber=0; variable RunNumber repere le numero de la boucle de mesure en cours

la:RunNumber=RunNumber+1; debut de la boucle de moyennage

Addcomment["********** NOUVELLE BOUCLE **********"]; commentaire dans boite noire
AddCommentVariable["RunNumber"]; ajoute dans la boite noire un commentaire du style RunNumber = 5

Iteration=0; Iteration donne le numero de l'iteration en cours au sein de la boucle de T1

lb:iteration=iteration+1; on passe àl'iteration suivante dans le tableau de mesures de T1

GetFromArray["NBCoups",iteration,1]; recupere dans le tableau le nb de points voulu en 1ere colonne ligne n°iteration et la met dans NBCoups
NB=NBCoups; initialise le nb de coups àNBCoups dans la fenetre d'acquisition

GetFromArray["TEMPS",iteration,2]; recupere dans le tableau la valeur de temps en 2eme colonne ligne n°iteration et la met dans TEMPS
TD=TEMPS; initialise le temps TD àla valeur de TEMPS dans la fenetre de sequenceur

AddCommentVariable["iteration"]; ajoute dans la boite noire un commentaire du style iteration = 3
AddCommentVariable["TEMPS"]; ajoute dans la boite noire un commentaire du style TEMPS = 0.05

StartNoSave; lance l'acquisition sans sauver àla fin : elle est donc dans ECHO dans la fenetre d'acquisition

AddAndReplaceT1["T1300K_","TEMPS"]; ajoute ECHO au fichier directoryT1300K_0,03.spe dans c:\NewSpectro\data\directory (et le cree si il n'existe pas) (ds cet exple, TEMPS=0.03)

compteNBcoupsDejaFait["NBCOUPSFAIT"]; met dans la variable NBCOUPSFAIT le nb de coups deja fait (qui peut differer du nb de coups àfaire si programme interrompu)
GetFromArray["NBTOTAL",iteration,5]; on recupere le nb de coups deja fait àcette itteration
NBTOTAL=NBTOTAL+NBCOUPSFAIT; on ajoute le nb de coups qui vient d'etre fait àce qui avait été fait avant

intEchoT1["CURSEURSIGNALLEFT","CURSEURSIGNALRIGHT",
,"BGDLEFT","BGDRIGHT","INTEGRALE"]; calcule l'integrale de l'echo (racine de somme des integrales au carre) et le met dans INTEGRALE

GetFromArray["INTEGRALETOTALE",iteration,4]; on recupere la vieille integrale
INTEGRALETOTALE=INTEGRALE; on ajoute la nouvelle a la vieille

INTEGRALENORMALISEE=INTEGRALETOTALE/NBTOTAL; renormalise l'integrale totale par le nombre de coups deja fait total

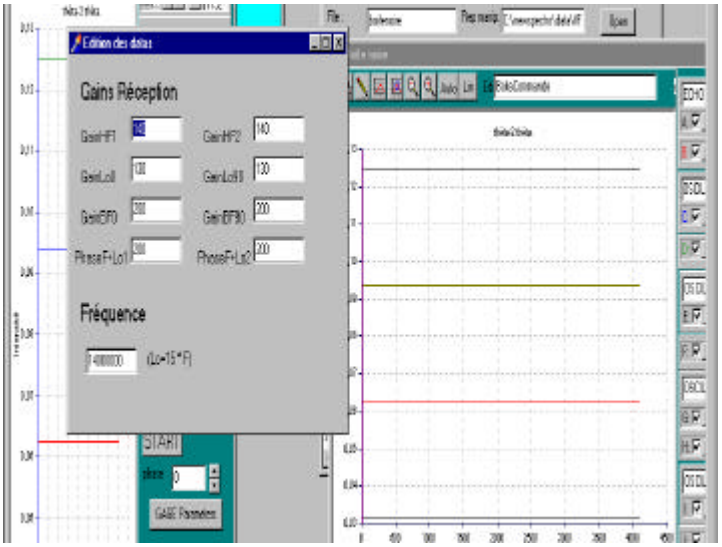
PutArray["INTEGRALENORMALISEE",iteration,3];ajoute l'integrale normalisee totale en 3eme colonne
PutArray["INTEGRALETOTALE",iteration,4];ajoute l'integrale totale non normalisee en 4eme colonne
PutArray["NBTOTAL",iteration,5]; met dans la 5eme colonne du tableau le nb coup total fait àcette iteration
SaveArray["Resultats T1 300K.dat"]; sauve la nouvelle version du tableau dans Resultats T1 300K.dat dans le directory de travail

if Runnumber=1 then AddComment["resultats sauves dans : Resultats T1 300K.dat"]; au premier run, indique dans la boite noire le nom du data ou est sauve le tableau de resultats
AddCommentVariable["NBCOUPSFAIT"," (nb coups fait àcette itération)"]; ajoute dans la boite noire un commentaire du style NBCOUPSFAIT = 55 (nb coups fait àcette itération)
AddCommentVariable["INTEGRALE"," (integrale àcette iteration non normalisee)"]; ajoute dans la boite noire un commentaire du style integrale = 2.45 (integrale àcette iteration non normalisee)

DrawArray[1,3]; trace l'integrale normalisee en fonction du temps

if iteration<NbIterations then goto la; fin de la boucle de T1
goto la; fin de la boucle de moyennage

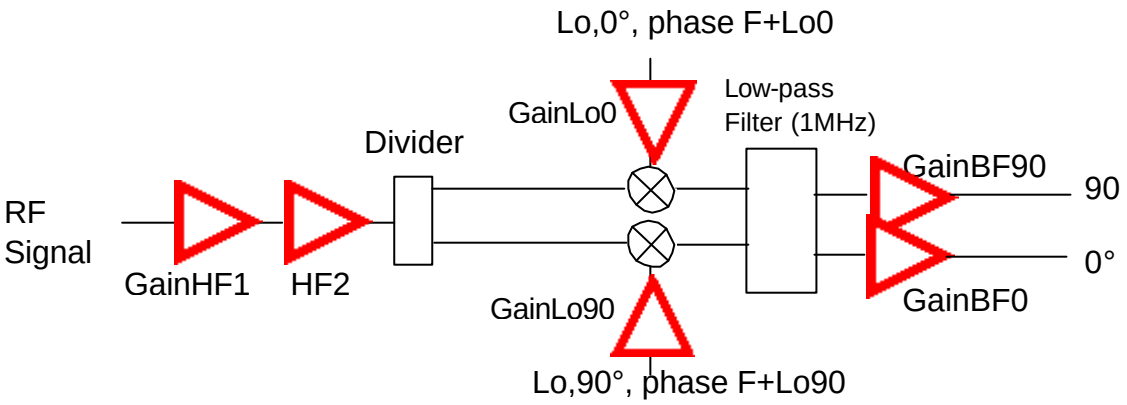
Reception Amplifiers Window



It allows to specify the parameters for reception. The frequence window specifies the Lo parameter and must be at 14000000.

Each time the program starts, it validates these parameters again.

Here is the scheme of the amplification :



The rf signal is first amplified by 2 equivalent amplifiers. Then it is demodulated, and the demodulation signal can be also amplified before demodulation by GainLo0, GainLo90. Finally, the demodulated low frequency BF signal is amplified on each phase. Note that the phase of each channel can be changed by changing the phase of Lo0 and Lo90. Hence, if you don't put the same value for Phase Lo0 and Lo90, or don't put the same gain on BF0 and BF90, you will introduce a dephasing of one channel to the other. (This can be used to recorrect wrong 90° dephasing).

Variation of the amplification versus the different gain values :

The gain varies as a function of an applied voltage. This voltage can be tuned between 0 and 255 on the reception window. The amplification is not linear at all with respect to this command. The graph below represents the total amplification (without the Doty preamplifier) when other reception gains are kept constant (to GainHf1 = 140, GainHf2 = 160, GainLo0 = GainLo90 = 181 GainBF0=GainBF90= 170) with two input signals (75 MHz) with amplitude -40 dB or -60 dB. The maximum corresponds to 0 (about +20dB) and minimum to 255 (about -10 dB). These amplifiers

saturate for an input signal of about -35 dB, the signal is deformed (but it is not easily detectable).

GainLo0,GainLo90 : Gain applied to the Lo frequency. The total gain should not depend from these values as long as the RF level is sufficient for the demodulation to work properly. In practice, values between 100 and 200 are correct and 181 is usually used. There are sometimes problems with offsets associated with these gains (see below *how to set gains*).

GainBF0,GainBF90 : Two low frequency gain (40 dB) on separate channels. For them, minimum corresponds to 0 and maximum to 255 (see graph).

How to set gains:

The usual values are :

GAINHF1 = 140	GAINHF2 = 160	At 75 MHz : gain between 60 and 70 dB
GAINLO0 = 181	GAINLO90 = 181	noise : 200 mV with 190
GAINBF0 = 170 to 190	G AINBF90 = 170 to 190	

One cannot set gains independently from each other. For given values of (GAINHF1,GAINHF2) (chosen to avoid saturation), there is only a limited range of (GAINBF0,GAINBF90) and (GAINLO0,GAINLO90) values that will work properly, which means they don't deform the signal, keep the quadrature to 90° and correspond to identical gains on the two channels. I usually keep GAINHF1 and GAINHF2 fixed and change GAINBF0 and GAINBF90 from 170 to 190 depending on the signal amplitude.

The choice of (GAINLO0,GAINLO90) is crucial to minimize **offsets**. With a wrong setting, these offsets can be large and dissymetric on the two channels (which can result in slightly different gains for the two channels even for the same parameter values). They are sometimes unstable with time, which can cause baseline problems (especially at low frequency ?). These offsets are not substracted with the usual sequences, see *acquisition* to see how to get rid of them and/or compensate inequal gains on 0° and 90°, if necessary. For some frequencies, these offsets obviously oscillate (look for example at 70 MHz). This occurs for multiples of 14 MHz, so that it might be necessary to shift the reference 14 MHz in order to work at these frequencies (but care must be taken that the operating frequency for the reception (normally 210 MHz =15*14 MHz) is still OK).

Other settings :

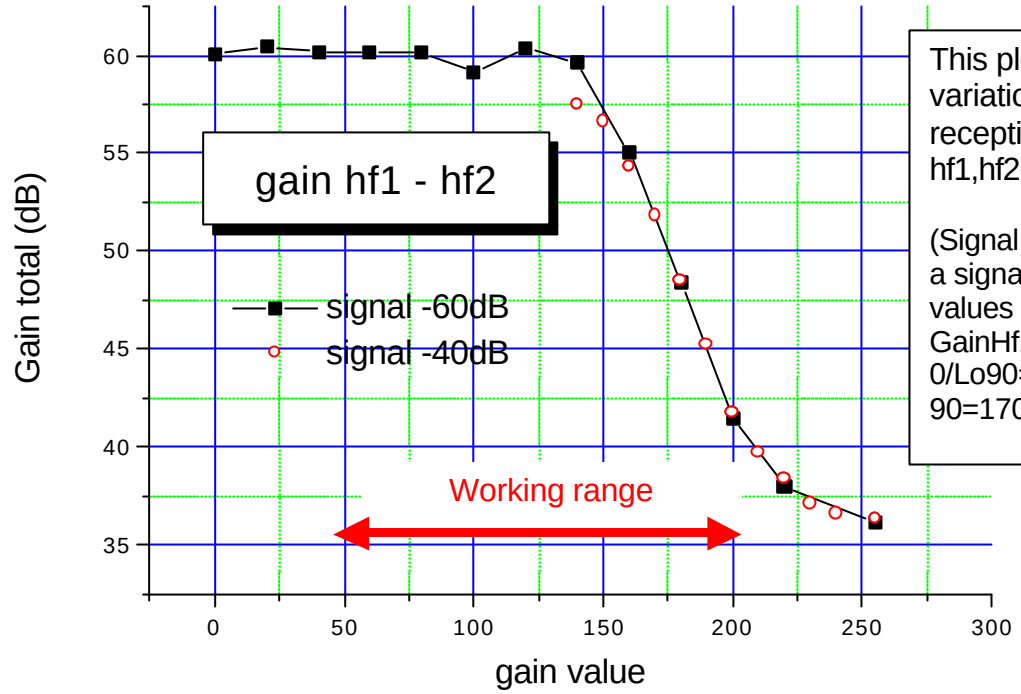
- For a higher gain (about 80dB which is near the maximum):

GAINHF1 = 140	GAINHF2 = 120
GAINLO0 = 131	GAINLO90 = 131
GAINBF0 = 200	GAINBF90 = 200

- For low frequency (used by Feri)

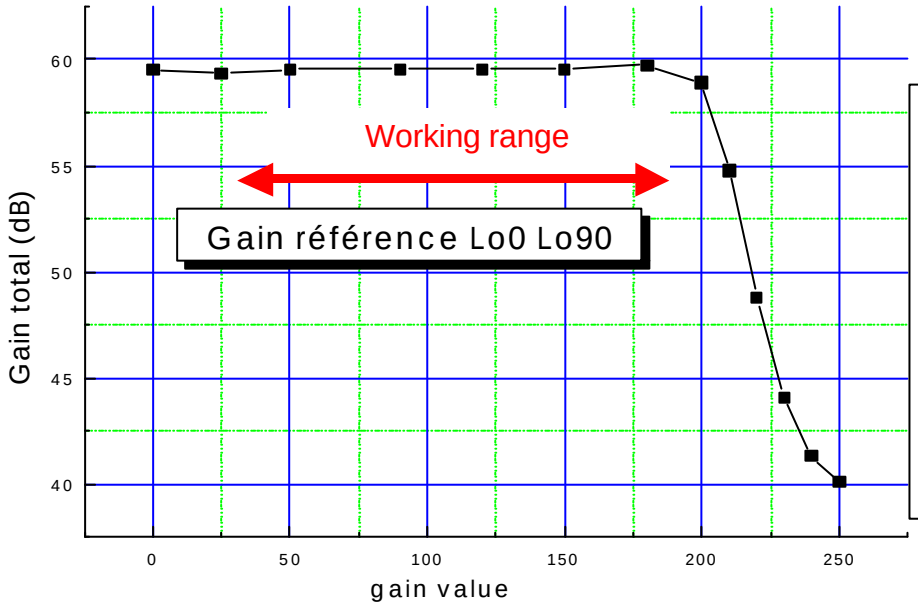
GAINHF1 = 175	GAINHF2 = 161
GAINLO0 = 208	GAINLO90 = 204
GAINBF0 = 184	GAINBF90 = 182

Warning : the gain seems to be anomalously low below 20-30 MHz. To be checked.



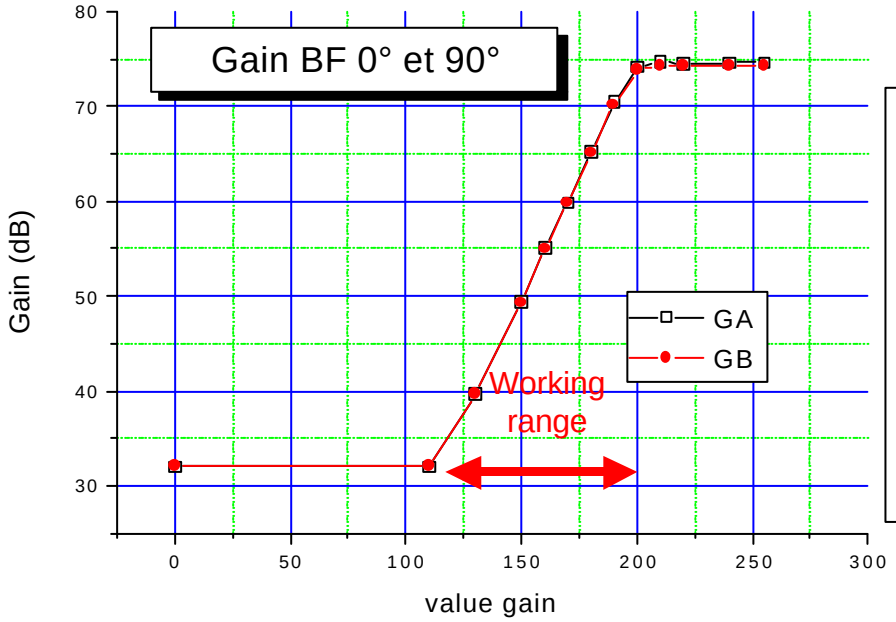
This plot shows the variation of the gain of the reception when changing hf1,hf2

(Signal taken at 75 MHz, with a signal -60dB, with the values GainHf1/Hf2=140/160;GainLo0/Lo90=181/181;GainBF0,BF90=170/170;)



This plot shows the variation of the gain of the reception when changing Lo0,Lo90

(Signal taken at 75 MHz, with a signal -60dB, with the values GainHf1/Hf2=140/160;GainLo0/Lo90=181/181;GainBF0,BF90=170/170;)



This plot shows the variation of the gain of the reception when changing BF0,BF90,

(Signal taken at 75 MHz, with a signal -60dB, with the values GainHf1/Hf2=140/160;GainLo0/Lo90=181/181;GainBF0,BF90=170/170;)

Help for those who would like to program

Version qui marche : dans c:\newspectro\pilotagemanip3,

Attention : Penser àavoir un directory c:\newspectro\data
Et dedans un fichier current.txt qui contient en 1ere ligne le nom du directory en cours (exple : c:\newspectro\data\nouveau) et dans ce directory c:\newspectro\data\nouveau, un autre fichier info.txt qui contient en 1ere ligne ànouveau ce nom c:\newspectro\data\nouveau, et en 2eme ligne le numéro du dernier fichier sauvé dedans (0 si pas de fichier).

General rules :

when you want to change something, first do a backup on the disk of the present version of the program, and put the date of the backup in the title of the directory (exple : c:\newspectro\backup pilotagemanip3 11 august 2001).
When you add something, put a remark as {groupe Alloul } in front of any new procedure. Try to put it also at the end of the unit if possible.

Data Format

The data format and its use is done in **editdata** :
All data are contained in a variable named und.spect which has the following properties :
Und.spect[i].nom is the name (exple 'ECHO') of data number i
Und.spect[i].X is the number of points
Und.spect[i].Y must be put at 1 (one dimensional data, no Y scale)
Und.spect[i].comment is a possible comment (exple 'this is a signal of NMR')
Und.spect[i].typex is defining the type of x data to distinguish between different kinds (here 'rmn')
Und.spect[i].typey is here single (no 2^{nd} dimension)
Und.spect[i].data[j,0].x contains the x value of the jth point of the data (second number at 0 no as 2^{nd} dimension, so that Und.spect[i].data[j,0].y is not used)
Und.spect[i].data[j,0].z contains the z value
Und.spect[i].data[j,0].mon contains another possible value

For NMR time signal: we use .x for the time, .z and .mon for real and imaginary intensity
For NMR Fourier Transform: we use .x for the frequency, .z and .mon for real and imaginary intensity

Indexof(name) gives the index associated to name. For example, to use the data 'ECHO', you can use und.spect[indexof('ECHO)]

One can also use different procedures applicable to und.spect :
Und.spect.clear, total, minmax, savetofile, readfromfile, savetoPAXY, readfromPAXY,free,
procedure free; addspectre, deletespectre, addition, soustraction, saveall, add, addpt.

Structure of the program

Important units/Files are the following:

*** PilotageManip** : the source of the project, which contains which window is created at begining
(exple : Application.CreateForm(TseqPar, SeqPar);) and which units are used in the full program.

*** PilotageManipMainForm/PrincipalForm** :the main window which contains all the other. Contains the initialisation of the initial data :
Exple :

```
ind:=und.addspectre('DATA',4096,1);
und.spect[ind].comment:='RMN';
und.spect[ind].typex:='rmn';
und.spect[ind].typey:='single';
for i:=0 to 4095 do begin
  und.spect[ind].data[i,0].x:=i;
  und.spect[ind].data[i,0].z:=1000*sin(i*2*3.14/10)*exp(-i/1000);
  und.spect[ind].data[i,0].mon:=1000*cos(i*2*3.14/10)*exp(-i/1000);
end;
```

At begining, it creates 4 files : DATA, ECHO, OSCILLO, TF of the rmn type.

*** AcquisitionRMN/NMRForm :** a graphic window which is aimed at the acquisition of the data only. Thus, the appearance of data is a little bit different than in usual graphic window : each data appears in two channel at once (A,B or C,D), and the .z part appears in A, the .mon part appears in B (respectively C,D). So real and imaginary part of the signal are always appearing together.

There is a Directory window and a n° window. These two contain the working directory and the number of the last saved data. To make it work, the program uses

- a file named c:\newspectro\data\current.txt which contains in text format the full path of the working directory (exple : c:\newspectro\data\nouveau). If this file is not present, the program cannot work.
- A file in the working directory named info.txt which contains again the working directory and the number of last saved filed

exple :

```
c:\newspectro\data\nouveau
56
```

If this file is not present, again it will not work.

To change working directory, press the directory button. Note that these informations are the same in the TF window as well.

The code of **acquisitionRMN** is modified as compared to a standard graphic window : it contains now a modified version of **tfrspectre** to allow the special im/real part display explained hereabove.
It is also linked to real acquisition of data through **Rafraichir** : a procedure called during acquisition and linked to a timer which regularly refresh the A/B channel putting in it the total signal, and in C/D the last instantaneous signal. So A/B is called ECHO, and C/D OSCILLO.

It has a blank command box above the graph which allows specific NMR commands. These commands appear in **traitchaine** in the following way: if for exemple to save a file, you want to type s or S you write the code in traitchain :

```
 if (s='S') or (s='s') then
```

```
  begin
  dataeditform.sauveNMR(cocourbe1.text);  {this goes in editdata and uses the procedure
sauveNMR to save data}
  RafraichissementPileEtDirectoryNMR; {this calls the refreshment of number of the
spectrum and directory informations}
  cocourbe1.text:=und.spect[1].nom; {this puts the new data name in the front pannel}
  end;
```

You can add there your specific commands for this window.

*** CompilRMN** : contains especially **compilo** , the procedure which compiles a pulse sequence from the SeqPar window. This is where the syntax of compilation of sequences is written.

*** EditData :** as explained above, contains the procedures and functions which deal with the data contained in und.spect. We added there specific procedures for NMR data format :

- sauveNMR saves the data in A/B. If the data is named ECHO, and the current directory is c:\newspectro\data\nouveau and the last saved file has number 10, it will save it in the name nouveauECHO11.spe in a format readable only by the program. This is similar with any other name (TF will lead to nouveauTF11).
- recallNMR recalls the data saved through sauveNMR (i.e in the format nouveauECHO10.spe if directory is named nouveau)
- DecaleSpectres : used in sauve and recall, reorganises the spectra in spectlist :

1st data is the present data, either the last one recalled or saved, or acquired…
2nd data is ECHO
3rd data is OSCILLO
4th data is TF
5th data is the previously called or saved,
6th data is the previously called or saved,
etc… up to 10 data maximum (this limit number of authorized files in spectlist is fixed in decalespectres in the variable MaxiNombreDeFichierDansSpectListe). - -

- TransfertSpectre : used by DecaleSpectres, please do not use it from anywhere else, it is very tricky.

*** Interpreteur/InterpreteurForm :** the unit allowing batchs for programation. If you want to add your own command, do it using one already existing as a model. For exemple, here is the function which changes a parameter in the pulse sequence window seqPar : one uses for exemple ChangePulse["Ta",0.01];

```
    function ChangePulse:integer;far;
      begin
      SeqPar.ChangeParametreSequence(pars[1],par[2]);
      result:=1;
      end;
```

one must also add to the list of commands ordre.addobject the line :
ordre.addobject('ChangePulse',tobject(@ChangePulse));
Each command has a form like : name["blabla",0.03,0.04,"boubou"];
Here, name is the name of the function called, and each variable is either a string (here blabla and boubou) or a real number (here 0.03 and 0.04). When you use these variables,

the string are noted **pars** and the numbers **par**. So here pars[1] = 'blabla', par[2] = 0.03, par[3] = 0.04, pars[4] = 'boubou'.

If you want to add a variable that you use then (example the variable ALPHA), first go to the line
if NMRFORM<>nil then begin
addvariable('TA',0,0,0,0,'',tobject(@ChangePulse));
and put the line
addvariable('ALPHA',0,0,0,0,'',tobject(@rien));

A more sophisticated command for variable allows you to use in the interpreteur a line like ALPHA=4. Let's assume we want that this command makes the alpha box in the TFNMRForm Window to be then changed to 4. The way to do it is : do the addvariable as explained above, but with the line
addvariable('ALPHA',0,0,0,0,'',tobject(@ChangeAlphaValue);

Then, add a function as explained above of the form :

```
function ChangeAlphaValue(s:string):integer;far;
  var Sender:TOBject;x:extended;chaine:string;
   begin
     x:=param[variable.indexof(s)].value;
     str(x:10:2,chaine);
   TFNMRForm.AlphaBox.text:=chaine;
   result:=1;
   end;
```

Finally, do not forget to add the command at the proper place
ordre.addobject('ChangeAlphaValue',tobject(@ChangeAlphaValue));

- **GestGage/acquisParam :** the unit governing the gage card for acquisition of the NMR signal.

- **ParSeq/SeqPar** : the unit governing the pulse sequences.

- **Reception/Fenetre_Reception :** the unit governing the reception parameters (amplifiers and frequency of the reception part of the spectrometer).

TFRMN/TFNMRForm : exactly like **acquisitionRMN**, but for TF purpose. All the S, R, directory procedures are the same. But it is also designed to do TF

When you add a new window/unit : It must be open through the pilotagemanip unit as
Application.CreateForm(TSeqPar, SeqPar);
You have in the Inspecteur d'objet of the window to put on the parameter FormStyle the choice fsMDIChild. You also have to put a TAG number not used by the other windows (each window must have its own TAG). TAG allows then the database to remember the parameters of the window.

Problems, bugs, to be done :
- gestGage est different des autres unités. Quand depuis une fenetre qcq (par exemple l'interpreteur), on veut modifier un parametre d'une autre fenetre (par exemple seqPar), il suffit de mettre dans le uses de l'interpreteur l'unité SeqPar, puis de tapper une commande du genre seqPar.Fenetre.text:='100'; (si on veut changer Fenetre qui se trouve

dans SeqPar). Ceci ne marche pas pour gestgage/acquisParam : qd on debugge, il a en fait perdu l'adresse de acquisParam (qui est ànil) dés qu'on quitte cette fenetre. Consequence : on ne peut pas lancer de run depuis gestgage ou ailleurs, ni faire quoique ce soit sur gestgage. idee pour s'en sortir : que gestgage àl'envers, scrute un drapeau lié àl'interpreteur. Qd ce drapeau est allumé, il lance l'acquisition mais pb circulaire car gestgage utilise alors l'interpreteur, mais comment
l'interpreteur peut il utiliser alors gestgage pour savoir qd c'est fini ?

- qd on change un parametre de la reception (exple un ampli), il met 6 minutes àle changer
- qd on lance la séquence de pulses, il met 10 secondes àla lancer
- probleme avec les axes du graphe : souvent fantaisistes, et pas de rescaling automatique qd on veut agrandir
- souvent un bug au démarage (aléatoire) de division par zéro
- comment organiser les fenetres du départ (et leur taille) comme on aimerait et que ca reste (sauvegarder config ne marche pas) ?

àfaire
la fenetre d'analyse: pouvoir rappeler et sauver des spectres soit avec .z et .mon àla fois comme dans les fenetres acquis et TF; soit en rappelant une des deux voies. Reflechir à comment sauver : en echo, en TF, en autre chose ?

qd on sauve, si nom du genre nouveauECHO14 déjà, ne pas faire
nouveaunouveauECHO1415
mais juste nouveauECHO15

conversion en ascii

mettre la base de temps en x dans le data, puis la bonne frequence dans la TF

des commandes dans l'interpréteur, genre sommer des spectres, faire des intégrales, faire la TF, rephaser automatiquement la TF, changer des parametres de TF

rappel et sauvegarde auto de parametres de TF ?

voir ce qu'on met dans la boite noire

qd on y arrive, transferer le bouton d'acquisition de acquisparam vers acquisitionRMN

verifier le changement de points depuis acquisparam

traiter les erreurs si info.txt ou current.txt n'existent pas avec fexis(nomfichier)…

faire ITC4

VERONIQUE's NOTEs

Pannes de spectro

Quand rien ne marche :
- Vérifier que les alims fonctionnent
- Essayer de tout redémarrer en éteignant et rallumant le spectro et le PC.
- Vérifier (en enlevant les cartes et en les remettant ?) qu'il n'y a pas de faux contacts sur les cartes PIA et/ou les bus.

Premières vérifications à faire à partir de la face avant :

Synthèse du 60 MHz : horloge
- La sortie 10 MHz du générateur de fréquence doit être un créneau d'amplitude pic-pic (sous 50 ohms) 2.4 V.
- La sortie 60 MHz TTL doit être une sinusoï de (un peu déformée) de niveau TTL (3.5 V sans 50 ohms). Elle n'est pas centrée àzéro (elle va de 0 à3.5 V).

Déphaseur double DDS
- La sortie Out DDS 1 est une sinusoï de àla fréquence de base utilisée pour construire LO (14 MHz pour nous) d'amplitude pic-pic 300 mV (sous 50 ohms).
- Sa phase doit varier selon la séquence utilisée (instructions PB01 dans la séquence).
Sinon : essayer de changer sa valeur àpartir de la fenêtre réception pour réinitialiser.

Bi-Synthese LO & F
- La sortie LO (n*F) doit être une sinusoï de à210 MHz. La mesurer avec un oscillo 200 MHz.
- La sortie F doit être une sinusoï de àla fréquence d'irradiation d'amplitude pic-pic 500 mV. C'est normal qu'elle paraisse un peu déformée.

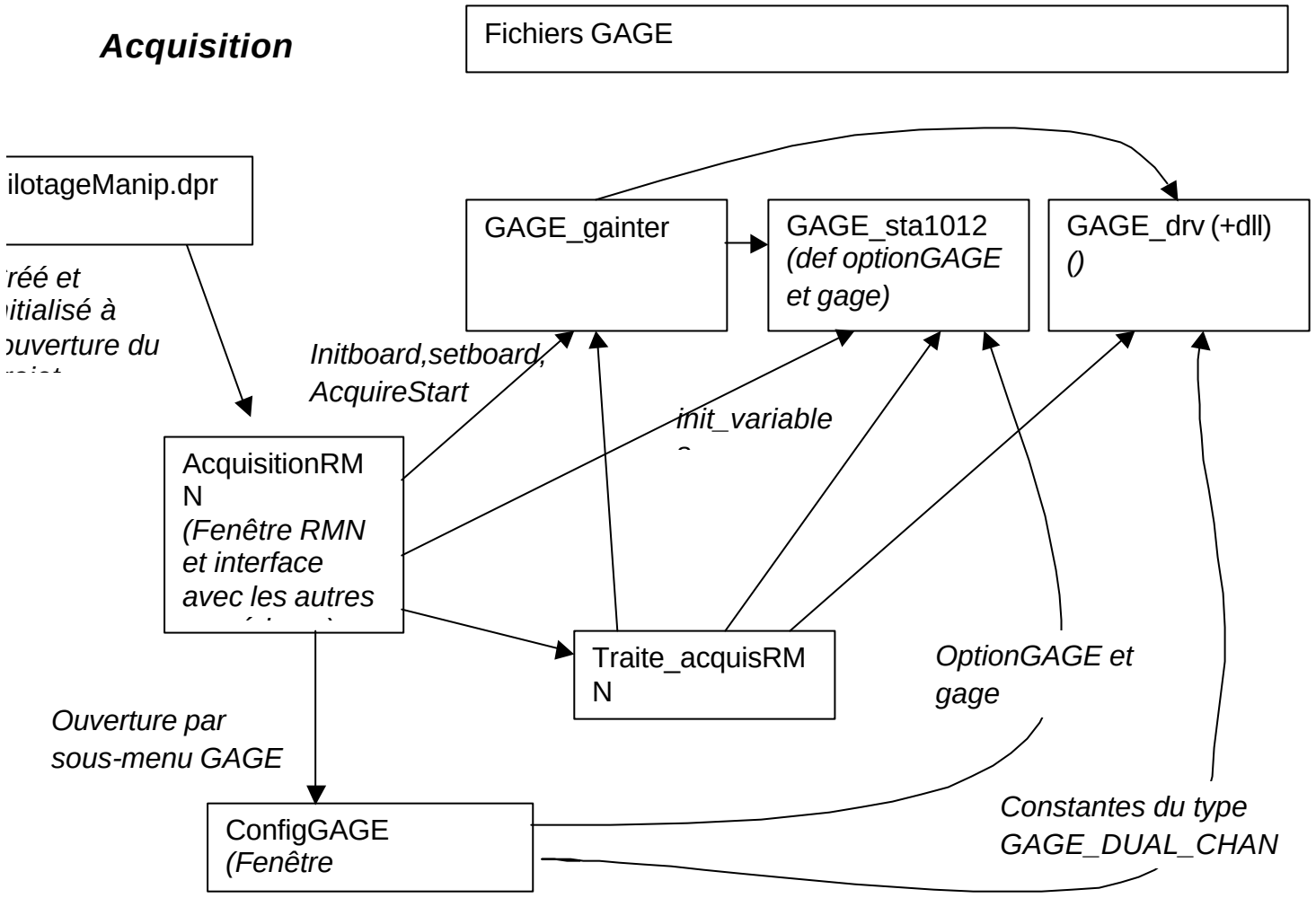
Modulation d'amplitude
- Il doit y avoir des pulses en sortie quand une séquence tourne (logique).
- Amplitude environ 300 mV pour AA=1000.
Attention : l'alim 5 V doit être au-dessus de sa valeur nominale pour que le séquenceur fonctionne normalement : environ 5.6 V. (A 5V rien en sortie, à5.5 V, seuls les temps courts marchent).

Réception
- Il peut arriver que les gains effectifs ne correspondent plus aux valeurs demandées (surveiller le bruit en sortie). En général, il suffit de redemander les bons gains (ou de redémarrer le programme) pour que ça redevienne normal.

Sorties CA, CB, CE, CF
Attention : CC et CD ne fonctionnent pas.
Ce sont des sorties TTL.
Ils ne sont pas très bien isolés les uns des autres. On peut par exemple souvent voir une petite impulsion sur A si on déclenche un autre switch.

Modifs Programme spectro (Juin 2001)

Acquisition



Rq : il n'y a pas grand chose dans Traite_acquisRMN (juste le transfert des data) et ça alourdit peut-être la structure : procédure àrappatrier dans AcquisitionRMN ?

1/ Il faut utiliser le programme gage_drv. pas utilisé par le programme de test de Fermon (dans programme/composant/fftacquis) sinon, il ne trouve pas la carte gage.
2/ Mais il faut utiliser gage_drv.dll fourni par gage (CD) adapté àla carte ISA cs1012 qu'on met dans Windows\system avec GageEx.dll et Gagedrv.vxd

3/ Plantages actuels : des instructions de communication avec la carte GAGE (Initboard, setboard) détruisent certains pointeurs, ce qui crée des erreurs en série.
- dans mon programme : les pointeurs détruits peuvent être ceux des fiches tels que NMRForm etc. Selon qu'on a ouvert ou non la fenêtre configGAGE, de nouveaux pointeurs peuvent être détruits.
- dans le programme test de Fermon :
    - Il marche bien en tournant tel quel
    - Introduit dans l'ensemble du spectro (voir NewSpectro/PilotageManip3), il plante si on utilise le fichier graph1D.pas au lieu du graph2 de fftacquis.

Seule config qui marche actuellement : PilotageManip3 en laissant en uses graph2 et la création d'un graphe correspondant, mais sans l'utiliser (parce qu'on n'a pas créé la

fenêtre correspondante. Il faut ensuite raffraichir àla main FIDTESTR dans une des fenêtres graph1d.

A remettre dans cette version : l'ouverture de la fenêtre réception

- Fenêtre 'Acquisition RMN' créée àl'ouverture du projet (voir source du projet, qui contient aussi l'initialisation de la carte GAGE et de la réception). Le fichier associé est AcquisitionRMN.pas : fichier graph1D.pas + procedures spécifiques RMN (àla fin) (àne pas confondre avec le fichier Fermon RMNacquisition.pas).
La partie du fichier copiée de graph1d de AcquisitionRMN était modifié pour afficher .mon en tant que partie imaginaire (réfléchir si on veut le faire dans toutes les fenêtres, ce qui nous empècherait de comparer 4 spectres)

- Les pointeurs optionGAGE et gage ne sont utilisés que par GAGE_sta1012 et GAGE_gainter. Ils sont définis dans GAGE_sta1012 qu'utilise GAGE_gainter.

- OptionGAGE : paramètres par défaut dans sta1012
Noter qu'il faut recopier le fichier Gage_drv.dll

Rq : par rapport àla syntaxe Fermon telle qu'elle est utilisée dans FFTacquis, j'ai renommée curpa (nom dans specrmn), optionGAGE.

Rq sur tous les problèmes d'initialisation :

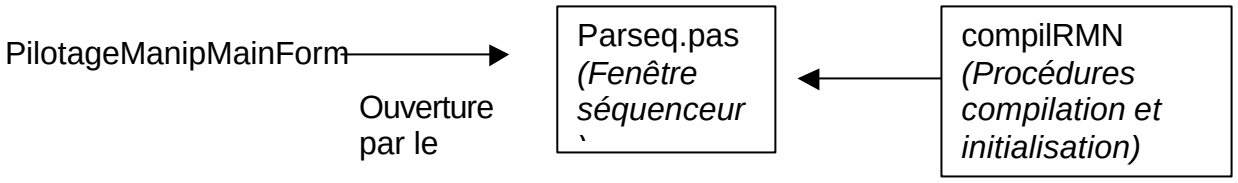
Pour ne pas détruire le pointeur sur NMRform en initialisant la carte, je le fais avant. Il y a tout de même des problèmes, au niveau de Init_variables (de sta1012) qui crée les pointeurs gage et optiongage, que le programme ne comprend plus selon qu'on a ouvert ou non la carte configGAGE.

J'ai essayé d'ouvrir gage tout le temps en changeant seuement la propriété Visible/ non visible, mais ça interfère quand même.

En particulier on a parfois la quantité gage^.buffer_mask à0 (ce qui provoque une erreur dans le transfert des data de TraiteAcquisRMN, elle devrait être à263000 etc. (ce qu'elle est au début).

Peut-être possible de s'en sortir en organisant convenablement tout ça et en créant et détruisant les pointeurs aux endroits sensibles...

Séquenceur



Parseq.pas

Warning (à faire) : il faudrait ne pouvoir ouvrir qu'une fois cette fenêtre, sans quoi on risque de sélectionner deux séquences, définir plusieurs fois les pointeurs sequenceur et gain_emission etc.

Contient une fenêtre appelée par le menu principal RMN / Séquence, qui affiche le texte de la séquence (dans le mémo) et ses paramètres (dans le TStringGrid).

La variable "sequenceur" est définie selon le type typseq de compilRMN.pas, elle contient tous les paramètres de la séquence et est rappelé/sauvé dans *.par avec chaque séquence. A chaque lecture/ecriture de sequence, on detruit le précédent pointeur pour en créer un nouveau.

Définition des gains pour l'emission dans gain_emission utilisé ensuite par compilRMN.pas.

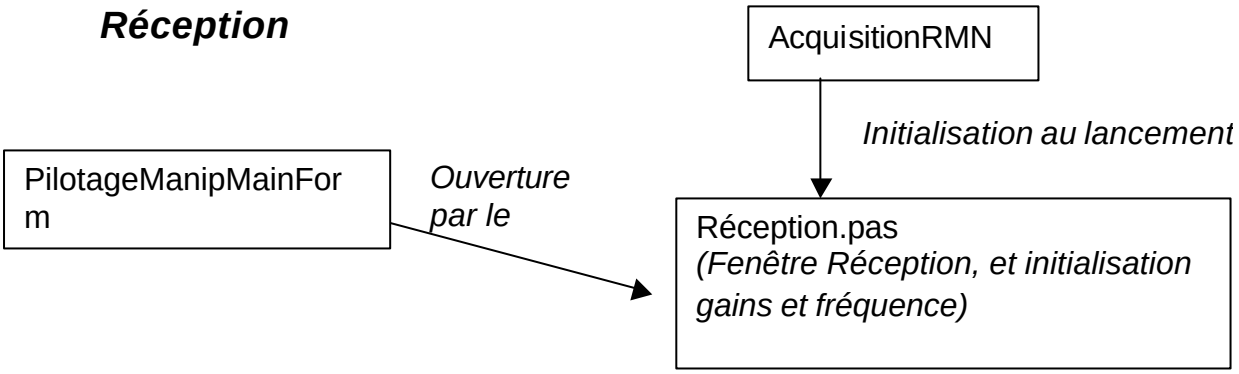
Le bouton "lancer" (qui a deux états stopped et running) appelle les fonctions de compilRMN.pas.

CompilRMN.pas

Définition de la structure des paramètres sequenceur (=typseq), ainsi que du type gain_emission.

Fonctions et procédures pour compiler et lancer les séquences. Appeler par Parseq.pas. Remarque : dans compilo, j'ai enlevé la possibilite de "gains préprogrammés divers" (syntaxe GADA) associés dans la version précédente àune autre série de paramètres commençant par D. Utile ??

Réception



Il faudra prévoir d'initialiser la réception au lancement.

- J'ai mis dans *réception.pas* les procédures de l'ancien *compil.pas* qui concernent la réception : initfreq, initfreqrec, prgrec (rebaptisé change_gainrec)
- Je n'ai pas repris ce qui concerne "l'ancienne réception" (initphase et initgain), ni initem et gainrec qui n'étaient pas utilisés.
- Il n'y a pas de pointeurs définissant les paramètres réception parce qu'ils ne sortent pas de la fenêtre.

- Pour l'instant : 1 seule fréquence utilisée
A terme, rajouter fréquence différente pour la réception , 2ème fréquence irradiation

Changement programmes principaux

- Dans PilotageManipMainForm.pas
  - Ajout dans le menu d'un bloc acquisition RMN lié au fichier configGAGE

- Dans PilotageManip.dpr
  - Certaines fenêtres non utilisées mises en remarque

- Dans l'interpréteur : plusieurs remarques

Dans les parties assembleurs (16 bits vs 32 bits)

- Le registre ax doit être appliqué àune variable de 16 bits : remplacer les 'integer' (maintenant 32 bits) par 'smallint'.
Pour al (8 bits) utiliser des 'shortint'.

- L'instruction Port n'est plus défini (ou porte un autre nom)
Equivalent assembleur de : port[ctrl_sequenceur]=$70

```
asm
mov dx,ctrl_sequenceur
mov al,$70
out dx,al
end
```

Ajout de graphes dans une fenetre :
Recopier le perbox et le combobox d'un graph existant.
Appeler le perbox perbox5 (si il y en a deja 4) et combobox : cocourbe5
Mettre dans propriete de perbox5 le spectre àla valeur 5
Changer dans le code :
Partout où il y a cocourbe, en ajoutant une ligne similaire pour le 5eme
Partout où il y a perbox
Dans procedure TGraphWindowNMRform.FormCreate(Sender: TObject);
La boucle for i:=1 to 5 au lieu de 4 do ga[i]:=1;
Et ajouter des couleurs :

```
coul[1]:=$000000;coul[2]:=$0000FF;coul[3]:=$FF0000;
coul[4]:=$007F00;coul[5]:=$FFFF00;coul[6]:=$FF00FF;
coul[7]:=$00FFFF;coul[8]:=$7F0000;
coul[9]:=$7F007F;coul[10]:=$00FF00;coul[11]:=$00007F;coul[11]:=$007F7F;
```

Dans procedure TGraphWindowNMRform.tfrspectre(i:integer;courbe:integer);
Ajouter les bonnes conditions àcelles existantes
Dans procedure TGraphWindowNMRform.changegain(var msg:tmessage);

```
begin
  if (msg.wparam<=5 au lieu de 4)and (msg.wparam>=1) then
```

Initial setup when installing the program

1. installer Delphi 4 (pas besoin d'installer intelservbase) n°serie 200-000-9552 8Dx2-5fx0
2. copier sur disque dur les programmes de Fermon dans un directory c:\newspectro (éviter les sous sous sous repertoires, ca fait planter la database ensuite).
   Il doit y avoir :
   \pilotagemanip qui contient \dataBaseManip et \sequences
   \programmes qui contient \composants qui contient \editpack, \fftacquis, \parser10, \unites, \graphxy
   \seq
   \sequences
   \spectro
3. depuis l'explorateur, verifier que ces programmes ne sont pas en lecture seule (se placer sur les fichiers et regarder propriété)
4. créer un répertoire c:\exec
5. mettre ieee_32m.dll dans le repertoire c:\windows et gage_drv.dll dans windows\system et gagescop.inc dans windows
6. lancer Delphi 4
7. cliquer Composant\Installer un composant\Dans nouveau paquet, mettre dans Nom du paquet "comprmn" puis dans Nom d'unité, mettre le composant grille.pas (qui se trouve dans programmes\composant). Puis confirmer l'installation du composant.
8. cliquer Composant\Installer un composant\dans paquet existant, et choisir àchaque fois une nouvelle unité et l'installer : installer ainsi multimemo.pas; \xygraph\xyholder.pas;
   \xygraph\xygraph.pas;\xygraph\xyreg.pas;\parser10\parser10.pas; fftacquis\DSXFastFourier.pas; \unites\EditReal.pas; unites\controlesDivers.pas; comrs232.pas (note : on peut en installer plusieurs àla fois en les cliquant sous un meme directory).
9. aller dans la fenetre Paquet comprmn.dpk (qui se trouve derrière), puis dans Options\Repertoire Conditions puis ajouter c:\exec aux lignes Sortie, Sortie de l'unité, chemin de recherche, repertoire destination BPL.
10. compiler (ctrl+F9)
11. aller dans projet\option\repertoire condition puis ajouter c:\exec aux lignes Sortie, Sortie de l'unité, chemin de recherche, repertoire destination BPL.
12. refermer Delphi (ne pas sauver les modifs de unit1, mais sauver comprmn)
13. lancer Borland\Common files (ou Borland Share)\Bde\Bdeadmin.exe
14. Cliquer Objet\Nouveau puis OK, puis renomer STANDARD par le nom databasemanip. Puis dans le champ de droite quand on clique sur databasemanip, specifier dans la case PATH comment acceder au directory databasemanip (qui se trouve dans pilotagemanip). Par exemple, il peut se trouver en C:\users\julien\spectro delphi\programme Fermon mai 2001\PilotageManip\dataBaseManip.
15. Cliquer Objet\Appliquer puis OK.
16. Relancer Delphi 4, puis cliquer Fichier\Ouvrir Projet et choisir pilotagemanip\PilotageManip.dpr

Note : si l'erreur "ne trouve pas cowon.res" survient, verifier que {$R c:\res\owncon.RES} est enlevé de composants\unites\ControleDivers.pas
Note importante :
J'ai modifié : dans comrs232 desactivation du timer en mettant en commentaire : {if not (csDesigning in ComponentState) then FNotifyWnd := AllocateHWnd(TimerWndProc);}

Dans Xygraph.pas : dans la procedure getmouseX, getmouseY et getmouseY2 je protege la condition ou FM=0 pour eviter un bug àl'ouverture de la fenetre sur le modele:

```
with FXAxis do
   begin
   if (FM<>0) then
    begin
     if FLogging then
       Result := FMin*Exp((x - FDimensions.FLeft) / FM)
     else
       Result := FMin + ((x - FDimensions.FLeft) / FM);
    end
            else Result:=FMin*1.1; {ajout Alloul de cette ligne pour eviter la plantade sur FM}
   end;
```