


minispec mq-Series

- Writing ExpSpel Applications for the NF Series
User Manual
Version 001



Copyright © by Bruker Corporation

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means without the prior consent of the publisher. Product names used are trademarks or registered trademarks of their respective holders.

This manual was written by

AIC

© December 21, 2015 Bruker Corporation

P/N: E1400012

For further technical assistance for this product, please do not hesitate to contact your nearest BRUKER dealer or contact us directly at:

Bruker Corporation
Am Silberstreifen
76287 Rheinstetten
Germany
Phone: +49 721-5161-6155/+1 978-667-9580
E-mail: minispec.SLS@bruker.com
Internet: www.bruker.com

Contents

1	Introduction	5
2	Commands that have changed in ExpSpel	7
3	Changes in the minispec.exe Interface	9
4	Introduction to Digital Filters and Group Delay Points	11
5	Setting up the Digital Bandwidth	13
6	New ExpSpel Commands	15
6.1	The sig_offset() Command: sig_offset(Temporal_Offset[ms])	15
6.2	The Range Command: range(start,stop,status).....	18
6.3	The Phase Correction Command: ph0corr(start, stop, phase, status).....	19
6.4	The Compatibility Check: isCompat(App_version)	20
6.5	The nextpulses Statement	22
7	The Group Delay Points and Data Processing	23
8	The decim and the Sampling Rate Enhancement (irate)	25
9	The Determination of the Number of Group Delay points	27
10	Experiments with Multiple Acquisitions	29
10.1	Pulse Sequence with Two Acquisition Commands Separated by Delays	29
10.2	Pulse Sequence with N acquisition Commands, e.g. CPMG.....	31
11	Examples	33
11.1	A Simplified FID Application (fid_example).....	33
11.2	A Simplified FID-ECHO application (FID-ECHO_example)	36
11.3	A Simplified CPMG Application (cpmg_example).....	39
12	Background of the Transition from ND to NF	43
13	Contact	45
	List of Figures	47
	List of Tables	49

1 Introduction

In the 4th Quarter in 2012 Bruker BioSpin switched to a new minispec mq-Series electronics, the so-called NF electronics. The new electronics are located in the same electronics housing as it has been the case before for the mq-Series ND systems.

The version of the electronic unit can be determined by looking at the serial number, which is located on the back of the housing. The first two digits of the serial number determine the version, NFXXXX for NF electronics and NDXXXX for ND electronics. The XXXX is the four digit serial number for the unit. Another way to verify the electronic unit version is in the minispec.exe software. Once you are connected to the instrument the serial number is displayed in the bottom of the minispec.exe screen.

In contrast to the ND-series minispec, the minispec NF has a digital filter to avoid folding back of signals and introduction of too high noise levels when acquiring data using the ADI or ASD commands.

This document explains the use of the ExpSpel programming of the minispec pulse sequences and the proper use of the digital filter as implemented in Firmware 3.2.x , FPGA 1BA and minispec.exe 3.00.

2 Commands that have changed in ExpSpel

This section lists the commands that became obsolete, had their functions modified, or were completely removed from the ExpSpel library. Such changes must be considered when using the minispec.exe software version 2.80 or higher. For further information about these commands, see the Help Topics in the minispec.exe at **Help | Help Topics | Find**.

Command	Status
abw()	Not supported
get_abw	Obsolete
get_customized_for	New behavior
get_phn / get("PHN")	Obsolete
get_pulse_atten	Obsolete
pgp()	Obsolete
pulse_atten()	Inactive
set("PHN")	Inactive
sf()	Inactive
sgp()	Obsolete
st()	Not supported
off_comp	Inactive

Table 2.1: Commands that have changed in minispec.exe Version 2.80 or Higher

In the table above:

- *No longer supported* means the m.exe compiler no longer recognizes the commands and will display an error message when trying to run the application.
- *Obsolete* means the command will return a value which is not used anymore in the application;
- *Inactive* means the command is recognized by the m.exe compiler, but it will not introduce changes in the application/pulse sequence.

abw()

This parameter used to be accessible in the parameter table, but due to the fact that NF has no user accessible analog filter it has been removed in the minispec.exe version 2.80.

get_abw / get_phn / get("PHN")/ get_pulse_atten

The commands will return 0.

get_customized_for

In previous versions of minispec.exe this command read information from the firmware. In recent versions this command reads the information directly from the instrument settings, referencing the entry in **Instrument ID** (field inaccessible to users).

pgp() / sgp()

These commands are obsolete. One should use the **nextpulses** statement in combination with the **ssp()** command. For more details see [The nextpulses Statement \[▶ 22\]](#).

pulse_atten()

This parameter used to be accessible in the parameter table as well as from the ExpSpel code, however it has been grayed out in the minispec.exe versions mentioned above.

Please note that the command **get("PAT")** is still functional, and returns the pulse attenuation written in the instrument settings.

sf()

This feature is not available in NF electronics.

st()

This command is no longer supported, to trigger gradient pulses one has to use the new command **nextpulses** in combination with **ssp()**. For more details see [The nextpulses Statement \[▶ 22\]](#).

off_comp()

This parameter used to be accessible in the parameter table and the parameter section of the ExpSpel code, but it has been removed beginning with the minispec.exe version 2.80.

3 Changes in the minispec.exe Interface

In previous sections it was discussed that some parameters have become obsolete, and have been removed from the parameter table and from the instrument settings. In this section these changes in the minispec.exe interface will be highlighted.

The following figure shows the entries in the **Instrument Settings** for (a) the minispec.exe version 2.80 or higher; (b) the minispec.exe version 2.71 or lower. The new entries are marked with green dashed lines and the entries that have been removed with a continuous red line.

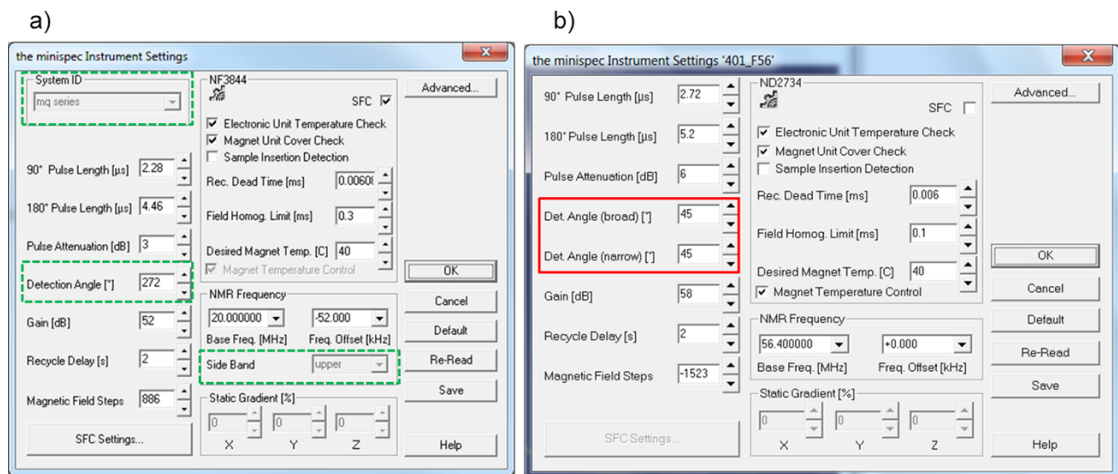


Figure 3.1: Modifications in the Instrument Settings.

The new entries: **System ID** and **Side Band** are not accessible to the user.

The figure below shows the **Parameter Table** in the minispec.exe: (a) version 2.80 and higher; (b) version 2.71 and lower.

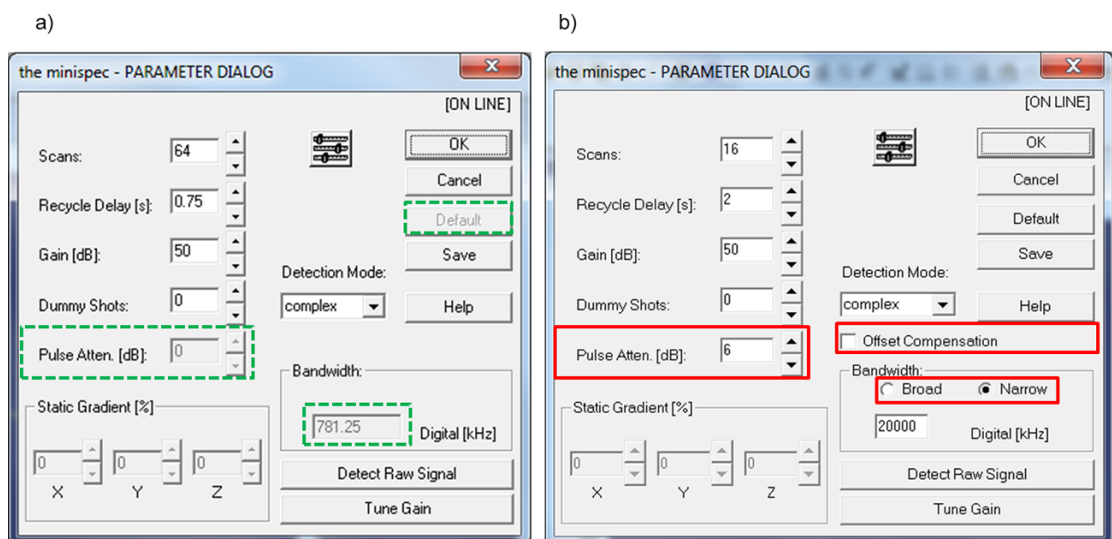


Figure 3.2: The Parameter Table in the minispec.exe

Changes in the minispec.exe Interface

As discussed in the previous section, the command `pulse_atten()` became inactive and because of that the respective entry in the *Parameter Table* has been grayed out. Moreover, one should define in the *ExpSpel* code the Digital Bandwidth to be used (*dbw*), that is why this field is grayed out in the *Parameter Table*, however this entry is updated once the application is loaded.

The following figure shows the **Settings Viewer** in the minispec.exe: (a) version 2.80 and higher; (b) 2.71 and lower. The main differences are highlighted.

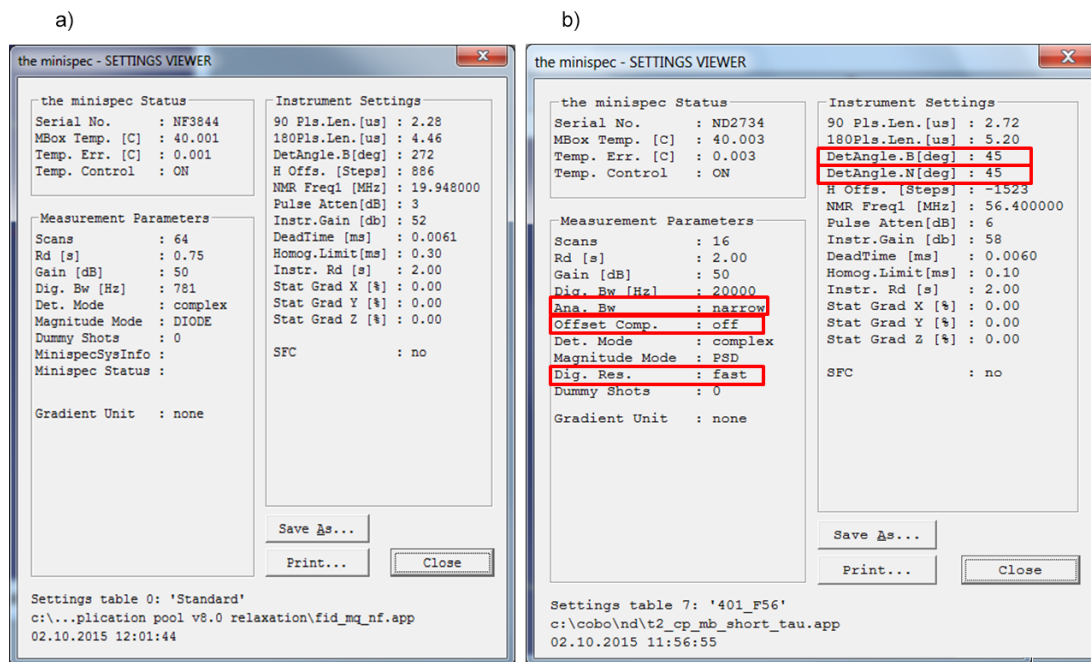


Figure 3.3: The Settings Viewer in the minispec.exe

4 Introduction to Digital Filters and Group Delay Points

On NF-series one can select the filter digital bandwidth (*dbw*) among several predefined values, using the following expression:

$$dbw(kHz) = \frac{12500}{2^N} \quad \text{Equation (1)}$$

Where $N = 0, 1, 2, 3, \dots, 12$.

In ExpSpel applications, in order to have the units of the parameters in seconds and for the sake of simplicity, expression (1 [▶ 11]) is replaced with:

$$swh(Hz) = \frac{12500}{decim} * 1000 \quad \text{Equation (2)}$$

Where *decim* = 1, 2, 4, 8, ..., 4096. As will be discussed in [Setting up the Digital Bandwidth \[▶ 13\]](#), the parameter *decim* can be changed in the ExpSpel code, so the user can set the desired *dbw* (*swh*).

Another important point regarding the digital filters is that the sampling rate of the NMR signal (also referred to as dwell time, *dw*) is directly related to the filter digital bandwidth:

$$dw(s) = \frac{1}{swh} = \frac{12500}{decim} * 1000 \quad \text{Equation (3)}$$

Moreover, as only discrete values of *dw* are possible, one has to make sure that the desired acquisition time (*aq*) is an integer multiple of the *dw* selected:

$$aq(s) = td * dw \quad \text{Equation (4)}$$

Where *td* is the number of points that the acquisition will have.

The last important point to stress regarding the digital filters is the presence of the group delay, which is a natural consequence of using digital filters. In terms of NMR signal, this can be visualized as a certain number of points (*grpDly*) with the intensity close to zero in the beginning of the data acquisition. After the group delay, the signal is comparable to the one obtained in the ND-series TD-analyzer. Consequently, the digital filter (by the group delay) leads to an apparent delay (*toffset*) in the signal, calculated by:

$$toffset(s) = grpDly * dw \quad \text{Equation (5)}$$

Physically the signal occurs at the same time as usual (ND), as this delay is just a data processing artifact related to the digital filter. There are new commands in ExpSpel used to compensate this apparent delay, which will be discussed in [New ExpSpel Commands \[▶ 15\]](#).

The next figure shows a free induction decay acquired using a NF-mq20. In this example one can see the group delay points marked in the red box at the beginning of the acquisition.

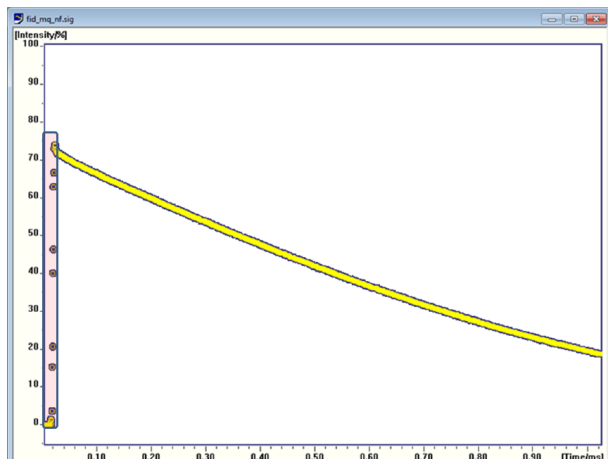


Figure 4.1: Example of NMR signal measured using the digital filters. Inside the region marked in red one can visualize the group delay points.

The following table summarizes the topics discussed in this section, showing how one can set the value of the digital bandwidth and dwell time by selecting *decim*. Note that the number of group delay points depend on the digital bandwidth selected.

<i>decim</i>	<i>dbw</i> (kHz)	<i>dw</i> (us)	<i>grpDly</i>
1	12500.00	0.08	35
2	6250.00	0.16	18
4	3125.00	0.32	33
8	1562.50	0.64	17
16	781.25	1.28	17
32	390.63	2.56	14
64	195.31	5.12	14
128	97.66	10.24	13
256	48.83	20.48	13
512	24.41	40.96	12
1024	12.21	81.92	12
2048	6.10	163.84	12
4096	3.05	327.68	11

Table 4.1: Allowed values for *dbw* and *dw* on NF-series minispec

For users who program their own applications, the syntax of *dbw* is still the same as before. However in NF one should use the allowed values for this parameter, given in the table. Examples of applications and how to define the digital bandwidth (*dbw*) can be found in the chapter [Examples \[33\]](#).



The values for *grpDly* written in the table above are specifically for firmware version 3.2.x and FPGA 1BA. Other combinations might have different values for *grpDly*.

5 Setting up the Digital Bandwidth

Most of Bruker's standard application uses *decim* of 16 as default value, resulting in a bandwidth of 781.25 kHz, matching the characteristics of all probes offered by Bruker and also covering the spectral line width of most of the samples of interest.

In standard applications it is also possible to change the digital bandwidth by changing the value of *decim*. To do so, open the ExpSpel editor and using search (**Ctrl + F**) for **Customize**. Typically one will find the following:

```
#-----Customize your filter settings? -----#
Decim = 16; #digital bandwidth of the filter is 12500kHz/decim,
decim=2^N #
irate = 4;      #interpolation of the fid data, 2^N#
#-----Customize your filter settings? -----#
```

In the ExpSpel code the user can find this box where he can customize the *decim* used for the experiments.

One can change *decim* to any of the values given in the table above, however one must be aware of the consequences of changing this parameter:

- The bigger the *decim* is, the narrower the digital filter is, therefore the noise levels will decrease, since the filter is cutting off higher frequency noises from the NMR signal that are not related to the actual signal of interest.
- One has to be sure that the filter digital bandwidth fits both: the probe bandwidth and the sample characteristics, i.e. the spectral line width, which can be estimated by the inverse of the transversal relaxation time (T_2^*).
- In principle, the bigger the *decim* is, the longer the dwell time is (slower sampling rates). This point will be further discussed later in the chapter [The *decim* and the Sampling Rate Enhancement \(*irate*\)](#) [25].

Moreover, often *decim* is defined in more than one part of the code, therefore one has to make sure to modify *decim* wherever it is defined in the code. The definition of *decim* is always done as depicted in the figure above, therefore keep searching for **Customize** and changing *decim* to the same value in all the boxes found.

Setting up the Digital Bandwidth

The following figure shows the behavior of the signal to noise ratio as function of the decimation for a 10R probe.

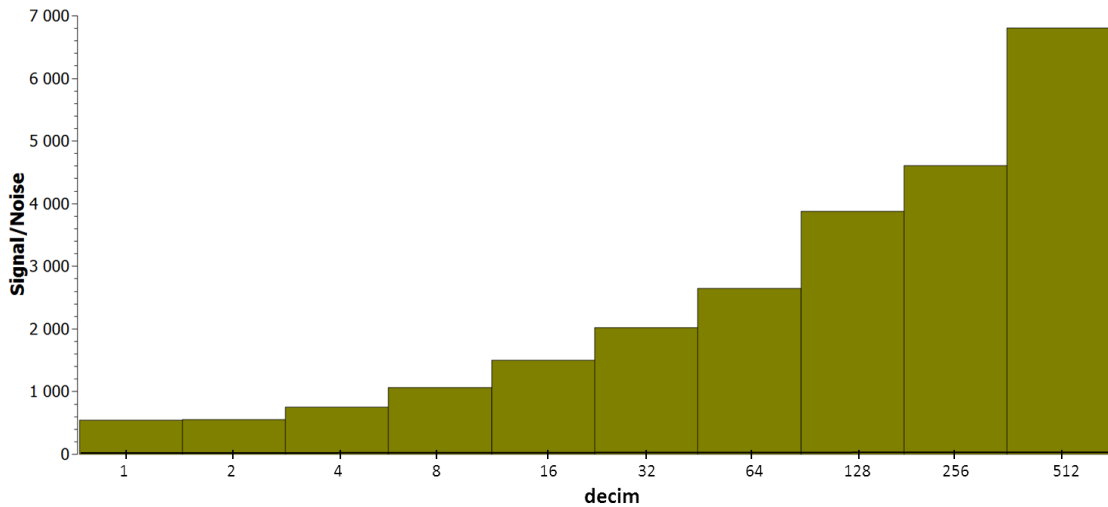


Figure 5.1: Signal/Noise Ratio as Function of decim.

This result can be translated in terms of the time necessary to achieve a certain *Signal/Noise* ratio (which is directly related to the standard deviation for repetitive measurements). Consider for instance that the desired *Signal/Noise* ratio is 7000. The time necessary to achieve this value for each decimation is illustrated the figure below:

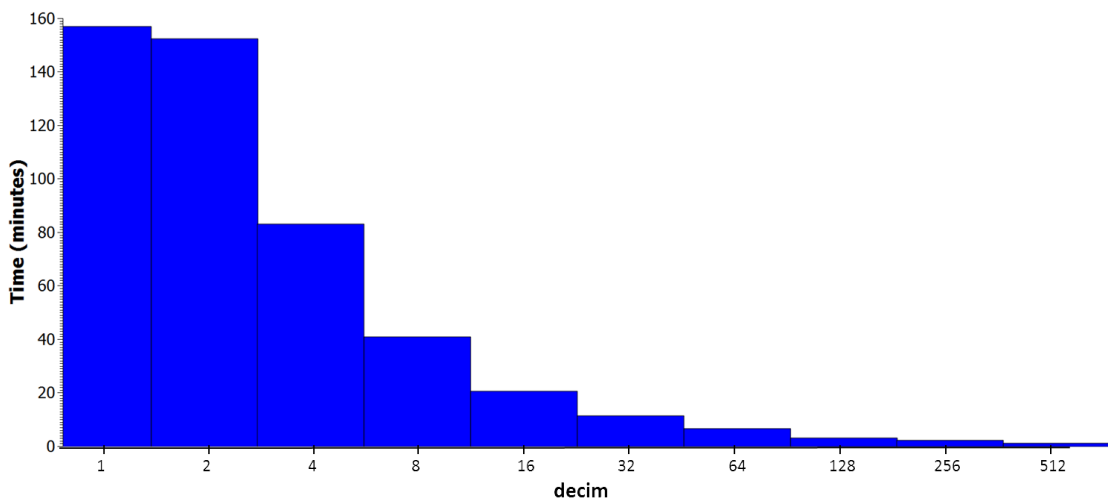


Figure 5.2: Time Required to Achieve the Signal/Noise Relation of 7000 for Several Values of decim.

Details on how to decide which decimation to use and its implications, and how to make use of new features present in NF electronics, will be described in [The decim and the Sampling Rate Enhancement \(irate\) \[25\]](#). It is worth pointing out that the decimation should be selected in order to fit the hardware features (magnet, probe etc.) and sample/experiment features.

6 New ExpSpel Commands

For convenience, there are new commands in ExpSpel which can be used to make the NMR signal be more similar to ND series minispec systems and to prevent non-compatible applications from being run. In this section the new commands are listed and explained.

6.1 The `sig_offset()` Command: `sig_offset(Temporal_Offset[ms])`

As discussed in [Commands that have changed in ExpSpel \[7\]](#), the existence of the group delay points results in an apparent delay in the NMR signal. To exemplify the temporal offset caused by the digital filter, consider the result illustrated below, which shows the free induction decay measured using a 10R probe. The group delay points are marked in the highlighted square, whose determination will be discussed in [The Determination of the Number of Group Delay points \[27\]](#).

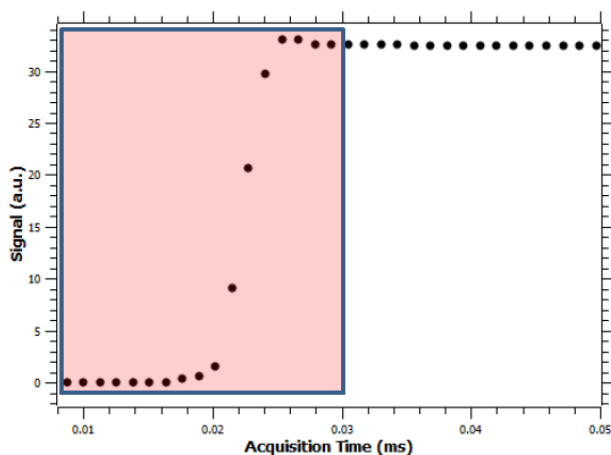


Figure 6.1: Free Induction Decay Obtained Using a 10R Probe, 90 Degree Pulse of $2.7\mu\text{s}$ and $RDT = 6.08\mu\text{s}$.

When the temporal axis is triggered before the 90° pulse, it is expected that the NMR signal will appear at $p90+RDT$, where $p90$ is the length of the 90° pulse and RDT is the Receiver Dead Time. For the experiment illustrated above, the probe has $p90 = 2.7\mu\text{s}$ and $RDT = 6.08\mu\text{s}$, therefore expectantly the NMR signal should start at $8.78\mu\text{s}$.

However, as illustrated in the figure, at $8.78\mu\text{s}$ the first point acquired corresponds to one of the group delay points and the actual signal appears only after all the group delay points, in this case around $30\mu\text{s}$.

This temporal offset is especially important in applications that use the temporal axis to make calculations, by using `sig_mean` or `mdt_mean` commands for instance, mainly for samples which has considerably sharp echoes, since the user has to keep in mind that there will occur an apparent temporal offset of the NMR signal acquired.

As an example, consider a Hahn Echo experiment performed in the mq-ProFiler instrument. This instrument has been selected due to its low magnetic field homogeneity, leading to considerably sharp echoes, therefore this can be considered a worst case scenario.

Usually the user desires to correlate an average around the echo top to some characteristic of their sample, and based on the delays used in the pulse sequence he calculates averages of the NMR signal around a specific temporal position of the signal. If the user does not consider the existence of the group delay points, the user might average the NMR signal in the *wrong region*. This is exemplified in (a) below, where the offset caused by the filter was not taken into account and the region marked with a cyanic line represents the region used to average the NMR signal.

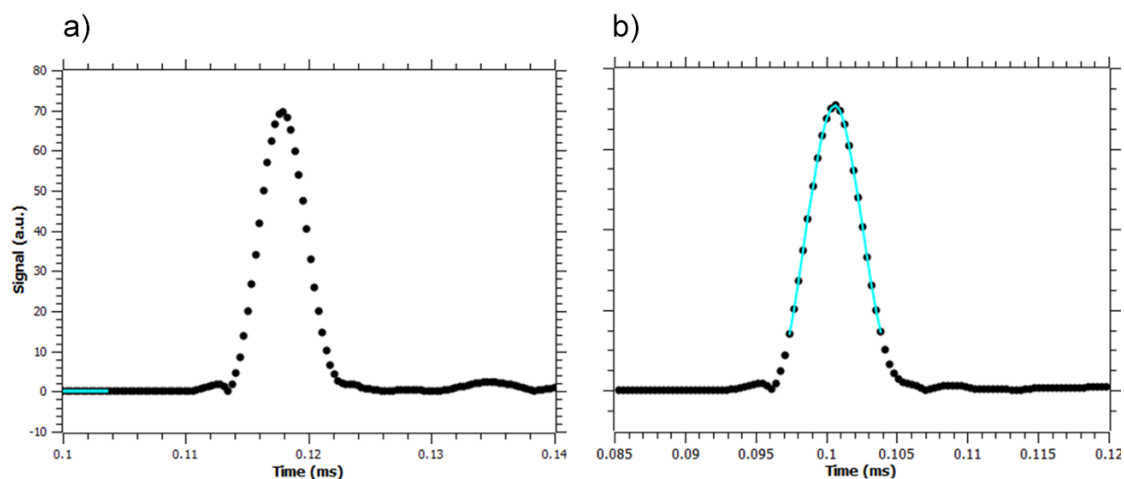


Figure 6.2: Solid Echo Signal Measured in the mq-ProFiler

In the figure above: (a) is the offset caused by the group delay is not considered; and (b) is the correction for the signal offset is done. Marked in cyanic is the region used for the data evaluation in both cases, being the center of this region the theoretical position of the echo top when the effects of the filter are not considered.

In order to easily correct the temporal offset introduced by the digital filter, Bruker has introduced a new command: **sig_offset()**. This command introduces an offset [ms] in the temporal axis given by the argument in the command.

As one will notice in Bruker standard applications, before the *measure* statement there is often in the code:

$$\text{sig_offset}(-\text{toffset}) \qquad \text{Equation (6)}$$

Where *toffset* is a hard coded variable, depending on the number of group delay points and the dwell time, being calculated according to [Equation \(5\)](#) [11].

This command can be added with the proper argument into the application (more specifically before the statement *measure*) used to obtain the result displayed in (a) in the figure above. Without any further changes, one gets the result displayed in (b) above, in which the echo top is exactly in the expected position of the acquisition window's time axis (if no digital filters were present/the one expected in ND instruments).

Commands sensitive to sig_offset()

One should keep in mind that in ExpSpel there are 2 main groups of commands for data processing, one which uses the raw data for the calculations (*mdt_*) and another which uses the displayed data for the calculations (*sig_*).

The command *sig_offset* affects only the processed data, keeping the raw data unchanged. Therefore, in the data evaluation one should use *sig_* commands to take into account the effects of the *sig_offset*.

The table below lists some of the most important commands that are related to the data processing and data storing.

Command: displayed data	Command: raw data
<i>sig_abscissa</i>	<i>mdt_abscissa</i>
<i>sig_ordinate</i>	<i>mdt_ordinate</i>
<i>sig_swap</i>	<i>mdt_swap</i>
<i>sig_max</i>	<i>mdt_max</i>
<i>sig_min</i>	<i>mdt_min</i>
<i>sig_mean</i>	<i>mdt_mean</i>
<i>sig_points</i>	<i>mdt_points</i>

Table 6.1: Examples of sig_ and mdt_ Commands

The function of these commands and many others can be found in the minispec.exe help, at **Help | Help Topics | Find**.

The physical meaning of the temporal offset

One should consider the temporal shift due to the digital filter simply as one offset in the x-axis, and not as an additional delay, for example to be summed to the receiver dead time.

Triggering the temporal axis in the middle of the excitation pulse

In ND and NF electronics, the temporal axis of the NMR experiment is triggered by the first pulse, being the time **0** set right before such pulse. In both cases it is possible to redefine a starting point of the temporal axis by using the command **cta**. However, in NF one can do further modifications in regards to where in the pulse sequence the temporal axis is *virtually* triggered by using the command **sig_offset()**. For example, in combination with the temporal offset caused by the group delay points, one can also subtract half of the excitation pulse, whereas virtually the temporal axis will be triggered in the middle of the excitation pulse: An example of such a procedure is shown in the chapter [Examples \[▶ 33\]](#).

6.2 The Range Command: `range(start,stop,status)`

As described before, the group delay points appear at the beginning of each NMR acquisition. The range command has been created to *hide* them and make the NMR signal more familiar for ND's users. This command defines the temporal region to be displayed on the screen.

Syntax: `range(start [ms], stop [ms] , status [TRUE/FALSE]);`

Where: start/stop is the temporal starting/final point to be displayed in the result window in milliseconds; and status can be TRUE (ON) or FALSE (OFF).

This command is usually used before the measure statement, so the result displayed in the screen has the temporal range defined by the range command.

Take for example the simplified *FID* application written in [A Simplified FID Application \(fid_example\)](#) [33]. By setting the status to *FALSE* (left figure below) one can see that the temporal axis starts at the first acquired data point, which is a group delay point, and ends at the last acquired data point. On the other hand, using the **range** command, i.e., setting its status to *TRUE* (right side below) one can select to display only the region of interest.

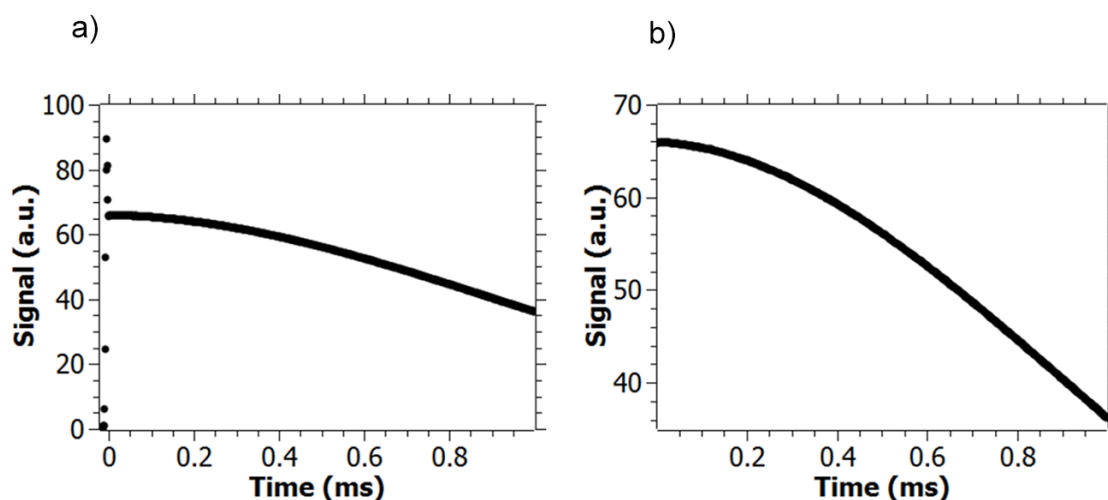


Figure 6.3: Using the Range Command. (a) Status set to *FALSE*; (b) Status set to *TRUE*.

Copying and pasting results

Please keep in mind that if you use **Ctrl + C** to copy the NMR data from the screen, only the displayed data in the screen will be copied.

Zooming out the displayed window

Reset the displayed region in the NMR data by right clicking in the window with the result (.sig). This will zoom out the axis and all data will be redisplayed.

6.3 The Phase Correction Command: `ph0corr(start, stop, phase, status)`

In ND electronics, one could set the analog bandwidth either to *broad* or *narrow*, and based on this should adjust the detection angle (broad or narrow) in the instruments settings.

However, it is known that changes in the receiver gain might lead to changes in the values of the detection angles, forcing the user to have to double check whether the values used for the sample A still hold for sample B if different receiver gains were used.

In NF electronics there is only one detection angle and there is an ExpSpel command to automatically calculate the detection angle, avoiding the need to set it beforehand. Regardless of the receiver gain used, a phase-corrected signal will be displayed.

Syntax: `ph0corr(start [ms], stop [ms], phase[deg], status[ON/OFF]);`

Where **start/stop** defines the starting/final temporal position to be used for the calculations, **phase** is the desired/expected phase in degrees and **status** is *ON/OFF*. It is recommend to use this command after the `signal_offset` command and before the `measure` statement, an example can be found in [A Simplified FID Application \(fid_example\) \[p 33\]](#).

The command will be executed for every scan in the experiment, showing the signal with the desired phase. Moreover, the raw data is also stored and it is used for the determination of the detection angle for subsequent scans, minimizing the effects of noise in the evaluation and allowing one to use phase cycling. The command is active only when the detection mode is set to complex.

Commands sensitive to `ph0corr()`

Similarly to the command `sig_offset`, the command `ph0corr` affects only the processed data, keeping the raw data unchanged. Therefore, in the data evaluation or when saving the data, one should use `sig_` commands to take into account the effects of the `ph0corr`.

Defining the phase

In principle one could use any phase from 0 to 359 degrees as argument for this command. What defines the phase to be selected is the expected NMR signal.

For instance, in the FID application one would expect a positive maximum of real signal and a minimum (close to zero) imaginary signal. Such relation defines a phase equal to 0 degrees.

On the other hand, in the inversion-recovery experiment for the first points in the inversion-recovery curve, one would expect a negative maximum of real signal and a minimum (close to zero) imaginary signal. Such condition defines a phase equal to 180 degrees.

The following illustrates how the phase (ϕ) is defined:

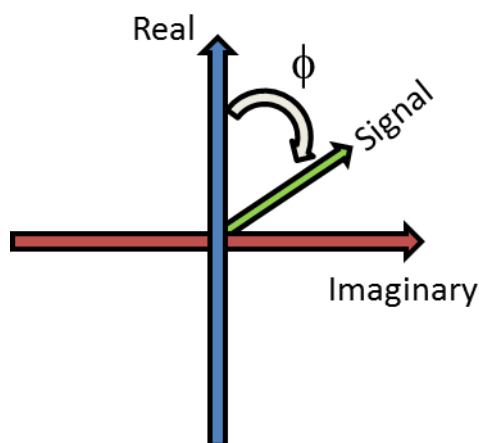


Figure 6.4: Definition of the Phase of the NMR Signal

Using the `ph0corr` command with loops (`while`)

There are applications that use `while` loops to run the application several times for different parameters. For instance, in the inversion-recovery application the delay between the first 180 degrees pulse and the 90 degrees pulse is changed automatically by a `while` loop.

In this experiment there is a flip in the phase of the NMR signal, starting at 180 degrees and later on flipping to 0 degrees. Therefore it is clear that one should make the phase calculation for only one of the delays, either the first or the last one (and select the phase accordingly).

To do so, one can create an auxiliary integer variable, herein named `first_run`, and set it to 1 for instance. Then, inside the `while` loop, you can check if the variable still has the initial value. If it does, then you can perform the phase correction and then change the value of the auxiliary variable. This way, the detection angle will be calculated only for the very first experiment, being this one used for the following experiments (inside the `while` loop):

```
...
int first_run;
first_run=1;
...
while (cnt<cnt_final)
...
if(first_run==1)
ph0corr(start,stop,phase,TRUE);
first_run=0;
endif;
...
endwhile
...
```

This procedure has been followed for any application that uses `while` loops, like the `t2_solid_echo_mq_nf` and the `t1_invrec_table_mq_nf` (inversion-recovery).

6.4 The Compatibility Check: `isCompat(App_version)`

In order to prevent users from accidentally using non-compatible applications in the new electronics, a new ExpSpel command has been created to check whether the application is compatible to the `minispec.exe` software installed, to the firmware and to the FPGA (Field Programmable Gate Array)

Whenever one application is loaded by the `minispec.exe`, the software verifies if the application contains the command `isCompat()`. If it does not have this command, the warning message displayed in the figure below will be displayed after trying to execute the application.

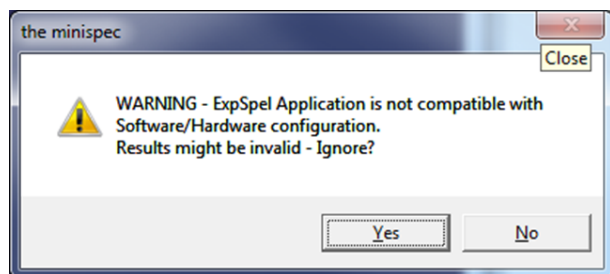


Figure 6.5: Error Message Displayed if the Application Does Not Have the `isCompat()` Command.

When the command is present in the application, it verifies if the installed software, firmware, FPGA and application in use are compatible to each other. If they are not compatible, a different warning message will be displayed after trying to execute the application:

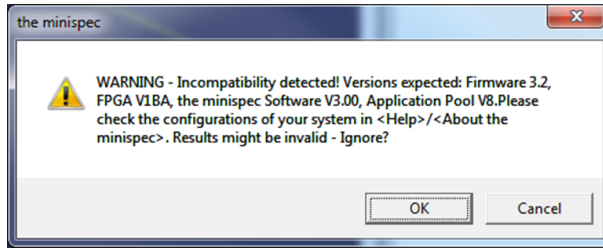


Figure 6.6: Error Message when Executing an Application which is not Compatible to the Firmware, FPGA, Software Installed.

In both cases the user can select to ignore the message, but Bruker strongly recommends making the necessary adaptations before proceeding.

Once the user has done the necessary modifications to ensure that their adapted application is compatible to his instrument, the following lines can be added in the **program measure** (and **program calibrate** if present):

```
#-----Compatibility check-V8.0-Rev0-FW320-----#
int compat, appv;
char comp_message1[500];
appv=8;
compat=isCompat (appv);

strcpy(comp_message1,"WARNING - Incompatibility detected! ");
strcat(comp_message1,"Please check the configurations of your system
in <Help>/<About the minispec>. Results might be invalid - Ignore?");

if (compat == 0)
print_line(CONFIRMBOX,comp_message1);
if (ESC) goto Escape;
else
print_line(RESULTBOX,"-----");
print_line(RESULTBOX," WARNING - Incompatibility detected! ");
print_line(RESULTBOX,"Please check the configurations of your system
in <Help>/<About the minispec>. ");
print_line(RESULTBOX," Results might be invalid.");
print_line(RESULTBOX,"-----");
endif;
endif;
#-----END-Compatibility check-----#
```

Such code is present in all standard applications, in general after the variable declaration in the **program measure** and **program calibrate**.

When to include such lines in the adapted applications

One should include these lines only after being absolutely sure the necessary changes have been done to make the application be completely compatible to the instrument's configuration.

Syntax and returned values

The syntax of the command is as follows: **isCompat(appv)**

Where **appv** must be an integer associated with the current version of the application pools released by Bruker. This version can be checked either in the standard applications or based on the name of the folders where the standard applications are. The version released when this manual has been written was 8 (minispec.exe version 3.0, FPGA 1BA and FW 3.2.3).

The command will return "0" if there is any incompatibility and "1" if the combination of firmware, FPGA, software and application is acceptable.

6.5 The nextpulses Statement

This statement is used in applications which have magnetic field gradients. The statement should be used between the *pulses* and *endpulses* statements, and it is used to define the pulse sequence for the gradient channel, running synchronous to the RF channel defined between *pulses*; and *nextpulses*;. See the example below (based on the application *self_diffusion_coefficient_mq_nf*)

```
pulses;
//Defining pulse sequence for the "channel A", i.e., no gradients//#
sp ( 90, 0, -1 );
sd ( dur1 );
sd ( dur2 );
sd ( dur3 );
sp ( 180, 90, -1 );
sd ( dur4 );
sd ( dur5 );
sd ( 2/5*dur6+500*dAdj+1000*grpDly*dw );
adi ( 1000*aq, td);
sd ( 2/5*dur6);
nextpulses;
//Defining pulse sequence for the gradient Channel//#
sd ( get("90P")/1000 );
sd ( dur1 );
ssp ( 1000*dur2, 0 ); # gradient pulse start #
sd ( dur3 );
sd ( get("18P")/1000 );
sd ( dur4 );
ssp ( 1000*dur5, 0 ); # gradient pulse start #
sd ( 2/5*dur6 +500*dAdj+1000*grpDly*dw);
sd ( 1000*aq );
sd ( 2/5*dur6);
endpulses;
```

The ssp command for applying gradients

The **ssp** command, when used between *nextpulses* and *endpulses* statements triggers the gradients using a PGU (Pulsed gradient Unit).

Syntax: *ssp(duration)*

The duration of the gradient is defined in μ s.

Overall duration of the pulse sequence in the gradient Channel

One must ensure that the overall duration of the sequence inside *nextpulses* (channel defining gradients) has the same overall duration (or is shorter) than the RF-channel (part of the sequence confined between the *pulse* statement and the *nextpulses* statement).

7 The Group Delay Points and Data Processing

As described in the previous sections, the group delay points always appear at the beginning of the NMR acquisition and if one wants to use ExpSpel functions based on the temporal axis for data processing, one should make sure to correct the temporal axis by using the command **sig_offset**.

However, there are applications where the whole NMR signal is used for fittings, for instance, *fid_mq_nf*, which uses in principle the whole NMR data for a mono or bi-exponential fitting. The user will notice that in cases like this, in Bruker standard applications, both the correction of the temporal axis is done using the **sig_offset** command and the group delay points are excluded from the raw data.

There are several possibilities of excluding the group delay points from the data processing. One possibility is to first correct the temporal axis using the *sig_offset* and then store only the points in arrays that come after the group delay points, based on the temporal axis values. As an example, refer to the simplified version of the *fid_mb* application, written in [A Simplified FID Application \(fid_example\) \[p 33\]](#).

In this application, the temporal axis is first corrected using the **sig_offset** command, which removes the temporal offset based on the group delay points, then triggers the temporal axis at the center of the first pulse. After this step, the first NMR data point (excluding the group delay points) should appear at $p90/2 + RDT$, where *p90* is the 90 degrees pulse length and RDT the Receiver Dead Time. The second step is to store the data starting from $p90/2+RDT$ into the arrays *x* and *y*, and the last step is to replace the displayed data with the one stored in the arrays.

Another possibility is to store the entire NMR data after using the *sig_offset* command. Then, using a *while* loop and a second set of variables (*x2,y2*), one can exclude the group delay points in the second set of variables, and later on replace the displayed signal with the data stored in *x2,y2*:

```
no = data_points( -1, -1 );
sig_abscissa(-1, -1, x ); #all pts included (also group Delay pts)#
sig_ordinate( -1, -1, y );
cnt=0;
no2= no-grpDly;
while(cnt<no2)
x2[cnt]= x[cnt+grpDly];
y2[cnt]= y[cnt+grpDly];
cnt=cnt+1;
endwhile;
replace_signal( x2, y2, no2);
```


8 The *decim* and the Sampling Rate Enhancement (*irate*)

As discussed in [Setting up the Digital Bandwidth \[13\]](#), one should select the digital bandwidth (*decim*) accordingly to the probe and sample characteristics. The results displayed in that chapter could mislead one to use extremely high decimations, since for those the signal to noise ratio is considerably better. However, the higher the decimation is, the narrower is the width of the digital filter, thus it is possible to filter out part of the NMR signal of some samples for some decimations. Another side effect of higher decimations is that, in principle, the dwell time (or sampling rate) is determined by the digital bandwidth, and the higher is the decimation the slower is the sampling rate. Both effects are visualized in the following figure:

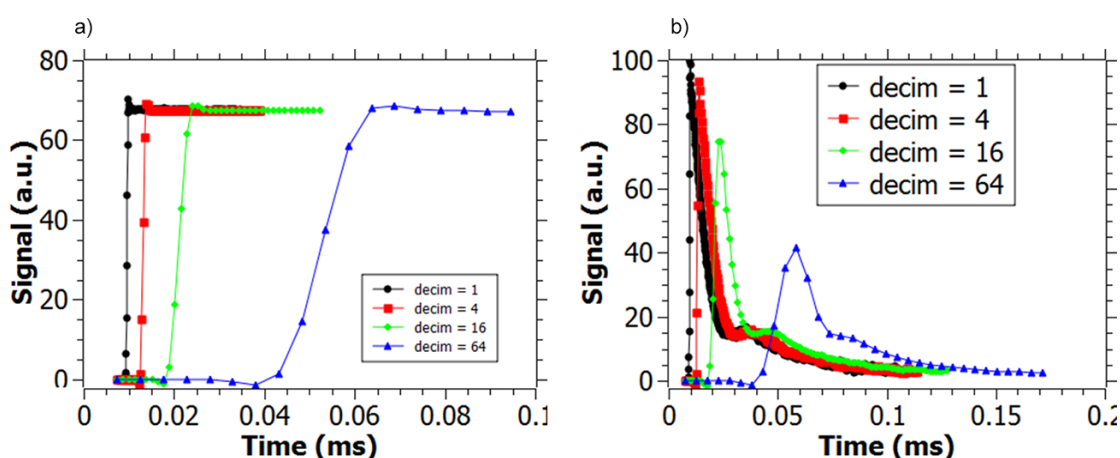


Figure 8.1: NMR Signal (FID) for Different Values of *decim*

The figure above shows the FID (including the group delay points and without correcting the temporal offset) for several decimations and two different samples: (a) the Daily Check Sample; (b) a Polymer with one component with very fast T_2 decay.

One important point is that on the left side of the figure the NMR signal has nearly the same amplitude for all decimations used, while on the right side the intensity is considerably reduced when the decimation used is higher than 16. This is directly related to the sample properties, whereas in the first case the sample was a liquid with reasonable T_2 values, in the second case the sample had a very short T_2 decay. It is recommended to adjust the decimation accordingly in order to prevent *cutting off* of the NMR signal coming from the more rigid components of the sample. Refer to [Setting up the Digital Bandwidth \[13\]](#) on how to change the decimation.

Another important observation is that in comparing the curves obtained in (a) in the figure, one can see that the sampling rate gets slower as the decimation increases, reducing considerably the amount of data points acquired. The same can be observed in (b), however this result could also mislead someone to use always lower decimations (e.g. *decim* = 1), which would result in fast sampling rates, but as discussed in [Setting up the Digital Bandwidth \[13\]](#), this would result in a digital filter bandwidth widely open, reducing considerably the Signal/Noise ratio.

In order to get both fast sampling rate (more data points per unit of time, lower decimations) and improved signal to noise ratios (narrow digital filter bandwidth, higher decimation), the *Sampling Rate Enhancement* has been developed, uncoupling the sampling rate from the digital bandwidth. There are different ways to code such features in ExpSpel. In the current

The *decim* and the Sampling Rate Enhancement (*irate*)

application Pool (V8.0), one hardcoded variable is responsible for increasing the sampling rate: *irate*. Whenever available in the standard applications, it appears in highlighted in the ExpSpel code, as shown in [Setting up the Digital Bandwidth \[13\]](#).

Syntax: *irate* is an integer variable assuming values of 2^N equal or lower than *decim*, i.e., $irate = \{1, 2, 4, 8, \dots, decim\}$. The resulting dwell time (dw') will be given by:

$$dw' [us] = irate * decim / 12.5 \quad \text{Equation (7)}$$

As an example suppose that one has selected *decim* = 16. This would lead to a dwell time of $1.28\mu s$ (without the Sampling Rate Enhancement). For *decim* = 16, one can select one of the following values for *irate*: {1, 2, 4, 8, 16}, resulting into dwell times of {1.28, 0.64, 0.32, 0.16, 0.08} μs , respectively. The following figure shows several experiments done for several values of *decim* and *irate*.

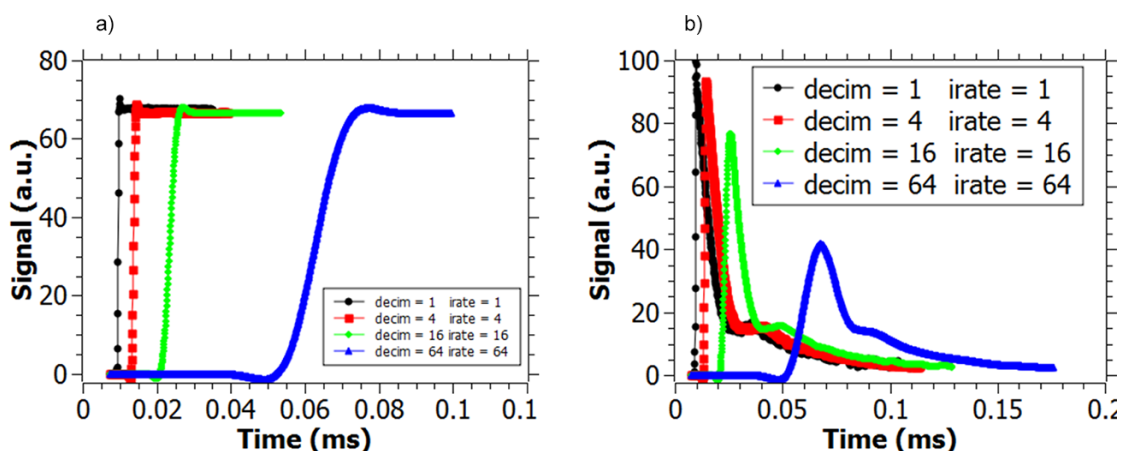


Figure 8.2: NMR Signal (FID) for Different Values of *decim* and *irate*.

In the figure above: (a) is for the daily check sample; (b) is for a polymer sample with one component with short T2 decay.

Comparing the signals displayed in the beginning and end of this chapter, one can see the main advantages in using the *Sampling Rate Enhancement*: better signal to noise ratios with fast sampling rates.

One important point is that when using *irate*, the number of points in the group delay also changes by $irate * GrpDly$, however the temporal offset is still the same:

$$toffset^{(s)} = grpDly' * dw' = (irate * grpDly) * \left(\frac{dw}{irate}\right) = grpDly * dw = toffset$$

Equation (8)

9 The Determination of the Number of Group Delay points

As discussed in previous sections, one should discard the group delay points before doing the data processing. For this reason, the number of group delay points for all available decimations have been determined and have been explicitly written in the ExpSpel code. It is important to emphasize that the number of group delay points is fixed for a given decimation, not depending on the application itself.

For the determination, the Daily Check sample (0% solid content) was used; this sample has only one main component of T_2 . For the NMR signal acquired, we have counted down how many points should be discarded until getting the signal from the sample without displaying filter artifacts. The figure below exemplifies it for 2 decimations.

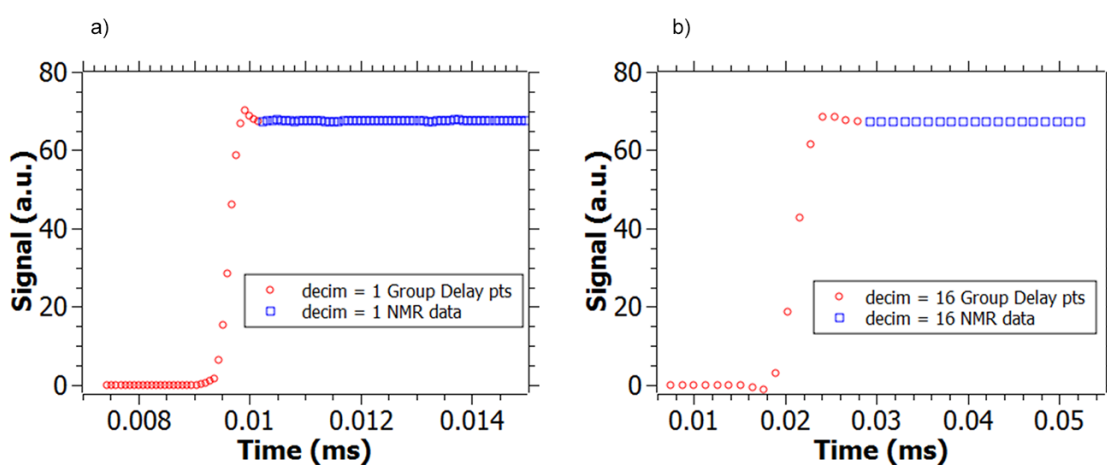


Figure 9.1: Number of Group Delay Points Determination: (a) decim = 1; (b) decim = 16.

In **a)** it is clear that the initial fast decay is an artifact from the digital filter, since the sample is characterized by only one T_2 value. However, as the filter becomes narrower, this decay becomes smoother, being under certain conditions hard to define what is the *actual signal* and what is an *artifact*, as illustrated in **b)**.

For most samples, discarding one or two points more should not matter, however for samples that have components with very short T_2 decay it might be relevant. Experienced users can make fine adjustments of the number of group delay points by looking for the *Group Delay table* in the ExpSpel code (usually there are multiple occurrences) and changing the values in these tables. All remaining calculations use these values.

10 Experiments with Multiple Acquisitions

When using multiple acquisitions in an application one must keep in mind that:

- The digital filter requires a minimum time to be reset to the initial condition, given by [Equation \(5\)](#) [[▶ 11](#)].
- When multiple acquisitions separated by delays are used, all of them will display the group delay points, moreover one should make sure to use delays between the acquisitions longer than the time that the filter requires to be reset to the initial condition.
- The offset caused by the digital filter must not be treated as a delay, since it is part of the acquisition time and shifts only the temporal position of the NMR signal display, not the signal itself.
- When back to back acquisitions are used (i.e., N consecutive acquisitions not separated by any kind delay) the filter is not reset to the initial condition and consequently only the group delay points from the first acquisition will appear in the NMR signal.

In this chapter the first three points will be in focus, first through a simple case in which only two acquisitions are performed will be discussed, e.g. a FID-ECHO-like application. And second, a case in which hundreds of acquisitions are performed, e.g. a CPMG-like application.

10.1 Pulse Sequence with Two Acquisition Commands Separated by Delays

As discussed above, in this case each acquisition will display the group delay points. In the chapter [A Simplified FID-ECHO application \(FID-ECHO_example\)](#) [[▶ 36](#)] one can find a simplified version of the FID-Hahn echo application, which will be used to exemplify the consequences of not making the correction in the temporal axis.

In this kind of application the user usually wants to make averages around the FID and Echo top, and based on these values establish correlations. The regions in the graphs marked in cyanic are the regions that the application will use for the calculations. In the figure **a**) below, when the correction in the temporal axis is not done (setting *toffset* to 0 in the application), part of the group delay points from the FID is used for the data evaluation. However, when the application acquires a considerable part of the echo, the region for the calculations in the Echo does not include the group delay points from the second acquisition. Furthermore, it is almost impossible to tell the difference between the regions around the Echo top used for the data evaluation on the **a**) side of the figure (not correcting the temporal axis) and on the **b**) side of the (correcting the temporal axis).

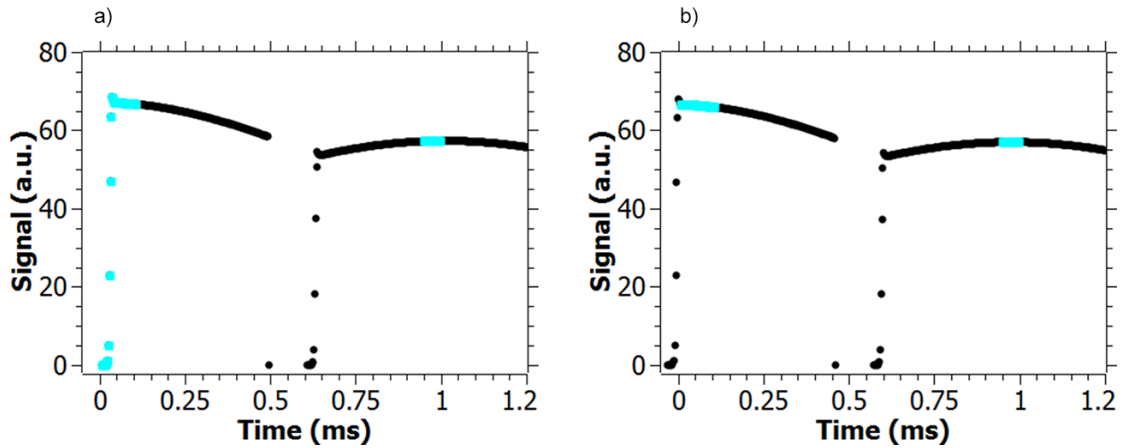


Figure 10.1: A FID-ECHO Experiment.

In the figure above, the cyanic line represents the region used for the data evaluations. (a) The shift in the temporal axis was not compensated. (b) The `sig_offset` command has been used to compensate the temporal offset introduced by the digital filter

The next figure shows a zoom around the Echo top from (a) and (b).

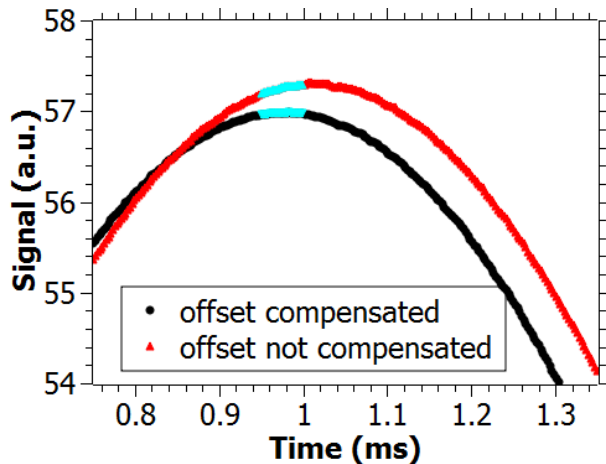


Figure 10.2: Zoom Around the Echo Tops Displayed from Previous Figure.

As one can see in the figure above, when not making the correction in the temporal axis, one will use for the calculations a region slightly shifted from the echo top, and the shift is given by [Equation \(5\)](#) [11].

Therefore the higher the *decim* (sharper digital bandwidth) is, the bigger the shift will be. This is why for this example $decim = 32$ was used, which leads to an offset of roughly $36\mu s$. Indeed, for samples with broad echoes this should not be a problem, as illustrated above, however for samples with sharp echoes this shift might result in not using the echo top at all. Moreover, if the acquisition starts right before the Echo top, one might end up using the group delay points for the calculations (like in the FID).

Another point worth discussing is how the correction in the temporal axis should be done when making multiple acquisitions. As one can see in [A Simplified FID-ECHO application \(FID-ECHO_example\)](#) [36], the procedure is exactly like the case where only one acquisition is performed. As pointed out in the beginning of this section, the group delay points are part of the acquisition itself, and should not be seen as additional delays. Therefore they do not have “accumulative effects” over the timings in the application. Thus, using a single `sig_offset` command with the proper argument will make all necessary corrections.

Finally, it has been remarked that the overall time between two consecutive acquisitions separated by any kind of delays (including pulses) should be longer than the time necessary to reset the digital filter. One can notice in the application which was used as an example in this section ([A Simplified FID-ECHO application \(FID-ECHO_example\) \[36\]](#)), there is a delay after the 180 degrees pulse of *rdt* (Receiver Dead Time) plus 0.1ms. This additional delay (0.1 ms) was used to ensure that the filter had enough time to be reset before starting the second acquisition. One could use a more elaborate way calculating the necessary delay, i.e. using **toffset**, which is exactly the minimum time required to reset the filter (see [Equation \(5\) \[11\]](#)).

The figure below shows the results that one gets running the application shown in [A Simplified FID-ECHO application \(FID-ECHO_example\) \[36\]](#) without the additional delay of 0.1 milliseconds.

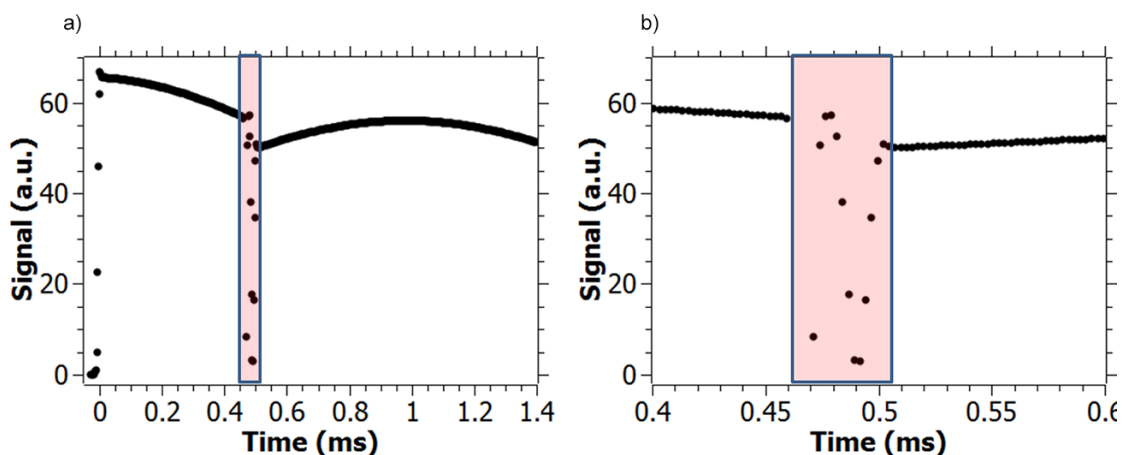


Figure 10.3: Two Acquisitions Performed with a Delay in Between Shorter than the Time Required to Reset the Digital Filter.

The region marked in red highlights the group delay points from the second acquisition (a) FID-ECHO; (b) Zoom in the marked region from (a).

For the settings used, the delay between the two acquisitions was roughly 10 μ s (when excluding the additional 0.1ms), being shorter than the time required to reset the digital filter for the decimation selected (decim = 32, toffset \sim 36 μ s). Therefore the filter is not completely reset before starting the second acquisition, consequently the beginning of the second acquisition will be a combination of *Pure Group Delay points* from the second acquisition and part of the signal from the first acquisition which is still in the digital filter. The resulting NMR signal can be seen in the figure above.

10.2 Pulse Sequence with N acquisition Commands, e.g. CPMG

There are applications where the user may be interested in acquiring thousands of echoes in a single scan, e.g. the *CPMG* (Car-Purcell-Meiboom-Gill) sequence. In these cases, one would use the *ASD* (Acquire Single Data point) command instead of the *ADI* (Acquire Data Incrementally), since the first can be used to acquire certain number of points around each echo top and display only the average of them. This saves CPU memory and consequently allows one to acquire tens of thousands of echoes. In NF (FPGA 1BA / FW 3.1.1 or higher) one can use both commands for this kind of experiment. The main changes in these commands will be described in this section.

As described throughout this document, the digital filter introduces the group delay points at the beginning of the acquisition of the NMR signal, and one has to take that into account when using *ASD* and *ADI* commands. This is especially relevant in pulse sequences similar to the *CPMG*, since the acquisition times are typically short and only one point is displayed for each acquisition of the train of echoes.

Experiments with Multiple Acquisitions

For users that intend to program their own applications it is recommend to use as basis the ProFiler applications (e.g. *t2_cpmg_table_mq_nf*). A simplified version of this application can be found in [A Simplified CPMG Application \(cpmg_example\) \[▶ 39\]](#).

In this application the number of points to be used in the acquisition is calculated based on the digital filter (*decim*) to be used and then the acquisition time is calculated (the acquisition time has a single allowed value for each decimation). Moreover, due to the digital filter the timings in the sequence will be asymmetric, i.e., the time before the echo acquisition (*pre*) is different from the one after the acquisition (*post*). The parameter that computes such asymmetry is *asdshift* (which is equal to *toffset*). In this kind of application (multiple acquisition commands in a single sequence) it is recommended to use a different group delay table, which can be seen in the following table.

<i>decim</i>	<i>dbw</i> (kHz)	<i>dw</i> (us)	<i>grpDly</i>
1	12500.00	0.08	48
2	6250.00	0.16	27
4	3125.00	0.32	27
8	1562.50	0.64	19
16	781.25	1.28	14
32	390.63	2.56	12
64	195.31	5.12	11
128	97.66	10.24	11
256	48.83	20.48	11

Table 10.1: Group Delay Table for Sequences with Multiple Acquisitions

Finally, the delays used in the sequence are recalculated to make sure that they are multiple of the temporal resolution of the instrument (20ns). The following figure shows the CPMG curve obtained using the application written in the [A Simplified CPMG Application \(cpmg_example\) \[▶ 39\]](#).

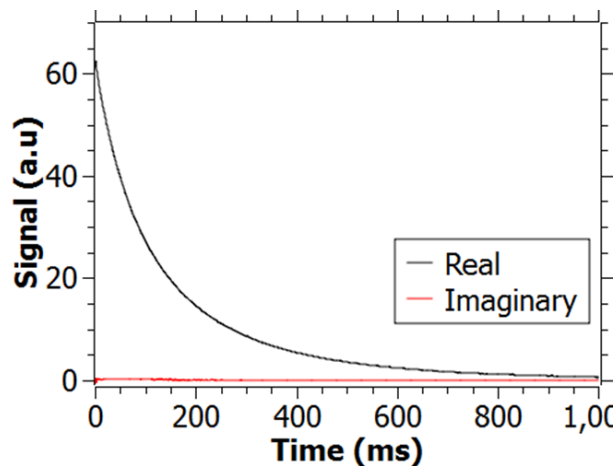


Figure 10.4: CPMG Results Showing the Real and Imaginary Components of the NMR Signal.

11 Examples

11.1 A Simplified FID Application (fid_example)

You can copy the code and paste it in a new ExpSpel program or ask us the application by e-mail.

```

program setup();
par;
scans (16);
rd (2.000000000);
gain (50);
det_mode("complex");
endpar;

program measure();

int no;
int tdFid, c0, grpDly, td, dispGrpDly;
int ph90[100], phrc[100],decim;
real x[32048], y[32048];
real duration;
real dw, aqFid, aq, rdt;
real p90, dur, swh;
real toffset;

#-----Compatibility check-V8.0-Rev0-FW320-----#
int compat,appv;
char comp_message1[500];
appv=8;
compat=isCompat(appv);

strcpy(comp_message1,"WARNING - Incompatibility detected! versions
expected: Firmware 3.2, FPGA V1BA, the minispec Software V3.00,
application Pool V8.");
strcat(comp_message1,"Please check the configurations of your system
in <Help>/<About the minispec>. Results might be invalid - Ignore?");

if (compat == 0)
print_line(CONFIRMBOX,comp_message1);
if (ESC) goto Escape;
else
print_line(RESULTBOX,"-----");
print_line(RESULTBOX," WARNING - Incompatibility detected!" );
print_line(RESULTBOX,"versions expected: Firmware 3.2, FPGA V1BA, the
minispec Software V3.00, application Pool V8" );
print_line(RESULTBOX,"Please check the configurations of your system
in <Help>/<About the minispec>. ");
print_line(RESULTBOX," Results might be invalid.");
print_line(RESULTBOX,"-----");
endif;
endif;
#-----END-Compatibility check-----#

```

Examples

```
#-----Customize your filter settings?-----#
decim = 16; #digital bandwidth of the filter is 12500kHz/decim,
decim=2^N #
#-----#

#-----V7.0-FW-3-1-0-----# #group delay table#
if (decim==1) grpDly=35; endif;
if (decim==2) grpDly=18; endif;
if (decim==4) grpDly=22; endif;
if (decim==8) grpDly=17; endif;
if (decim==16) grpDly=17; endif;
if (decim==32) grpDly=14; endif;
if (decim==64) grpDly=14; endif;
if (decim==128) grpDly=13; endif;
if (decim==256) grpDly=13; endif;
if (decim==512) grpDly=12; endif;
if (decim==1024) grpDly=12; endif;
if (decim==2048) grpDly=12; endif;
if (decim==4096) grpDly=11; endif;
#-----END-V7.0-FW-3-1-0-----#

#-----Defining the dwell time & digital band width-----#
swh=12500000/decim;
dw=1/swh;

par;
dbw(swh/1000);
endpar;
#-----#

#-----Phase Cycle:-----#
ph90[ 0] = 0; phrc[ 0] = 0;
ph90[ 1] = 180; phrc[ 1] = 180;
ph90[ 2] = 90; phrc[ 2] = 90;
ph90[ 3] = 270; phrc[ 3] = 270;
ph90[ 4] = REDO; phrc[ 4] = REDO;
#-----Phase Cycle End-----#

#-----application parameters-----#
duration = 1;
p90 = get("90P");
rdt = get("RDT");
#-----#

#-calculating number of pts for desired dwell time and acquisition
time-#
aqFid=duration/1000;
tdFid=trunc(aqFid/dw);
tdFid=tdFid+grpDly; #little trick to keep the acquisition with always
the same duration regardless the decim selected#
aqFid = tdFid * dw; #redefining the acquisition time: the closest to
the one desired #
```

```

#-calculating number of pts for desired dwell time and acquisition
time-#

pulses;
ssp (p90, ph90);
sd (rdt);
adi (1000*aqFid, tdFid, phrc);
endpulses;

#-----data pre-processing-----#
toffset=1000*grpDly*dw; #Temporal offset caused by the group delay #
sig_offset(-toffset-p90/2000); #Triggering the temporal axis at the
center of the excitation pulse#
range(p90/2000+rdt,p90/2000+rdt+1000*(aqFid-dw)-toffset, TRUE);
#adjusting the range of the window with the NMR data #
ph0corr(p90/2000+rdt,p90/2000+rdt+10*1000*dw,0,ON); #Correcting the
phase of the NMR signal #
#-----data pre-processing-----#

measure;

#-----Storing the NMR data into arrays-----#
no = data_points( p90/2000+rdt, -1 )-1;
sig_abscissa(p90/2000+rdt, -1, x ); #stores all NMR points excluding
group delay points (sig_offset used)#
sig_ordinate( p90/2000+rdt, -1, y ); #stores all NMR points excluding
group delay points (sig_offset used)#
#-----#

replace_signal( x, y, no ); #replace the data by the ones in the
array: definitive change#
label Escape;
return( TRUE );

```

11.2 A Simplified FID-ECHO application (FID-ECHO_example)

You can copy the code and paste it in a new ExpSpel program or ask us the application by e-mail.

```

program setup();

par;
scans (4);
rd (1.000000000);
gain (50);
det_mode ("magnitude");
dig_res ("high");
dummy_shots (0);
endpar;

int decim;
real swh;

#-----Customize your filter settings?-----#
decim = 32; #digital bandwidth of the filter is 12500kHz/decim,
decim=2^N #
#-----#

swh = 12500000/decim;
par;
dbw ( swh/1000 );
endpar;

return (TRUE);

program measure();

int decim,td1,td2,grpDly,phrc90[16],phrc18[16],ph90[16],ph18[16];
real acq1,acq2,dacq1,dacq2,toffset,dw,swh,te;
real p90,p18,rdt,strt1,strt2,stop1,stop2,WIN_FID,WIN_ECHO;

#-----Compatibility check-V8.0-Rev0-FW320-----#
int compat,appv;
char comp_message1[500];
appv=8;
compat=isCompat(appv);

strcpy(comp_message1,"WARNING - Incompatibility detected!");
strcat(comp_message1,"Please check the configurations of your system
in <Help>/<About the minispec>. Results might be invalid - Ignore?");

if (compat == 0)
print_line(CONFIRMBOX,comp_message1);
if (ESC) goto Escape;
else
print_line(RESULTBOX,"-----");
print_line(RESULTBOX," WARNING - Incompatibility detected!" );
print_line(RESULTBOX,"Please check the configurations of your system
in <Help>/<About the minispec>. ");

```

```

print_line(RESULTBOX," Results might be invalid.");
print_line(RESULTBOX,"-----");
endif;
endif;
#-----END-Compatibility check-----#

#-----Initialization of the parameters-----#
te=1; #echo time ms#
WIN_FID = 0.1; #Window for FID average#
WIN_ECHO = 0.1; #Window for ECHO average#
#----END--Initialization of the parameters-----#

#-----Customize your filter settings?-----#
decim = 32; #digital bandwidth of the filter is 12500kHz/decim,
decim=2^N #
#-----#

#---Defining the digital bandwidth-----#
swh = 12500000/decim; dw = 1/swh;
par;
dbw ( swh/1000 );
endpar;
#---Defining the digital bandwidth-----#

#-----V7.0-FW-3-2-0-----# #group delay table#
if (decim==1) grpDly=35; endif;
if (decim==2) grpDly=18; endif;
if (decim==4) grpDly=22; endif;
if (decim==8) grpDly=17; endif;
if (decim==16) grpDly=14; endif;
if (decim==32) grpDly=14; endif;
if (decim==64) grpDly=14; endif;
if (decim==128) grpDly=13; endif;
if (decim==256) grpDly=13; endif;
if (decim==512) grpDly=12; endif;
if (decim==1024) grpDly=12; endif;
if (decim==2048) grpDly=12; endif;
if (decim==4096) grpDly=11; endif;
#-----END-V7.0-FW-3-1-0-----#

toffset=1000*grpDly*dw; #Temporal offset caused by the digital
filter#

rdt = get_instr_param("RDT");
p90 = get("90P");
p18 = get("18P");
rdt = get("RDT");

#-----Acquisition parameters-----#
acq1= te/2 - rdt - p90/2000 - p18/2000; #desired FID acquisition
time#
td1 = trunc(acq1/(1000*dw)); #Closest number of pts for the desired
acq time#
dacq1=acq1-1000*td1*dw; #deviation desired - closest possible#
acq1=1000*td1*dw; #Acquisition time to be used #

acq2= 2*(te/2 - p18/2000 - rdt);

```

```

td2 = trunc(acq2/(1000*dw));
dacq2=acq2-1000*td2*dw;
acq2=1000*td2*dw;
#-----Acquisition parameters-----#

sig_offset(-toffset-p90/2000); #offset correction: -p90/2000 to
trigger the temporal axis at the center of the first pulse#

#-----Defining regions for the average-----#
strt1 = p90/2000 + rdt;
stop1 = strt1 + WIN_FID;
strt2 = (te) - WIN_ECHO/2; #the echo will be shifted due to the group
delay#
stop2 = strt2 + WIN_ECHO/2;
#-----Defining regions for the average-----#

mark( -1, -1, 0 );
mark( strt1, stop1, 1);
mark( strt2, stop2, 1);

#-----Phase cycling-----#
ph90[ 0] = 0; phrc90[ 0] = 90; ph18[ 0] = 0; phrc18[ 0] = 270;
ph90[ 1] = 90; phrc90[ 1] = 180; ph18[ 1] = 0; phrc18[ 1] = 180;
ph90[ 2] = 180; phrc90[ 2] = 270; ph18[ 2] = 0; phrc18[ 2] = 90;
ph90[ 3] = 270; phrc90[ 3] = 0; ph18[ 3] = 0; phrc18[ 3] = 0;
ph90[ 4] = REDO;phrc90[ 4] = REDO; ph18[ 4] = REDO;phrc18[ 4] = REDO;
#-----Phase cycling-----#

par;
magn_mode ( "PSD" );
endpar;

pulses;
ssp ( p90, ph90 );
sd ( rdt );
adi ( acq1,td1,phrc90);
sd (dacq1);
ssp ( p18, ph18 );
sd ( rdt + 0.1); #delay to ensure that the filter is reset to initial
condition#
adi ( acq2, td2, phrc18);
endpulses;

measure;

label Escape;
return( TRUE );

```

11.3 A Simplified CPMG Application (cpmg_example)

You can copy the code and paste it in a new ExpSpel program or ask us the application by e-mail.

```

program setup();

par;
gain (50);
scans (32);
rd (1.500000000);
det_mode ("complex");
magn_mode ("PSD");
dummy_shots (0);
endpar;

int decim;
real dw,swh;

#-----Customize your filter settings?-----#
decim = 8; #digital bandwidth of the filter is 12500kHz/decim,
decim=2^N #
#-----#
swh = 12500000/decim; dw = 1/swh;
par;
dbw(swh/1000);
endpar;

return(TRUE);

program measure();

real dw,swh,aq,tEcho,pre,post,tres,dur;
int decim,grpDly,td,asdshift,echoes;
int ph90[16],ph18[16],phrc[16];

#-----Compatibility check-V8.0-Rev0-FW320-----#
int compat,appv;
char comp_message1[500];
appv=8;
compat=isCompat(appv);

strcpy(comp_message1," WARNING - Incompatibility detected! ");
strcat(comp_message1,"versions expected: Firmware 3.2, FPGA V1BA, the
minispec Software V3.00, application Pool V8." );
strcat(comp_message1," Please check the configurations of your system
in <Help>/<About the minispec>. ");
strcat(comp_message1," Results might be invalid.");

if (compat == 0)
print_line(CONFIRMBOX,comp_message1);
if (ESC) goto Escape;
else
print_line(RESULTBOX,"-----");
print_line(RESULTBOX," WARNING - Incompatibility detected! ");
print_line(RESULTBOX,"versions expected: Firmware 3.2, FPGA V1BA, the
minispec Software V3.00, application Pool V8." );

```

Examples

```
print_line(RESULTBOX,"Please check the configurations of your system
in <Help>/<About the minispec>.");
print_line(RESULTBOX," Results might be invalid.");
print_line(RESULTBOX,"-----");
endif;
endif;
#-----END-Compatibility check-----#

#-----Customize your filter settings?-----# #V7.1#
decim = 8; #digital bandwidth of the filter is 12500kHz/decim,
decim=2^N #
#-----#

#---Defining the digital bandwidth----#
swh = 12500000/decim; dw = 1/swh;
par;
dbw(swh/1000);
endpar;
#---Defining the digital bandwidth----#

#-----V7.1-FW-3-2-0-Profiler-CPMG--# #group delay table#
if (decim==1) grpDly=48; endif;
if (decim==2) grpDly=27; endif;
if (decim==4) grpDly=27; endif;
if (decim==8) grpDly=19; endif;
if (decim==16) grpDly=14; endif;
if (decim==32) grpDly=12; endif;
if (decim==64) grpDly=11; endif;
if (decim==128) grpDly=11; endif;
if (decim==256) grpDly=11; endif;
#-----V7.1-FW-3-2-0-Profiler-CPMG--#

#-----Phase Cycle:-----#
ph90[ 0] = 0; ph18[ 0] = 90; phrc[ 0] = 90;
ph90[ 1] = 0; ph18[ 1] = 270; phrc[ 1] = 90;
ph90[ 2] = 180; ph18[ 2] = 90; phrc[ 2] = 270;
ph90[ 3] = 180; ph18[ 3] = 270; phrc[ 3] = 270;
ph90[ 4] = 90; ph18[ 4] = 0; phrc[ 4] = 180;
ph90[ 5] = 90; ph18[ 5] = 180; phrc[ 5] = 180;
ph90[ 6] = 270; ph18[ 6] = 0; phrc[ 6] = 0;
ph90[ 7] = 270; ph18[ 7] = 180; phrc[ 7] = 0;
ph90[ 8] = REDO; ph18[ 8] = REDO; phrc[ 8] = REDO;
#-----Phase Cycle:-----#

#-----Global parameters-----#
tEcho = 0.5; #echo time#
echoes = 2000; #Number of echoes#
#-----#

#---calculating the acquisition parameters---#
td=2^(trunc(log(2*grpDly)/log(2)));
aq=1000*td*dw;
asdshift=grpDly; #shift caused by the filter#
#---calculating the acquisition parameters---#

tres=40/1000000; #resolution of 40ns for delays#
tEcho=trunc(tEcho/tres +0.5)*tres;

pre= tEcho/2-aq-get("18P")/2000+1000*asdshift*dw;
```



```
post=tEcho/2-get("18P")/2000-1000*asdshift*dw;
dur=tEcho/2-get("90P")/2000-get("18P")/2000; #V8.0#

#---making the delays multiple of temporal resolution---#
tres=40/1000000;
pre=trunc(pre/tres +0.5)*tres;
post=trunc(post/tres +0.5)*tres;
dur=trunc(dur/tres +0.5)*tres;
#---making the delays multiple of temporal resolution---#

pulses;
ssp(get("90P"),ph90);
sd(dur);
ssp(get("18P"),ph18);
ploop(echoes)
sd(pre);
adi(aq,1,phrc);
sd(post);
ssp(get("18P"),ph18);
endploop;
endpulses;

sig_offset(-get("90P")/2000);
ph0corr(1*tEcho,5*tEcho,0,ON);

measure;

label Escape;

return(TRUE);
```


12 Background of the Transition from ND to NF

At the end of 2012 Bruker introduced a new electronics unit, the minispec NF Series. The strategy for the new series was to maintain all the advantages of the minispec mq-Series, but improve the NMR signal filtering through the introduction of a digital filter. The digital filter, as it has been implemented on NF systems, improves the Signal to Noise (S/N) ratio on average by 20%, but this is strongly application dependent. This following list describes in detail which parts of the minispec mq-Series have been updated and which remain unchanged:

- Probes, pre-amplifiers, magnets and most of the magnets tempering electronics remained unchanged.
- For the Pulsed Gradient Unit: Most of the components / electronics remained unchanged.
- Same for the software surface: The minispec.exe and minispec Plus software remain mostly unchanged in the way they appear to the instrument user.
- All minispec users running standard applications (80% of minispec customer base) will not recognize the electronics change. The software and the applications that are delivered with the instruments for the routine applications will appear as before.
- For the user the difference is only evident when R&D creates their own applications. Here it should be mentioned that all parts of the application program remain unchanged, except for the data acquisition and data processing.
- In the data acquisition part of the ExpSpel application program the use of the digital filter needs consideration. The application pool for the NF series has all this new programming for the digital filter included, so a programmer or a scientist may easily copy the needed changes from there.
- For the Bruker routine applications Bruker has made significant efforts to validate the new systems and to make sure that the values of the new and the old systems are in agreement. On average, customers will benefit from improved system performance.

Conclusions

- Routine users will not even recognize the electronics transitions.
- Users writing their own NMR sequences will need to adjust the data acquisition and data post-processing due to the digital filter and the group delay points. Bruker can provide examples on how to program this.
- The minispec ND and NF systems (with the corresponding programming) provide comparable results as was before between different ND systems.
- Typically users will benefit from a 20% increase in S/N ratio (depending on the application).

13 Contact

Manufacturer:

Bruker BioSpin GmbH
Am Silberstreifen
D-76287 Rheinstetten
Germany
Helpdesk Europe: (+49) 721-5161-6155
Helpdesk USA: (+1) 978-667-9580
E-Mail: minispec.SLS@bruker.com
<http://www.bruker.com>
WEEE DE43181702

Bruker BioSpin Hotlines

Contact our Bruker BioSpin service centers.

Bruker BioSpin provides dedicated hotlines and service centers, so that our specialists can respond as quickly as possible to all your service requests, applications questions, software or technical needs.

Please select the service center or hotline you wish to contact from our list available at:

<http://www.bruker.com/service/information-communication/helpdesk/magnetic-resonance.html>

List of Figures

Figure 3.1:	Modifications in the Instrument Settings.....	9
Figure 3.2:	The Parameter Table in the minispec.exe.....	9
Figure 3.3:	The Settings Viewer in the minispec.exe.....	10
Figure 4.1:	Example of NMR signal measured using the digital filters. Inside the region marked in red one can visualize the group delay points.....	12
Figure 5.1:	Signal/Noise Ratio as Function of decim.....	14
Figure 5.2:	Time Required to Achieve the Signal/Noise Relation of 7000 for Several Values of decim.....	14
Figure 6.1:	Free Induction Decay Obtained Using a 10R Probe, 90 Degree Pulse of 2.7 μ s and RDT = 6.08 μ s.....	15
Figure 6.2:	Solid Echo Signal Measured in the mq-ProFiler.....	16
Figure 6.3:	Using the Range Command. (a) Status set to FALSE; (b) Status set to TRUE.....	18
Figure 6.4:	Definition of the Phase of the NMR Signal.....	19
Figure 6.5:	Error Message Displayed if the Application Does Not Have the isCompat() Command.....	20
Figure 6.6:	Error Message when Executing an Application which is not Compatible to the Firmware, FPGA, Software Installed.....	21
Figure 8.1:	NMR Signal (FID) for Different Values of decim.....	25
Figure 8.2:	NMR Signal (FID) for Different Values of decim and irate.....	26
Figure 9.1:	Number of Group Delay Points Determination: (a) decim = 1; (b) decim = 16.....	27
Figure 10.1:	A FID-ECHO Experiment.....	30
Figure 10.2:	Zoom Around the Echo Tops Displayed from Previous Figure.....	30
Figure 10.3:	Two Acquisitions Performed with a Delay in Between Shorter than the Time Required to Reset the Digital Filter.....	31
Figure 10.4:	CPMG Results Showing the Real and Imaginary Components of the NMR Signal.	32

List of Tables

Table 2.1: Commands that have changed in minispec.exe Version 2.80 or Higher	7
Table 4.1: Allowed values for dbw and dw on NF-series minispec.....	12
Table 6.1: Examples of sig_ and mdt_ Commands	17
Table 10.1: Group Delay Table for Sequences with Multiple Acquisitions	32





Bruker Corporation

info@bruker.com
www.bruker.com

Order No: E1400012