BioPack

User Guide

Pub. No. 9100094200 Rev. A 11/12/09

VARIAN

BioPack

User Guide

Pub. No. 9100094200 Rev. A

Copyright © 2009 Varian, Inc.

2700 Mitchell Drive
Walnut Creek, CA 94598 USA

http://www.varianinc.com

All rights reserved. Printed in the United States.

The information in this document has been carefully checked and is believed to be entirely reliable. However, no responsibility is assumed for inaccuracies. Statements in this document are not intended to create any warranty, expressed or implied. Specifications and performance characteristics of the software described in this documentation may be changed at any time without notice. Varian reserves the right to make changes in any products herein to improve reliability, function, or design. Varian does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Inclusion in this document does not imply that any particular feature is standard on the instrument.

VnmrJ is a trademark of Varian, Inc. Other product names are trademarks or registered trademarks of their respective holders.

Contents

Chapter 1 Introduction

Biomolecular NMR experiments utilize a wide variety of macros, menus, pulse sequences, layouts, text files, parameter sets and other files for the execution of experiments, typically in H_2O solutions. This manual describes the nature of these files for the benefit of those who wish to modify or add similar files for new experiments, as well as to just understand normal operation.

The basis of the implementation of these experiments is the reproducibility and predictability of the NMR console as well as the spectral nature of biologically relevant materials. This combination makes possible the automatic setting of power levels, pulse widths, and shaped pulses present in modern pulse sequences. Several styles of pulse programming are possible, depending on the preference of the user. They all have the same common attribute- ease of setup and high performance.

1.1 Activating BioPack

BioPack comes pre-installed in VnmrJ versions 3.0 and later as an “appdir” (/vnmr/biopack). This means that all BioPack files are present in this one directory and that they are active if a biopack appdir is “enabled” in the Applications file (selected by MainMenu…Edit…Applications). Enabling this mode causes the interface files to change from the default mode to a BioPack mode. The Main Menu Experiments menu also changes and presents a set of options for “activation” of BioPack. This involves creation or copying of a probefile, installing a ghn_co parameter set and creation of a number of global parameters (in ~/vnmrsys/global). One of these is “BPinstall”, which is set to 1.0 when the activation process is complete. At this time the user should exit VnmrJ and restart. Following this, the Main Menu Experiments menu will have active experiment selection. For more details on the actual operations see the macro BPbiopack1a below (Table 12 in Section 7.0).

1.2 Updating BioPack

BioPack is regularly updated with new experiments, bug fixes and enhancements. These are incorporated into the standard software and are available at the time of release of new versions of VnmrJ. Since the time between obtaining new releases may be long, BioPack updates may be obtained on-line by visiting www.varianinc.com and selecting Magnetic Resonance/NMR from the Products menu and then “User Pages/Software Corner/User Library”. At this time a login and password is required (this may be obtained by using the link to the “Registration Form” on the same page).

Once logged in, select “Recent Additions and Updates” and choose the latest BioPack update. Follow the instructions in the Readme file.

The update process installs all new files and compiles any new pulse sequences, leaving your Probefile as it was.

This update process should be done regularly to benefit from new capabilities and bug fixes. The associated Readme file details all the changes.

1.3 Operation

All NMR experiments require, at a minimum, a compiled pulse program and a parameter set. At this level, the user must make choices of parameter values. Shaped pulse text files may be required to be created, and pulse powers and pulse widths must be calibrated and set. Modern pulse programs may have dozens of RF pulses and gradients so that experiment setup in this manner may take several hours, even by expert operators.

To make the same operation just as accurate, but have the time to set up in seconds rather than hours, other types of files must be involved. At first, the user needs to “find” the experiment in a menu or other visual mode. Ideally, the user would “click” on the desired experiment name and the full experiment would be set up, including setting of all parameters. At this stage the user would only change parameter values affecting needed sensitivity and resolution (acquisition time in the detected dimension, number of increments in indirect dimension(s) and the number of transients per FID).

It is crucial that the performance of the experiment is not compromised by this automatic setup. This can be assured by proper pulse sequence design and the predictability of the spectrometer. If a user must “tweak” parameter values it is an indication of non-reproducibility of spectrometer manufacture or non-predictability of the hardware performance.

This automatic mode of experiment setup is entirely governed by macros. These macros access a probe calibration file to retrieve parameter values and make experiment-specific changes. The Probefile can be edited manually, but automatic calibration and Probefile updates can be done via macros as well.

1.4 Experiment Selection

Experiments can be selected in a variety of ways. Since the process involves running a macro, this macro could be entered on the command line. Of course, this depends on the operator remembering the exact macro name (usually the name of the pulse sequence). These names can be long and this mode of operation has the highest error probability.

Modern interfaces use powerful and intuitive graphical displays that can present choices to the operator in an obvious manner. This is often in the form of a drop-down menu. Other times it may be in the form of a selection of buttons. In all cases, a system text file that is viewable and editable by the user governs the menus or button selections.

1.5 Interface

The VnmrJ interface is modern and “rich”. The richness arises from the flexibility of the java structure and the resolution of modern displays. VnmrJ uses a variety of mechanisms to set up experiments. One method is to use a drop-down menu that may offer choices of sub-menus. Usually, only one or two sub-menus are needed to get to the desired experiment. Clicking the name results in the experiment macro being run. In other cases, experiments are set up via a button within a “Protocol” list, or present on a panel.

When the new parameter set is installed the parameter area (layout panels) changes to display XML files relevant to the new experiment. Some of these files might be experiment-specific, others are defaults common to many other experiments. These files are located in /vnmr/biopack/templates/layouts, a user-installed Appdir (~/vnmrsys/biopack/templates/layout) or the corresponding user directory (~/vnmrsys/templates/layout). These files are editable and viewable. While these files are understandable, editing them directly is not convenient or rapid. For this, a graphical editor is present using “drag-and-drop” techniques for rapid user customization. (For details, see the *User Programming Manual*.)

The layout panels offer methods for changing all relevant parameters (entry boxes, check boxes and menus). They are organized for efficient operational flow. Panels are available for experiment setup, acquisition, data processing, data display and plotting, and for archiving. In the most flexible types of panels, virtually any operation used by an expert operator can be made without resort to the command line. Obviously, this allows all users (who are expert enough in the NMR knowledge) to work from the start without spending a long “apprentice” phase of learning commands and parameters.

Many panels have “widgets” that initiate complex operations such as full automatic calibration of all RF channels and gradients, or updating the probefile with new values. These “widgets” are simple in themselves, only starting an appropriate macro.

Just above the parameter panels are a series of “tabs” that control the display of parameter panels. They can be considered “folders” in which “pages” (the displayed parameter panels) are stored. Associated with each “tab” is a set of “action” widgets immediately to the right of the “tab”. These are cause actions appropriate to the type of task and do not depend on the displayed parameter panel. The buttons or menus run macros or commands and are produced from “action” XML files in the templates/layout/default directory.

The “tabs” follow the general experimental approach of setting up for a new sample in the probe, selecting a workspace in which to run an experiment, selecting an experiment, acquiring the data, processing the data, displaying the spectrum and plotting the result. There are more “tabs” than in the standard VnmrJ interface because of the increased number of operations possible or needed in biomolecular NMR.

Chapter 2 The Setup Folder

The Setup folder contains pulse sequence-independent pages. These are present as XML files in the templates/layout/default directory. The specific names can be viewed by using the MainMenu/Edit/Parameter Pages dropdown menu that produces the parameter panel editor popup. Double-clicking on the relevant page reveals the name of the XML file.

2.1 Locking and Temperature Control

The Lock page is used to set lock power/gain/phase parameters and to show the magnitude of the lock signal via the single-handed clock. The lock condition can be obtained by manually locking using the “Lock Scan” button in the action bar. For most biomolecular studies in H_2O/D_2O the lock is usually automatically re-established upon changing samples. The Eject and Insert buttons on the action bar control the sample ejection and insertion.

Since the pulse sequence has yet to be specified, the “FID Scan” button is not used at this time, but will be used later in association with the “Shim” page. The “Setup RF” button is used to set the channels to the desired nuclei so that the probe can be tuned (typically only for 1H) and is not used until an experiment is selected.

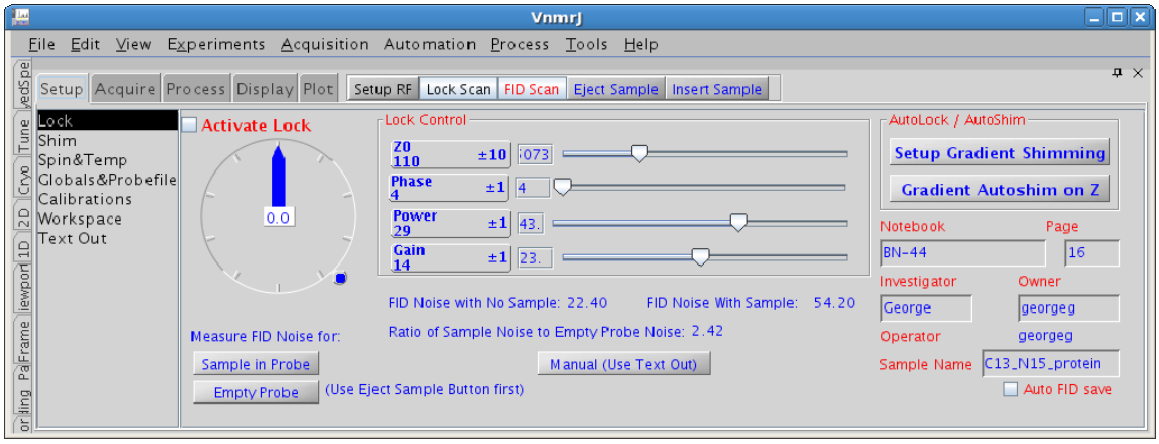


Figure 1 The Lock Page

After the lock is found and the lock signal is “on-resonance”, select the Activate Lock checkbox to establish the field/frequency lock condition.

The next step is to specify values for the various “global” parameters via the displayed entry boxes. These parameters are not present in the current parameter set, but are in the vnmrsys/global file so that their values remain, even when a new parameter set is selected. These values are important for Locator sorting of data sets for desired data set properties. Thus, one could display all experimental data sets described by the notebook name, notebook page number or sample name. These values are stored in the Locator database when a FID is stored.

Automatic FID saving at the end of an experiment is activated by the “Auto FID Save” checkbox (as well as in the Save Data Setup method via the drop-down menu).

Buttons are provided to measure the time-domain noise using the pulse sequence delay-obs (i.e. no pulse) for the cases of an empty probe and a probe with sample. The results are then displayed. (While not a significant factor for non-lossy solvents, the sample is a considerable source of noise for salty aqueous samples when using cryogenically-cooled RF coils.)

Once the sample is locked, any new temperature needed for the sample is selected via the “Spin/Temp” page.

This page permits changing temperature. If the probe has been calibrated with a standard test sample so that 0.1 degree settings are meaningful, the temperature can be entered to this precision. The actual temperature is displayed. Spinning is normally turned off for biomolecular NMR experiments.

Particularly for biomolecular NMR, it is important to avoid accidental changes of temperature caused by retrieving old data and setting up the hardware with the temperature parameter active. This may be avoided by specifying control of VT only by this panel (selecting the checkbox at the bottom of the page).

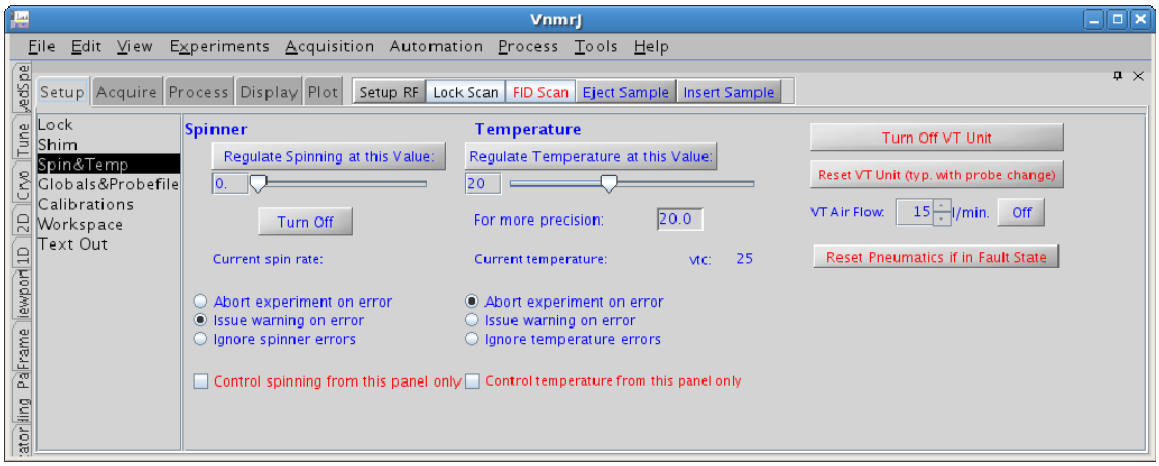


Figure 2 The Spin/Temp Page

2.2 Shimming

When the sample temperature is stable, it is feasible to shim the magnetic field. This can be done using Gradient Shimming using the current field map by clicking on the “Gradient AutoShim on Z” button on the Lock page. This is the normal and preferred method of shimming biomolecular NMR samples. This uses the H2O signal in the sample and optimizes typically z1-z6.

The “Lock” page also has a button to “Setup Gradient Shimming”. This runs the “gmapsys” macro which permits making a new shim map.

After gradient shimming the transverse (or “non-spin”) gradients may be checked or optimized via the “Shim” page.

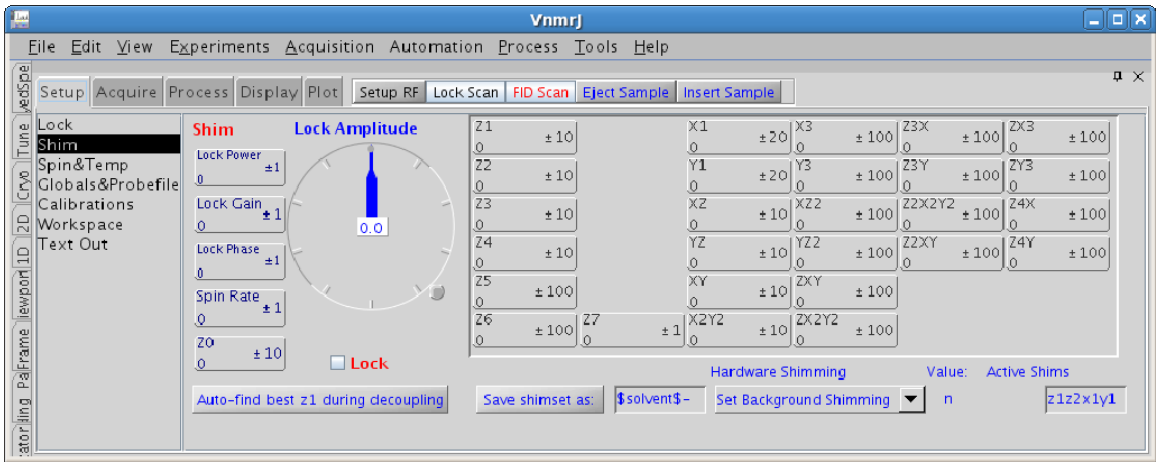


Figure 3 The Shim Page

This panel shows a two-handed clock having a more sensitive second hand, the power/gain/phase controls for the lock channel and a z0 control. The shim set present in the magnet determines the shimset parameter which controls the number of shim widgets present on the panel. A current shim set for acquired data may be saved by specifying a name in the entry box. Clicking the “Save shim set as:” button stores the shim set in vnmrsys/shims under that name and these can be displayed and used with the Locator.

Background shimming can be specified via the menu as in a delay, in a presat period or none. The active shims used by this process are specified in the “Active Shims” entry box and are indicated by the TextMessage under “Value”.

The “Auto-find best z1 during decoupling” button is used to start an automatic process wherein the lock signal is optimized under the execution of the current pulse sequence and parameters, even in the presence of gradients. This is done by automatically setting the global parameter “gradient disable” to ‘y’, running the pulse sequence repeatedly while the lock signal is sampled while changing the z1 shim value. After the best z1 value is found, the experiment is stopped and gradient disable is reset to ‘n’. This permits the best shimming to be obtained for experiments involving heteronuclear decoupling in which the z1 shim may change during decoupling (more likely with a cold probe that might have a temperature-dependent magnetic susceptibility.)

The Shim Page is used when using the “FID Scan” action button to shim on the FID or spectrum. When the “FID Scan” button is clicked, the Shim Page shows radio buttons at the left to permit selection of FID or Spectrum. If the FID radio button is selected, the Lock channel widgets are replaced by widgets permitting changes in the FID display. The lock amplitude is not displayed, but rather the FID area, by clock and direct value display. This spectrum shimming is valuable for small molecules with sharp lines.

2.3 Using “Experiments”

Now that the sample is locked, at a stable temperature and well-shimmed, the “Workspace” page is used to select a workspace (Experiment) in which to work.

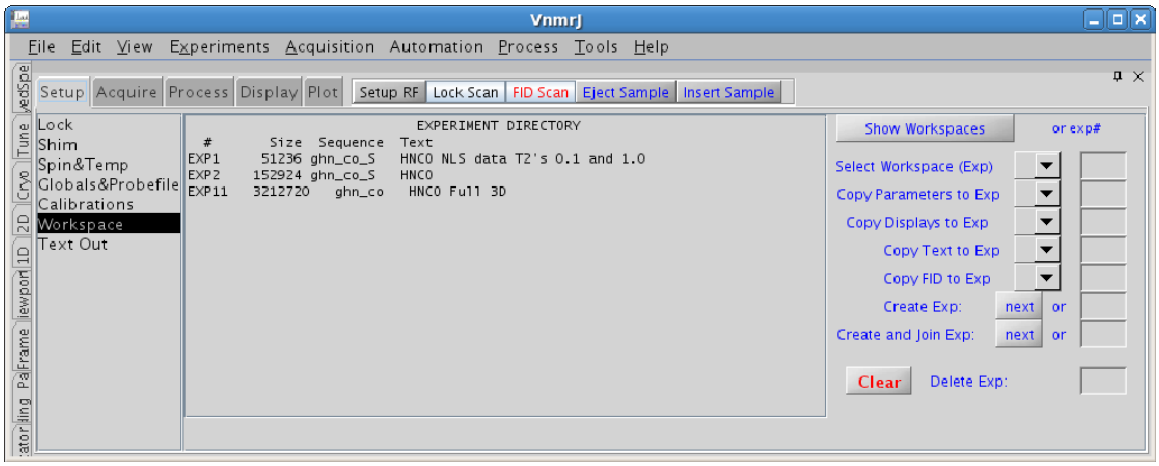


Figure 4 The Workspace Page

This page contains elements that facilitate copying files between “Experiments” or “Workspaces” such as exp1, exp5, etc. The list of exp’s in vnmrsys are displayed in the text window with the “Show Workspaces” button which runs the macro “explib”.

The text window may be cleared of its contents with the “Clear” button.

There are several menu widgets that give selections from 1-9. Selecting a number and releasing the mouse button performs the indicated action from the current Experiment to the requested Experiment. Thus, while “in” exp100, selecting 5 from the list can do the following:

- “Join” Experiment 5 (jexp5),
- Copy the current parameter set to Exp5 (mp(100, 5)),
- Copy stored display files (s1, s2, s3…) to Exp5 (md(100,5)),
- Copy the current Text file to Exp5 (mt(100,5)),
- Copy the FID and current parameters and Text to Exp5 (mf(100,5)),

These actions can also be done via the Main Menu. The entry boxes are provided for Experiments greater than 9. In this case, use the return key after entering the Experiment number.

A new workspace may be created (cexp) by the “next” button or entry box after “Create Exp”. The “next” feature automatically selects the smallest unused Experiment number. These features can also be combined with “joining” the new Experiment. A specific Experiment may be deleted (delexp) by specifying the number followed by the return key.

Once these operations have been done a specific experiment is normally selected from the MainMenu/Experiments drop-down menu. When this happens, a macro by the same name as the pulse sequence name is executed to install a parameter set in the current Experiment (or Workspace). The parameter set is automatically adjusted using values from the current probefile so that the experiment is ready to be run after specifying the number of transients (and increments for nD experiments).

There are three other pages present in the Setup Folder that have less frequent use and are used for controlling BioPack settings or for displaying useful text.

2.4 The Text Output Page

This page has a number of miscellaneous useful functions. The text output window is used for display of text files.

The “Show a File” menu is used to display text files. This menu permits display of an experiment log file, the parameters “dg”, “dg2” and “ap”, as well as the Text file. The “Edit a File” menu permits editing of these files using a popup editor (anytime a new parameter is added to a pulse sequence, the parameter set should be modified to add this parameter and it should be displayable as well).

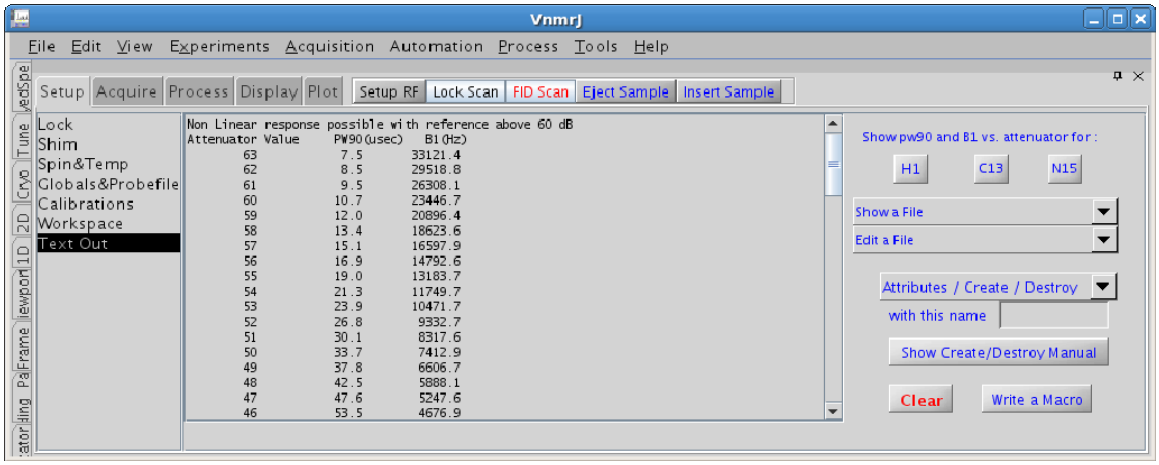


Figure 5 The Text Out Page- Showing a Text File Generated by the “H1” Button

The “Attributes/Create/Destroy” menu is used to display characteristics of a parameter, to create it or to destroy it. This utility allows the rapid addition of new parameters to a parameter set. There is also a button to display the manual for Create/Destroy for further information.

The three buttons at the top run the BPattval macro which produces a table showing power levels, pw90’s and B1 values for different attenuator values for H, C and N (based on the pw/compH, pwC/compC and pwN/compN values in the current parameter set). This is useful when a desired pw90 or B1 value is desired.

The “Write a Macro” button produces a popup editor for writing a macro.

The next step in normal operation would be to select an experiment from the drop-down Experiments menu and to proceed to the Acquire folder. However, there are a couple of pages in the Setup folder that are BioPack-specific pages that are not associated with any specific NMR experiment. They are used to set “global” parameters, update the probefile and perform automated calibration and analysis. The first of these is the “Globals&Probefile” page.

2.5 The Globals & Probefile Page

This page is shown in Figure 6. The values displayed here are often set once and then left unchanged for normal operation.

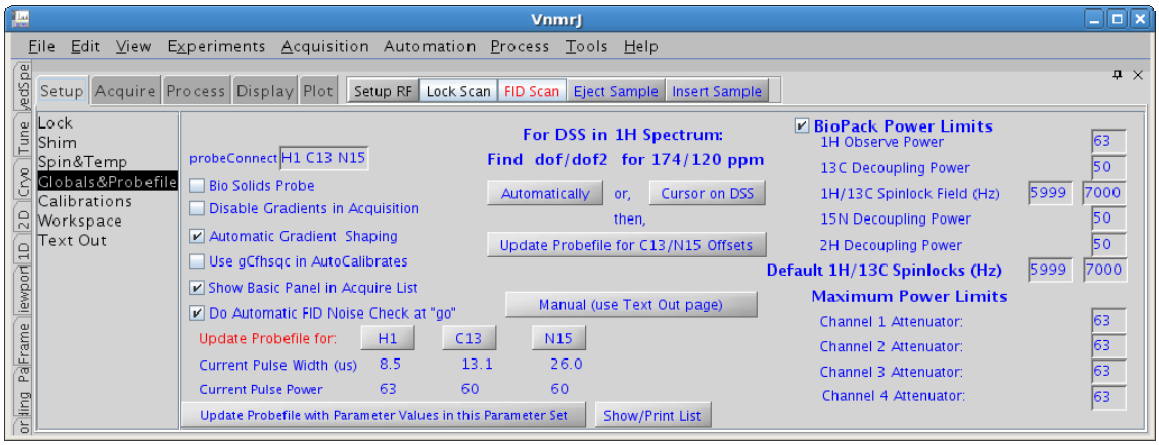


Figure 6 The Globals & Probefile Page

The first time BioPack is used the operator has the choice to “activate” global settings to permit certain operations. These are described below:

2.5.1 Activation Buttons and Global Variables

The following table describes the buttons available.

Button	Description
"Activate Absolute Power Limits for each Channel"	This button creates global parameters for each channel and displays these attenuator limits at the bottom right of this page. These are pulse sequence and parameter set independent limits that are used by the "go" program. They are useful for probe protection. Because they are global variables for each user, they only apply to the current user.
“Activate Qcomp Option"	This button creates the "qcomp" global variable and displays a check box. Qcomp should be set to 'y' for cold probes used on INOVA or Unityplus spectrometers only. The activate button is not visible or used for VNMRS consoles.
"Activate Disabling Gradient Option"	This button creates the "gradientdisable" global variable and associated checkbox. When the checkbox is checked any pulse sequence using zgradpulse is executed with the gradient amplitudes set to zero, to permit shimming while running the experiment. This may be useful if decoupling induces a change in field homogeneity.
"Activate Gradient Shaping Option"	This button creates the "gradientshaping" global variable and associated checkbox. When the checkbox is checked all zgradpulse events have a "wurst" shape to provide a smooth start and end to the gradient. It is often used for cold probes or any probes that have a longer than desired field recovery. Gradient calibrations may change for this option being used or not.
"Activate ProbeConnect Option"	This button creates the "probeConnect" global variable and associated checkbox.

Several other checkboxes are present without activation.

Checkbox	Description
"Show Basic Panel in Acquire List"	This makes the "Basic" page visible in the Acquire folder. The Basic pages permit rapid setup of all experiments, with the choice of experiment type (1D, F1F2, F1F3, F2F3 or 3D) with default 1ppm resolution in indirect dimension(s) and nt=8. The time of the experiment is displayed after a choice is made. Customization of parameters (nt, ni, ni2) is possible and the pulse sequence manual is displayable on request. The use of a Basic page requires that the Probefile be accurate for the sample used.
"Do Automatic FID Noise Check at 'go'"	This sets the global parameter "BPautonoisecheck" to 1.0. The Acquisition Actions menu on the action bar for the Acquire tab runs "BPgo" when an acquisition is selected. If BPautonoisecheck=1, the macro BPnoisecheck is run prior to the experiment to obtain time-domain noise under standard conditions (no pulses) to measure the effect of the sample on the noise level. (This is done manually with and without a sample in the probe by buttons on the Lock panel in the Setup folder.) AutoTest "history" files are automatically created and updated for these two cases.
"Use gCfhsqc in AutoCalibrates"	This uses the gCfhsqc (3919 watergate) pulse sequence in place of hcch_tocsy (with no spinlock or decoupling) for the 1H and 13C calibration steps. The sequence is simpler and may be less prone to error when BioPack AutoCalibrate is used the first time on a system.
"BioPack Power Limits"	This permits user-defined attenuator limits for the rf channels, including limits for decoupling waveforms. When Autocalibrate procedures run, macros re-create decoupling waveforms and if the resulting power levels are in excess of these limits, the waveforms are re-created at these powers. A maximum 13C spinlock field (in Hz) is also available, and this value is used for re-creating spinlock waveforms if needed during the autocalibrate runs. Updating a probefile with any button involving 13C will also result in a re-creation of 13C decoupling and spinlock waveforms. 1H spinlocks are also limited by the global parameter BPtpwrspinlock which is enterable in the BioPack Power Limits group.

Checkbox	Description
"Default Spinlock Field Strengths"	The default value for experiments using a 1H spin lock is given by the global parameter BP1Hspinlock. This value is enterable under the title "Default Spinlock Fields" along with the 13C default spinlock (BPspinlock) normally used for all experiments. These values are often lower than the maximum user-defined spinlock values. The 13C value is used for probefile updates and when new spinlock waveforms are generated. Entry of this parameter in the PulseSequence page via an entry box also runs the appropriate macro to regenerate only the spinlock waveforms. All 1H experiments involving a 1H spinlock (tocsy) have setup macros that set the tocsy spinlock field strength to the default value BP1Hspinlock described above. The user is still free to change the spinlock field strength (usually governed by the parameter "strength" by the appropriate entry box). The entry of a value will automatically run a macro BPspinlock which checks the new value against the BioPack spinlock power limit BPtpwrspinlock and, if in excess, sets the strength=BPtpwrspinlock (if BioPack power limits are active). The same check is done for 13C spinlocks, and if the spinlock field strength is changed a new waveform cctocsy.DEC is regenerated by Pbox.

2.5.2 DSS in Spectrum

If a 1H spectrum is run using DSS as a reference, either of two buttons may be used to calculate proper values for dof and dof2 to be at 174ppm and 120ppm respectively. The "Automatically" button will search for the strongest line between -0.5ppm and 1.5ppm and will use this as the DSS line. This runs the BPfindfrqs macro.

Alternatively, the user may place the cursor on the DSS line manually and then use the "Cursor on DSS" button to run the macro BPfindfrqs(1) macro.

In either case, equations relating the ratios of 1H/13C/15N frequencies on an absolute basis are used to determine the proper values of dof and dof2. Then the probefile can be updated to store these values in the probefile.

There may be small differences between this method and using "setoffset".Setoffset is used when BioPack is first installed and via the menus on the Channels.xml page.

2.5.3 Update Probefile Buttons

Button	Description
“Update Probefile Buttons H1, C13 and N15”	These buttons run macros that update the current probefile using the powers and pw90's listed (these are obtained from the current values in the current parameter set). The “H1” button only updates the its section of the probefile, as does the “N15” button. The “C13” button updates the 13C pw90 and also re-creates all 13C waveforms. The 1H update scales up or down all fine powers present in the 1H section of the probe for flipback pulses, to reflect the change in tuning. The 15N update also creates waveforms for decoupling

Button	Description
"Update Probefile with Parameter Values in this Parameter Set"	This button is only visible when the ghn_co parameter set is present. This runs the BPbiopack2 macro which does a full update on all nuclei, including re-creation of all waveforms and shaped pulses. It must be used at installation after reasonable parameter values have been entered for the rf channels. The same macro is run during the AutoCalibrate process.

All the global parameters used in BioPack are created by the macro BPglobalpars.

2.6 The Calibrations Page

The Calibrations Page in the Setup Folder contains menus and buttons to perform automatic calibrations.

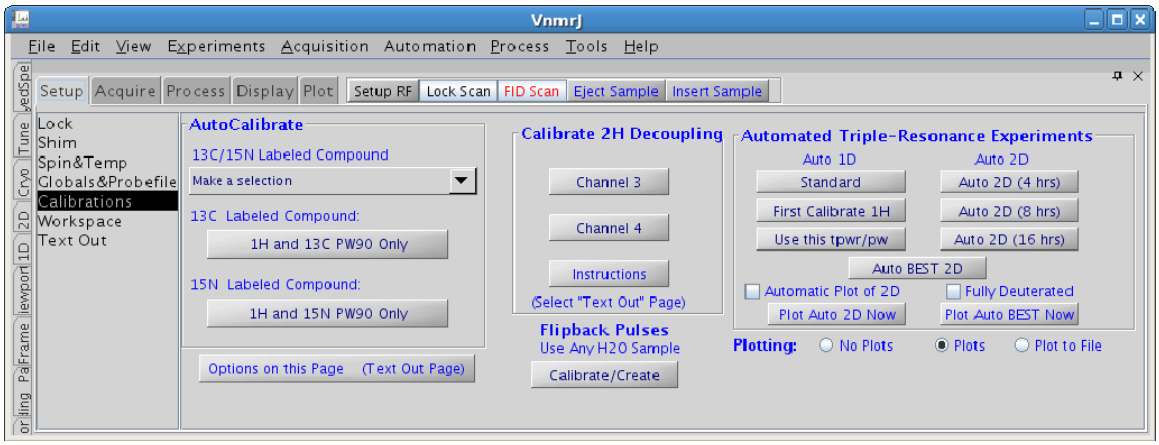


Figure 7 .The Calibrations Page

There are several BioPack Options in "Calibrations" Page for use in labeled and unlabeled molecules. They run macros that setup acquisition, process data and plot data. The data are automatically saved in directories in vnmrsys. Protein solutions in H_2O or D_2O are used, typically.

2.6.1 AutoCalibrate Buttons and Menu

When using a 13C, 15N-labeled compound (menu)

Button	Description
"Full, Using Probefile Values"	This button begins a fully automated calibration of all important parameters for doing Biomolecular NMR experiments. It should be used after installation of BioPack so that all parameters will be properly calibrated and values stored in the probefile. Many parameters such as gradient levels and small-angle phase shifts are calibrated in this process. These do not change as a function of sample, so the "Full" autocalibrate does not need to be done for each sample. In addition to calibrations, first-increment spectra are obtained for a wide variety of experiments and these spectra are displayed and plotted in a side-by-side manner to permit easy comparison of "receptivity" for the sample used. A copy of the probefile is also printed. One important operation that is included for any of these buttons is the automatic update of the probefile and the automatic re- generation of all shaped pulse files and waveforms used for spinlocks and decoupling
"Full, But use Current 1H Offset and pw"	This button does the same as that described above, but uses the tof and pw values in the current experiment as starting guesses. It is useful if the user has calibrated the 1H pw90 manually
"Full RF (Faster)"	This button will use the same experiments as above, but will stop after the RF and NH gradient coherence transfer calibrations. It can be used once the "Full" calibration is performed and is often preferred because of the shorter time required.
"RF and Gradients (Fastest)"	This button will use similar experiments as above, but will not do amplifier compression determinations or flipback pulse calibrations. Use of this assumes a reasonably calibrated probefile and is used for a quick check of the most important parameters, pw90's and gradients.
"1H pw90 only"	This button will use the similar experiments as above, but will only do 1H pw90 and related calibrations. This may result in a slightly different value than that obtained by using a single pulse because the pulse sequence used here involves multiple 90's and 180's. This results in a more appropriate value for Biomolecular NMR experiments

For a 13C-labeled Compound (button)

Button	Description
"1H and 13C PW90 only"	This button will use the same experiments as above, but will only do 1H/13C pw90 and related calibrations.

For a 15N-labeled Compound (button)

Button	Description
"1H and 15N PW90 only"	This button will use the same experiments as above, but will only do 1H/15N pw90 and related calibrations.

In all cases data are stored in ~/vnmrsys/AutoTripRes. New runs cause the existing directory to be moved to ~/vnmrsys/BioPack.dir/backups with a date stamp appended.

Calibrate 2H Decoupling Buttons

Button	Description
"Channel#"	This button initiates a fully automatic calibration of the channel used for 2H decoupling. The proper button should be used for your hardware configuration. The experiments involve pulses derived from the 2H decoupling channel, with observe in the tn=lk mode so that the observe channel detects 2H signals via the lock diplexer, not the normal broadband preamp. This ensures that the RF path is the same as that used for 2H decoupling. Thus, the 2H decouplng channel should be connected to the proper relay in the "magnet leg" or "front end". After the 2H pw90 and power level are determined, the probefile is updated with proper power and dmf values for broadband decoupling of 2H from 13C. Experiment setup macros use these probefile values to install proper numbers so that 2H decoupling is typically called for by setting dm3='nyn' (for sequences having internally coded 2H decoupling portions).
"Instructions"	This button produces an instruction manual which is displayable by selecting the "Text Out" page.

Flipback Pulses

Button	Description
"Calibrate/Create"	This button performs the flipback(selective pw90 on H2O) pulse calibration on the H2O in the sample. These are optimized for power and phase and new shaped pulses "H2Osinc_*" are created for specific use in several pulse sequences. Any H2O sample can be used, but this process is normally done in the above experiments.

2.6.2 Automated Triple-Resonance Experiments

Button/ Check Box	Description
Auto 1D	
"Standard"	This button skips the calibration parts of the "Full..." buttons above, but does the first increment spectra for comparison purposes. It takes much less time and can be used when changing samples or adjusting conditions for the same sample (pH, temperature, etc.).
"First Calibrate 1H"	This button does the same as "Standard", but first calibrates the 1H pw90 (and updates the probefile). This is useful if the probe tuning changes or the salinity of the sample is changed, but all other aspects are the same.
"Use this tpwr/pw"	This button does the same as "Standard", but uses the 1H pw90 in the current parameter set. This is useful if the user has manually calibrated the sample.
In all Auto 1D cases, data are stored in ~/vnmrsys/AutoTripRes1D. New runs cause the existing directory to be moved to ~/vnmrsys/BioPack.dir/backups with a date stamp appended.	
Auto 2D	
"Auto2D (# hours)"	This button runs a series of 2D experiments using both 2D and 3D pulse sequences. It relies on a calibrated probefile. Fixed numbers of increments are used for a screening mode. The relaxation delay is adjusted so that all experiments will finish within # hours. Data are stored in the user's vnmrsys/AutoTripRes2D
“Fully-Deuterated Check Box"	This check box should be checked if the sample is per-deuterated. This results in CH-detected 2Ds being skipped.
“Auto BEST 2D”	This button runs a series of BEST (fast) 2D experiments using both 2D and 3D sequences. It relies on a calibrated probefile. Fixed numbers of increments are used for a screening mode. The relaxation delay is typically short. By default, 100msec is used. The acquisition time is also dropped in half from normal experiments because of the higher decoupling duty cycle. The macro that begins the experiments is BPauto2dBEST (optional relaxation delay, optional halving "at"). Data are stored in the user's vnmrsys/Auto2DBEST directory.
"Do Automatic Proc/Plot of 2D (at end)"	This check box causes automatic processing and plotting of the 2D experiments at the end of the Auto2D process. If checked, the BEST experiments plot at the end of every experiment.
"Do it Now" (4,8 or 16 hour experiments)	This button performs the same automatic proc/plot as above, but can be done once the experiments have finished.

"Plot Auto Best Now"	This button performs the same automatic proc/plot as above, but can be done once the BEST experiments have finished.

Plotting Mode

Mode of the Plot	**Description**
“No Plots"	No plotted output of AutoCalibration data is produced. However, raw data is still archived in ~/vnmrsys/AutoTripRes2D.
"Plots"	Spectra are plotted as the experiments complete in the AutoCalibration process. Data are archived in ~/vnmrsys/AutoTripRes2D.
"Plot to File"	Spectra are plotted as the experiments complete in the AutoCalibration process, but are plotted as PostScript files. The destination directory is the AutoCalibration data directory. The active plotter must be of postscript type. This mode is useful for remote operation, or where hard-copy plots are not desired. There is a button to display this section of the manual in the Text Out page.

Chapter 3 Setting up a Specific NMR Experiment

Once the sample is locked and shimmed, the NMR experiment is selected by using the Main Menu/Experiments drop-down menu. This displays options for several options (Water Suppression, Protein Backbone Assignment, Protein 13C/15N, RNA/DNA and C13 Observe). Clicking one of these displays a sub-menu. When a particular NMR experiment is chosen, the macro having the same name as the sequence is run (see later sections of this manual for details). The parameter set is updated using the stored calibrations and settings in the active probefile (specified by the global variable “probe”). Included in this process are parameter settings involved with 13C, 15N and 2H decoupling using channels 2, 3 and 4, as well as parameter settings for shaped pulses and gradients. Certain setup macros also re-generate shaped pulses. For those experiments involving 2H decoupling, calibrated parameters for the 4th channel are also obtained from the probefile.

3.1 Water Suppression Experiments Sub-Menus

3.1.1 1D

This menu displays a list of 1D water suppression experiments (presat, Watergate, WET,).

3.1.2 2D

This menu displays a list of 2D (primarily) homonuclear experiments using a variety of water suppression techniques. Again, the parameter set is updated from the probefile settings.

3.2 Protein Backbone Assignment Experiments Sub-Menus

3.2.1 NH-Detected

This menu lists triple-resonance 3D (primarily assignment) experiments. Only one entry is made for each basic sequence, even though each sequence may have multiple options such as TROSY or phase-cycling vs. gradient coherence selection. The specific choice of sequence version and options is made in the PulseSequence page of the Acquire Folder.

3.2.2 Fast Methods…

This menu lists a variety of 3D assignment pulse sequences designed to allow very fast acquisition, with relaxation delays like 100msec, with full sensitivity relative to “normal” versions.

3.2.3 CH-Detected…

This is similar to NH-Detected but here the detected magnetization is CH.

3.2.4 Aromatics…

3D triple-resonance experiments for aromatic resonance assignments.

3.2.5 For Fully-Deuterated Proteins…

These are designed for only fully-deuterated proteins for resonance assignment

3.2.6 Coupling Measurements…

These produce spectra in which it is possible to make highly accurate spin-spin coupling measurements involving 1H, 13C and 15N.

3.2.7 Non-Linear Sampling (Wagner)…

These triple-resonance 3D pulse sequences acquire data in which the evolution times are obtained from a list instead of having regular sampling. This permits collection of data in significantly shorter time (contributed by the Gerhard Wagner Lab, Harvard Medical School).

3.2.8 3D Projection Reconstruction (TILT)…

These experiments allow 3D information to be acquired from a limited set of 2D experiments, permitting much faster collection of data.

3.2.9 4D Projection Reconstruction (TILT)…

These experiments allow 4D information to be acquired from a limited set of 2D experiments, permitting much faster collection of data.

3.3 13C and 15N Experiments Sub-Menus

3.3.1 N15 2D…

This menu gives different indirect detection experiments in which N15 is the indirect dimension.

3.3.2 N15 Coupling Measurements…

This menu gives a variety of experiments for measuring spin-spin couplings.

3.3.3 N15 Relaxation…

This menu offers different experiments for measurement of T1, T2 and T1rho as well as relaxation dispersion.

3.3.4 N15 Diffusion Measurements…

This menu permits selection of Diffusion Measurement.

3.3.5 C13 2D…

This menu offers experiments for 13C indirect detection and heteronuclear relaxation.

3.3.6 C13/N15 3D NOESYS…

This menu shows different combinations of HSQC, HMQC and NOESY involving 13C and 15N filtering or editing.

3.3.7 C13/N15 3D TOCSYS…

This menu shows different combinations of HSQC, HMQC and TOCSY involving 13C and 15N filtering or editing.

3.3.8 C13/N15 Projection Reconstruction (TILT)…

This menu shows different 3D heteronuclear Projection Reconstruction TOCSY/NOESY/ROESY HSQCs.

3.3.9 C13/N15 4D…

This menu displays a list of 4D sequences involving HMQC, HSQC and NOESY.

3.4 RNA/DNA Experiments Sub-Menus

3.4.1 N15 2D…

This menu gives different indirect detection experiments in which N15 is the indirect dimension.

3.4.2 C13 2D…

This menu offers experiments for 13C indirect detection and heteronuclear relaxation.

3.4.3 C13/N15 3D…

This menu shows different combinations of HSQC, HMQC, TOCSY and NOESY involving 13C and 15N filtering or editing.

3.4.4 Water Suppression…

This menu displays a list of 1D water suppression experiments (presat, Watergate, WET …).

3.4.5 Homonuclear 2D…

This menu displays a list of 2D water suppression experiments (presat, Watergate, WET …).

3.4.6 Heteronuclear 3D…

This menu shows a list of 3D assignment experiments for polynucleotides.

3.5 C13 Observe Experiments

This menu offers a variety of 1D, 2D and 3D 13C observe experiments for biomolecules.

After selection of the experiment the VnmrJ layouts for the parameter panels are active for the selected pulse sequence.

Chapter 4 The Acquire Folder

4.1 Action Bar Operations

Once the Acquire Folder tab is selected there is a new set of “widgets” displayed just above the parameter panel in the “action bar”. These are controlled by the “acq_action.xml” file in templates/layout/default. The action bar has the following functions

Function	Description
“Setup RF”	This button runs the “su” command which sets the RF for each channel (to allow probe tuning) and VT (if the temperature is controlled by the local parameter “temp” instead of by the Spin/Temp page in the Setup Folder.
“Show Time”	This button runs the “time” command and displays the total time needed for the experiment with the current parameter values.
“FID Scan”	This button initiates acquisition with the current pulse sequence and parameter values. The data are not time-averaged, but phase-cycling is active. All processing and display are governed by the local parameter settings. These settings may be changed at any time during the real-time FID/spectrum display. Controls to modify the display are present on the Shim page in the Setup Folder. They are present only when the FID Scan is active. Choices are FID or Spectrum display. When the FID is displayed there are options for real, imaginary, etc for the FID. If the Spectrum option is used, the spectrum is displayed in real-time, permitting shimming on a spectral feature. This can be quite valuable for small molecules in water where the D2O signal may be broad. FID Scan is stopped by pressing the same button.
“Lock Scan”	This button activates the normal pulsed lock display. The scan is stopped by pressing the same button.

Function	Description
"Acquisition Actions"	A drop-down menu is displayed when this is selected. When one of these options is selected the corresponding action is performed: "Start" Runs the macro "BPgo" which ends with the command "go" "Acquire and then Process" Runs the macro "BPgo" which ends with the command "ga" "Automation" runs the macro "BPgo" which ends with the command "au" "Start Manual Shim in "ss" Runs a macro that disables the gradients and runs the experiment in a mode that allows manual shimming of the lock signal. When the shimming is complete, the… "Stop Manual Shimming in "ss" Stops the acquisition and re-enables the gradients. This process permits adjustment of the shims to compensate for any de-shimming produced by decoupling (may be useful in Cold Probes when heteronuclear decoupling is used). "Array a Parameter" Starts up the Array popup window. "Use Pbox" Starts up the Pbox popup window.

4.2 Basic Page

Figure 8 shows the first panel in the Acquire Folder, the "Basic" page. This is optional and can be disabled by setting BPbasic=0 (or by using the checkbox in the "Globals&Probefile" page in the Setup folder). This panel shows buttons to automatically set up 1D, 2D or 3D experiments under default conditions (1ppm resolution in indirect dimensions with 8 transients per fid). The experimental time is indicated on the message line. The experiment is started by a "Begin" button. There is also capability for the user to change nt, ni and ni2 within this panel. The resulting data should be high-performance and accurate assuming that the probefile is accurate for the current sample.

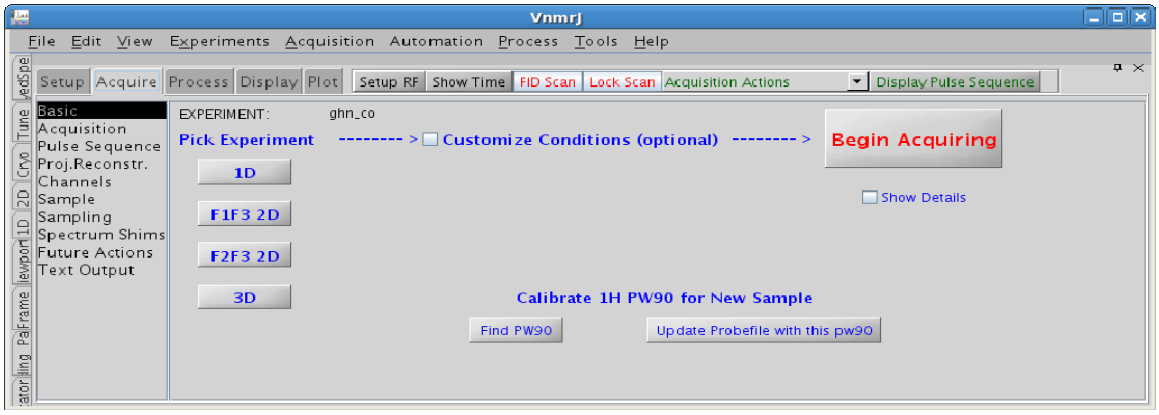


Figure 8 The Basic Page

4.3 Acquisition Page

The next page in the list is the “Acquisition” page.

This details important parameters controlling the spectral windows in each dimension, the variables controlling resolution in those dimensions, and parameters controlling the mode of nD data collection. Also given are parameters governing signal strength, relaxation delay and, number of transients per FID.

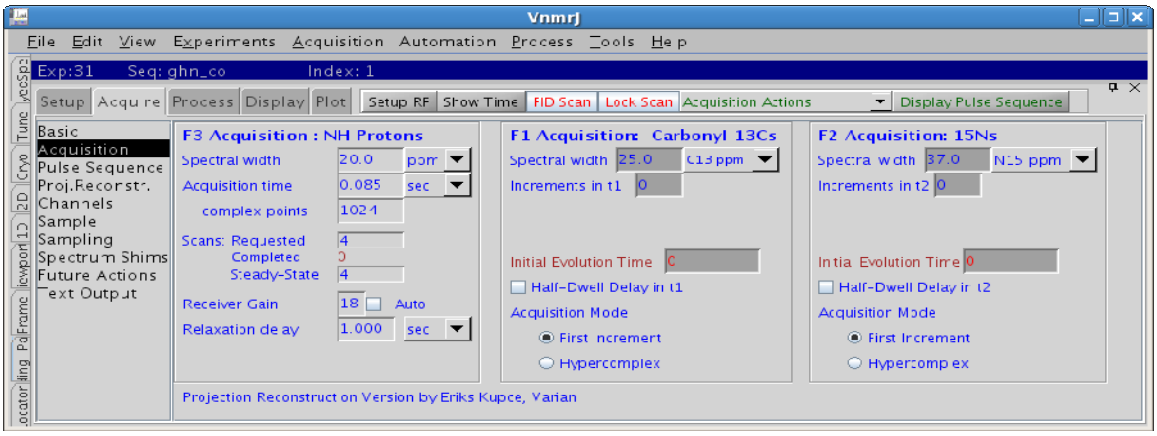


Figure 9 The Acquisition Page

Note that the titles are descriptive for the experiment and that the spectral windows can be entered in either Hz or ppm. The F3 acquisition panel is essentially the same for all pulse sequences since all sequences need parameters controlling direct detection. The number of boxes is governed by the number of dimensions in the experiment. If a value >1 is entered for a number of increments, the maximum evolution time is displayed for that dimension, as well as the resolution determined by that maximum evolution time.

4.4 Pulse Sequence Page

The Pulse Sequence page details the specific parameters needed for the experiment. Other parameters may be active but may not need to be displayed. Their values may be obtained by using the “Display Pulse Sequence” action button or by interrogation via the command line.

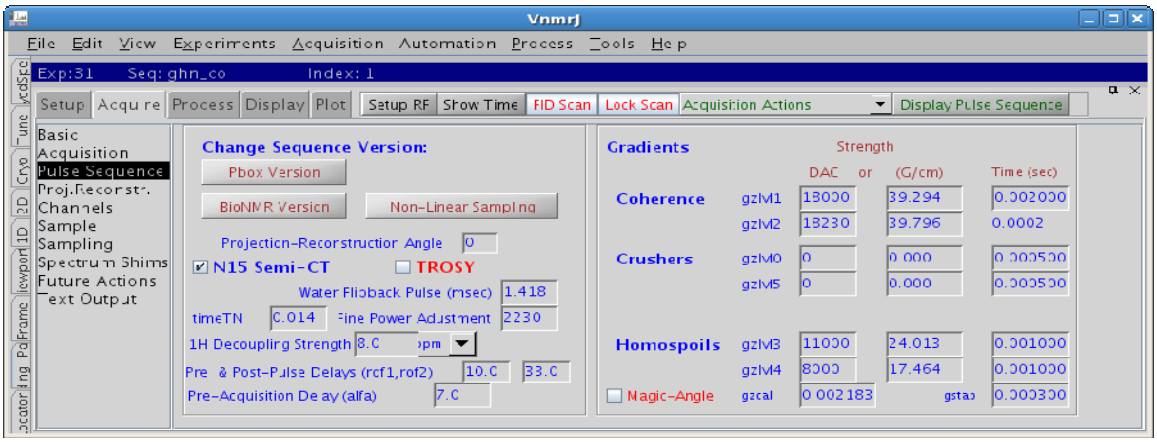


Figure 10 The Pulse Sequence Page

This panel is the vehicle for changing to different versions of the same basic experiment. These may have been written using different styles or capabilities. For example, the Pbox version (ghn_coA.c) and BioNMR version (ghn_coP.c) both do HNCO, but they both use Pbox internally to calculate shaped pulses “on-the-fly”. The Non Linear Sampling version just switches seqfil (the variable containing the sequence name) to ghn_coNLS to use the (Wagner) NLS approach.

4.5 Project Reconstruction

If the sequence does Projection Reconstruction as an option, there are one or more angles to be set. In this case the variable “**pra**” is set via an entry box. If the angle is between 0 and 90 degrees, both t1 and t2 co-evolve in order to get 3D information from 2D experiments.

Options are usually presented as checkboxes (N15 Semi-CT and TROSY, as well as magic-angle coherence transfer gradients when using a triax probe).

4.6 Channels Page

The setup macro normally sets all the channel-specific parameters such as offsets, powers, etc. The values can be viewed on the Channels page, see Figure 11.

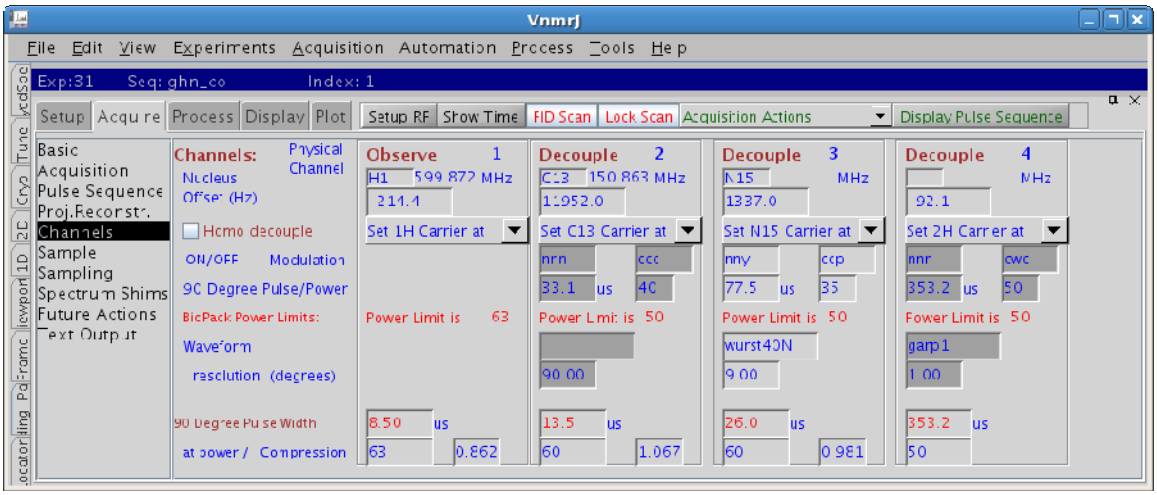


Figure 11 The Channels Page

This page is similar to that in the standard VnmrJ interface, but differs in that it shows the values of the BioPack power limits (if BPpowerlimits=1, i.e. the checkbox for BioPack power limits is checked in the Globals&Probefile page in the Setup Folder) and there are menus for selection of carrier positions for the RF channels. These menus use the “setoffset” macro (in /vnmr/maclib) to permit direct setting of the carrier offsets (tof, dof, dof2 and dof3) to specific chemical shift values.

The page also differs in that it shows the BioPack calibrations at the bottom (pw/tpwr/compH, pwC/pwClvl/compC, pwN,pwNlvl,compN and 1/dmf3(=pw90 at dpwr3),dpwr3).

Some sequences have enough parameters that there is not enough space in the Pulse Sequence page to hold the gradient parameters. In these cases there are separate Gradients pages.

In principle, the files in templates/layout (named by sequence name) could each have separate files for each page listed at the left of the page. However, apart from the Acquisition and Pulse Sequence pages, most experiments can share a common set of files in templates/layout/default. So, for example, the Sample, Sampling, Spectrum Shims, Future Actions and Text Output pages are stored in templates/layout/default. Certain other files for 2D, 3D or 4D (for processing, display and plotting) are in other directories in templates/layout such as default2DBP, default3DBP and default4DBP.

The main user interaction in setting up an experiment and starting acquisition is with the Acquisition and Pulse Sequence pages. The setup macros set all the necessary parameters in the Channels page (for normal operation). The other pages in the Setup Folder are described below.

4.7 The Sample Page

Figure 12 shows the Sample page. This page allows entry of values for parameters that are not normally changed, but may need to be changed for certain operations.

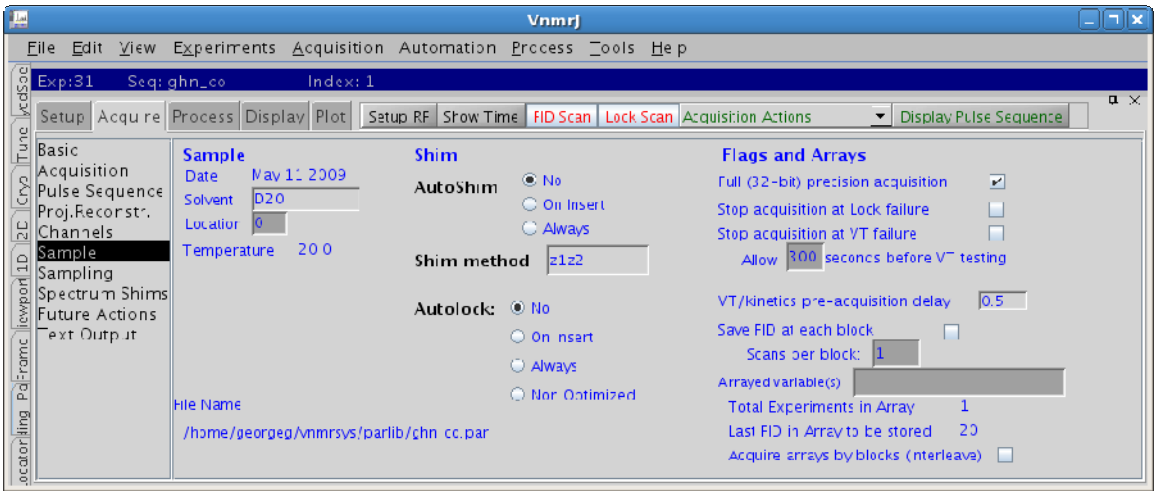


Figure 12 .The Sample Page

The AutoShim and AutoLock sections refer to autoshim on the lock signal using “background” method shimming.

4.8 The Sampling Page

Figure 13 shows the Sampling page.

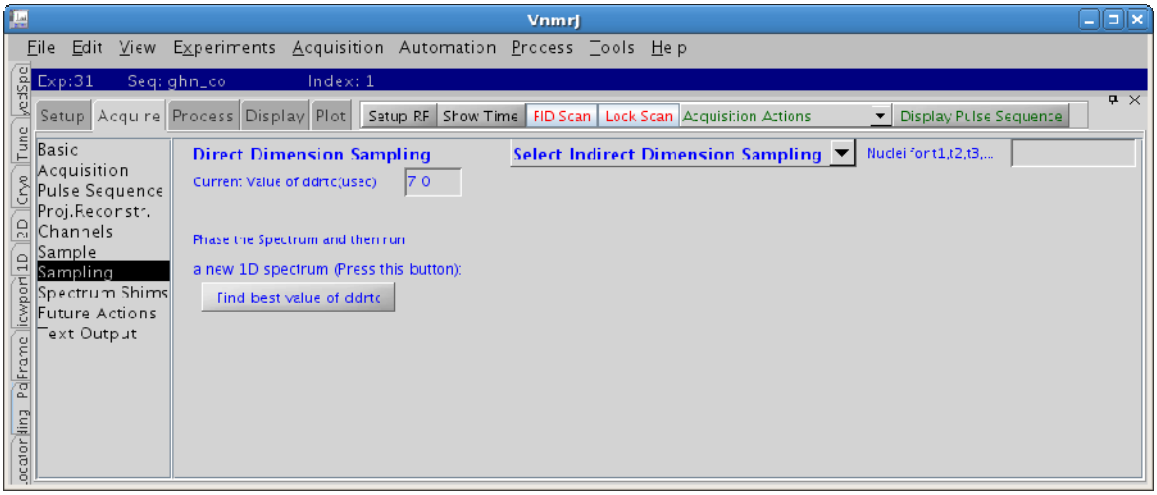


Figure 13 The Sampling Page

The parameter “ddrtc” is displayed. For convenience, after a 1D spectrum has been phased and it has a non-zero value of “lp”, the “Find best value of ddrtc” button will re-acquire the data and after transform the spectrum should be able to be re-phased with rp only and have lp=0. This new value of ddrtc can then be put into the probefile with the newly displayed “Update the probefile with this value” button.

The “Select Indirect Dimension Sampling” menu is used for Non-Linear Sampling and Automatic Spectral Compression in multi-dimensional experiments (see Chapter 16 for details).

4.9 The Spectrum Shims Page

Figure 14 shows the Spectrum Shims Page. This page shows the shim values used for the current data collection. They are stored in the saved parameter set when the FID is saved.

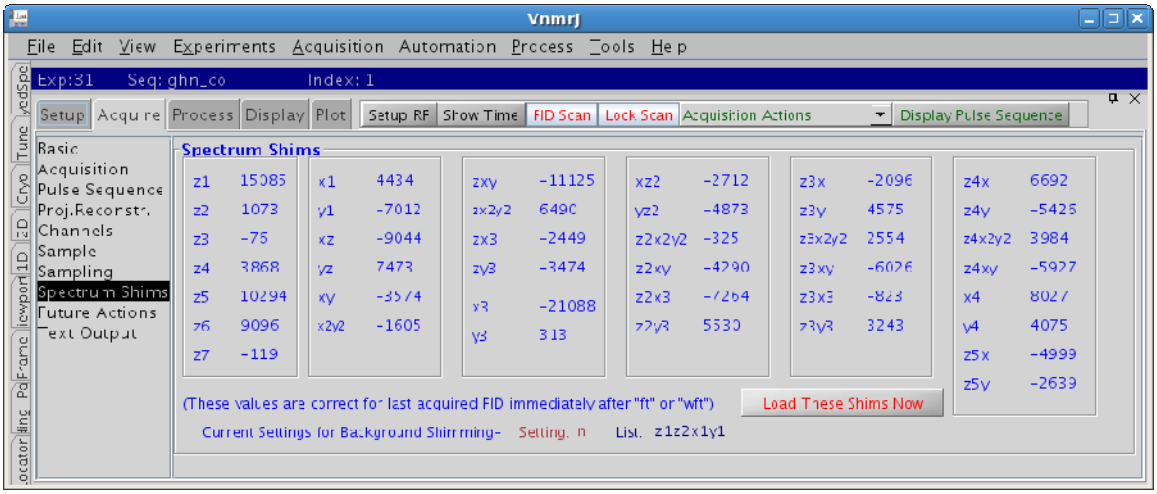


Figure 14 The Spectrum Shims Page

When a archived data set is recalled, the shims are displayable on this page. Those shims values can be installed in the hardware by using the “Load These Shims Now” button.

4.10 The Future Actions Page

This page is shown in Figure 15. The values for the parameters “`wexp`”, “`wbs`”, “`wnt`” and “`werr`” are displayed here. These values are for actions done during and at the end of acquisition. These parameters have no action under the “go” and “ga” commands, and are only active with the “`au`” command.

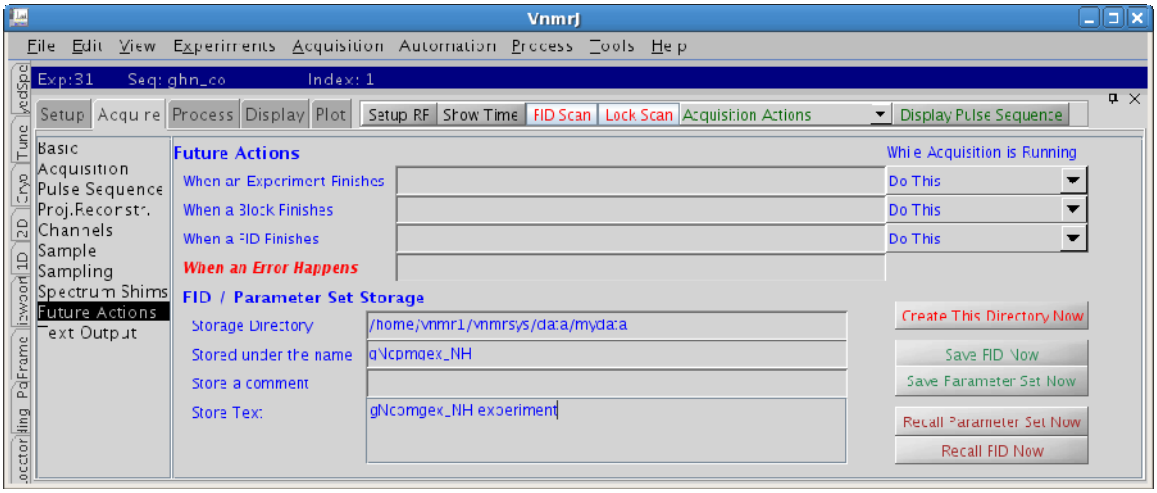


Figure 15 The Future Actions Page

After acquisition has started the *commands* “wexp”, “wbs”, “wnt” and “werr” are executable by using the menus. These are used to execute the corresponding entered value of the same parameters. The menu at the right permits the user to change the action, stop a planned action, or do nothing.

The other function of this page is for data or parameter set storage or retrieval. The “Storage Directory” entry determines the value of the global variable “svfdir”, while the “Stored under the name” entry sets the global variable “svfname”. These are also the variables used in the Save Data Setup accessed via the Main Menu.

If the Storage Directory does not exist it can be created by entering the value and then using the “Create This Directory Now” button.

Saving or recalling data or parameter sets is controlled by a set of buttons. The proper value of the archive directory must be entered before using a “Recall” button. If the action of saving a file would overwrite an existing file the user must enter ‘y’ or ‘n’ to the question displayed in the message line.

Comments may be entered. These are stored in a text variable “comment” and are different from the normal “text” file. Comments are searchable within the Locator database and can help to group data by comment in the Locator.

The “Store Text” field is used for modifying the contents of “text” within the active experiment (curexp). This file is saved along with the binary fid file and parameter set when the FID is archived.

The “Save FID Now” button should be the normal way a FID is archived. This button runs the BPsvf macro which runs the VnmrJ standard macro “svf”, but also archives other important information within the stored .fid file: the pulse sequence source code, the probefile, the global file and all waveforms used in the experiment **(***.RF and *.DEC files). This way, data files possess documentation of anything that could affect the NMR performance.

Similar capabilities are present in other “File Save” pages within the Display or Plot Folders.

4.11 The Text Output Page

Figure 16 shows the Text Output Page. This page includes a “textfile” widget that is used to display text produced by VnmrJ commands.

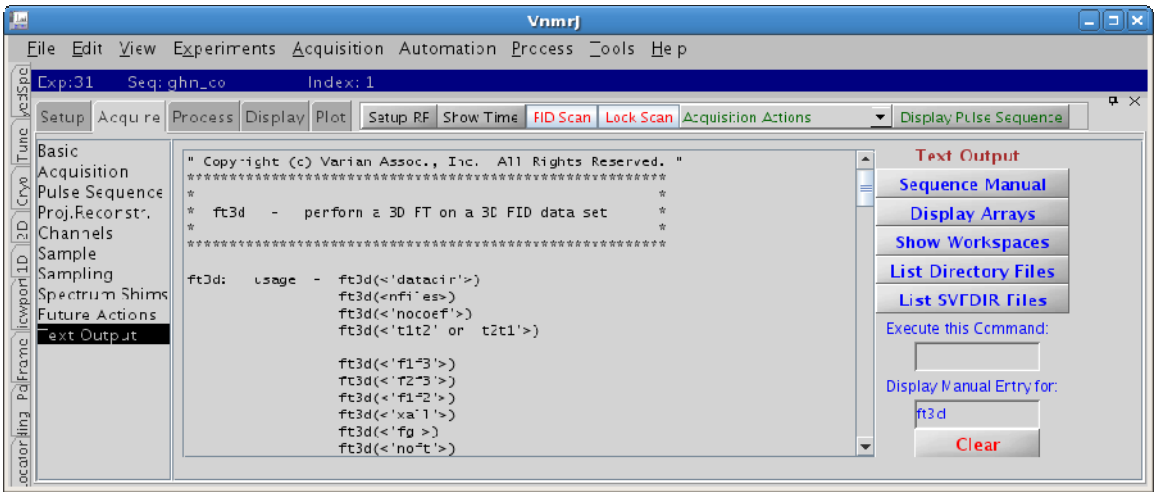


Figure 16 The Text Output Page

- Buttons at the right permit useful text displays.
- The “List Directory Files” button does a Unix/Linux “ls” on the current directory. “List SVFDIR Files” does an “ls” on whatever directory is specified by the global variable “svfdir” (see Future Actions Page).
- A “mini” command line is also given. Just enter a text string and then the return key.
- A manual entry from /vnmr/manual can be displayed just by entering the name followed by the return key. Here, the ft3d command is described by using this entry box.
- The “Clear” button clears the text window.

Chapter 5 The Process Folder

The Process Folder contains pages for processing 1D, 2D and 3D data. The pages displayed are governed by the default library specified by the sequence-specific layout file in templates/layout. The actions buttons on the Action Bar are appropriate to data processing and display:

Button	Description
"Adjust Weighting"	Runs the VnmrJ command "`wti`" that puts up an interactive display in the graphics area to permit mouse-controlled selection of weighting functions.
"Transform"	Does an "ft" of the (first) FID.
"Weighted Transform"	Does a "wft" of the (first) FID.
"Display Spectrum"	Runs the VnmrJ command "ds" which shows a 1D spectrum.
"Stacked Display"	Runs the VnmrJ command "dssh" which stacks spectra horizontally.

5.1 The Direct Dimension Process Page

Figure 17 shows the panel used for processing the direct dimension. It is common for 1D, 2D, 3D and 4D data sets. The page labels will differ depending on dimensionality, but the direct dimension processing page always has the same content.

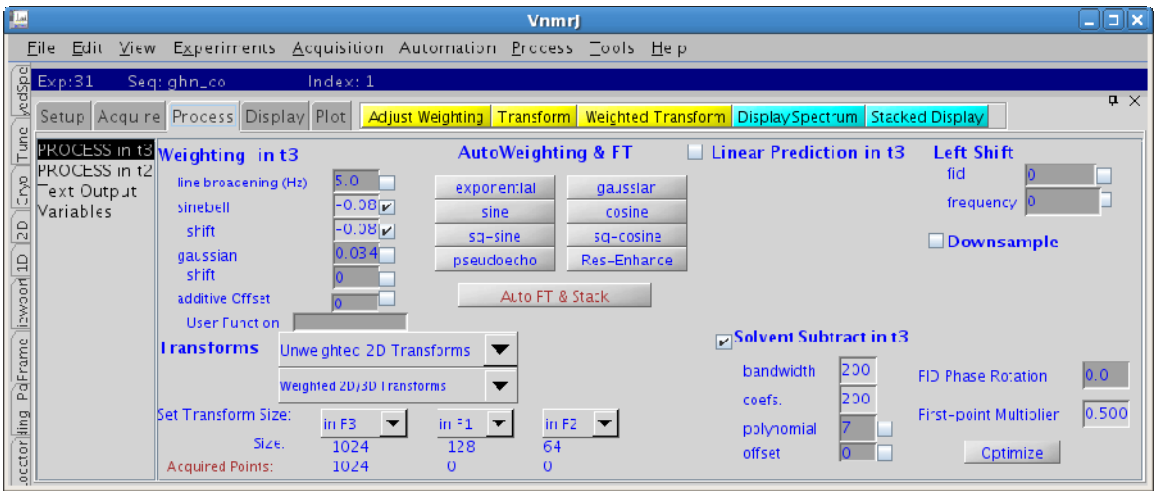


Figure 17 The PROCESS in t3 Page

Weighting functions are displayed at the left. The page includes options for linear prediction, left shifting of fids or spectra, downsampling, phasing of the fid, first-point multiplier (with a button to do find the best value) and digital filter line suppression.

A series of buttons set the desired type of weighting function, matched to the acquisition time “at”, does the wft and displays the spectrum. If the experiment involves an array of a parameter, the “Auto FT & Stack” does a wft(‘all’), displays the data in a side-by-side stack, on-scale, and analyzes the data for the maximum peak height. It also displays the values of the arrayed parameter at the bottom with a determined best value.

The number of acquired complex points in each dimension is shown along with the transform size selected. Menus are provided to change the transform size.

Two menus are provided to execute transforms. The only differ in the inclusion or absence of weighting functions. The options are transforms of first fid, all fids, interferogram along t1 or t2, F1F3 2D FT, F2F3 2D FT, same but with DC correction, same but with BC correction, full 3D FT, full 3D FT followed by plane extraction of F1F3 planes, same but for F2F3 planes, same but for both F1F3 and F2F3 planes and same but for F1F3, F2F3 and F1F2 planes.

When a parameter is arrayed, an “Array” page appears in the list. It has the same functionality as the PROCESS in t3 page, but it has buttons to do transforms of the arrayed data, including individual array member transforms.

5.2 The Process in t2 Page

This page is displayed in Figure 18 and is very similar to the analogous “Process in t1” page used for 2D,3D and 4D layouts.

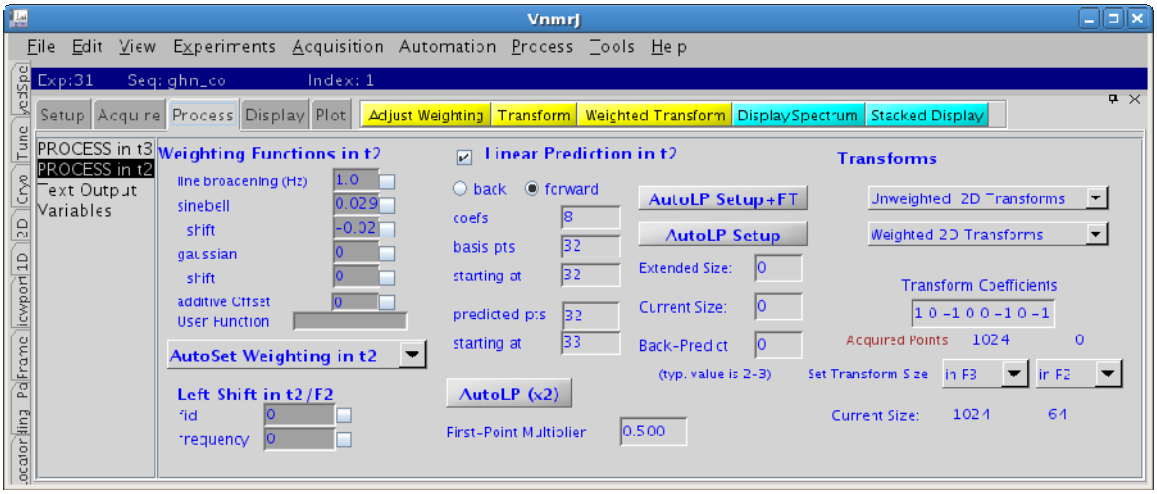


Figure 18 The PROCESS in t2 Page

Similar capabilities are given here as in the PROCESS in t3 page. The associated parameters (for 3D) would be lb2 instead of lb, for example.

The major differences include the menu-driven choice of weighting functions matched to t2max (ni2/sw2) governed by the menu “AutoSet Weighting in t2”, enterability of the “f2coef” parameter for phase-sensitive 2D/3D processing (whose value is set by the experiment setup macro), menus for transforming along t2 and automatic linear prediction parameter setting. The transform menus have internal logic to determine the correct arguments for wft2da for both 2D and 3D data sets.

Button	Description
“AutoLP (x2)”	This is used when an experiment has run to completion so that the number of increments is known. It sets the LP parameters for doubling of the data size along the indirect dimension and sets the weighting function for time-constants matched to the extended data set. Use this button for a quick setup of parameters.
“AutoLP Setup”	The button uses the three entry fields below it to set up LP parameters. The user chooses the increments from which to extend and the total number of increments in the extended data set. Optionally, a number of points at the beginning of the data set can also be linear predicted (to possibly correct for distorted first points). All LP parameters will be used as part of the FT process. Data sets are typically extended by a factor of two to four. This button is useful during data acquisition to view the 2D spectrum (albeit at lower resolution). The “Current Size” value would typically be the number of fids acquired divided by two (by four for a 3D experiment).
“AutoLP Setup + FT”	The button couples the “AutoLP Setup” with the 2D FT. This button is only valid for 2D experiments, while the “AutoLP Setup” button can be used for 3D experiments, followed by usage of a transform menu selection.

5.3 The Variables Page

This page is shown in Figure 19. It permits entry of parameter values for variables stored in the parameter table, both real numbers and text strings. They are useful for temporary storage of values from calculations and are often used in macros.

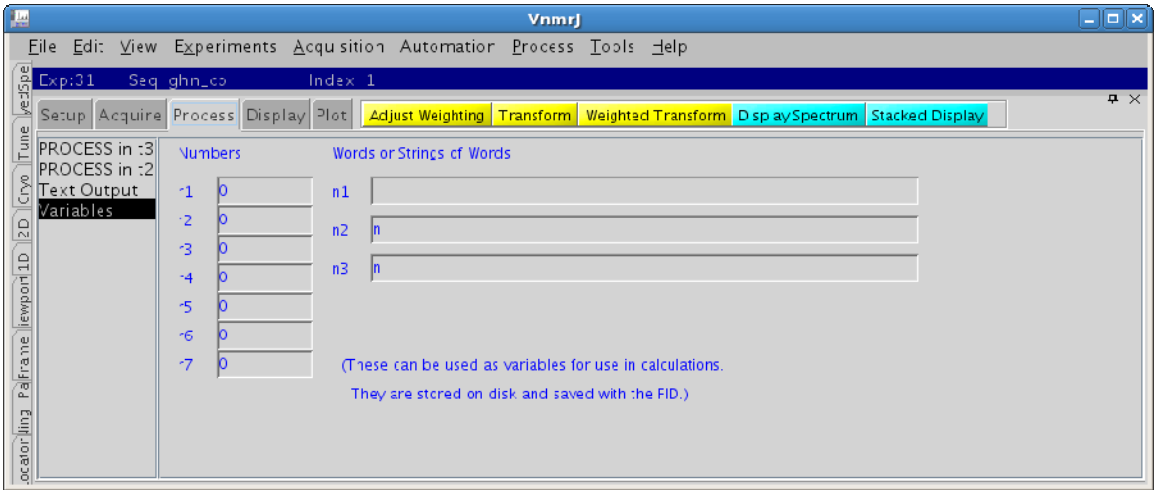


Figure 19 The Variables Page

Chapter 6 The Display Folder

The Display Folder contains pages for displaying 1D, 2D and 3D data. The pages displayed are governed by the default library specified by the sequence-specific layout file in templates/layout. The actions buttons on the Action Bar are appropriate to data processing and display:

Button	Description
“Transform”	does an “ft” of the (first) FID.
“Weighted Transform”	does a “`wft`” of the (first) FID.
“Display Spectrum”	runs the VnmrJ command “`ds`” which shows a 1D spectrum.
“Stacked Display”	runs the VnmrJ command “`dssh`” which stacks spectra horizontally.
“Display Text”	displays the current parameter set text file.

6.1 Display Folder Pages

Figure 20 shows the panel used for displaying 1D spectra. This page is useful for rapid setting of display modes, sizes, scales, referencing, etc.

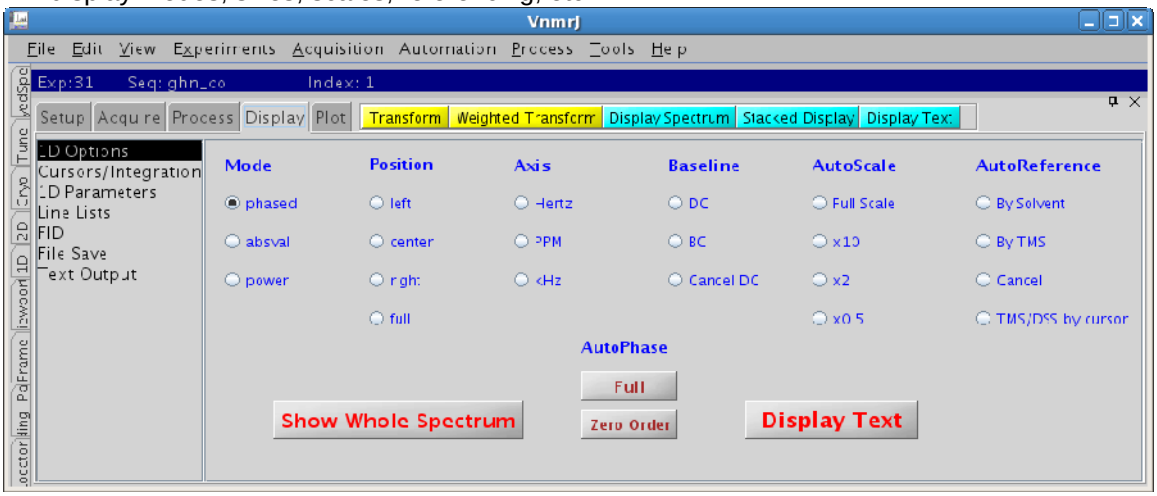


Figure 20 The 1D Options Page

The Cursors/Integration page gives options for adjusting the observe transmitter position, putting the cursor on the nearest line, showing the linewidth and setting homodecoupling offsets. Two cursors are used on the spectrum to measure S/N, to reset spectral width and to inset a display. There are several buttons to affect the integral display.

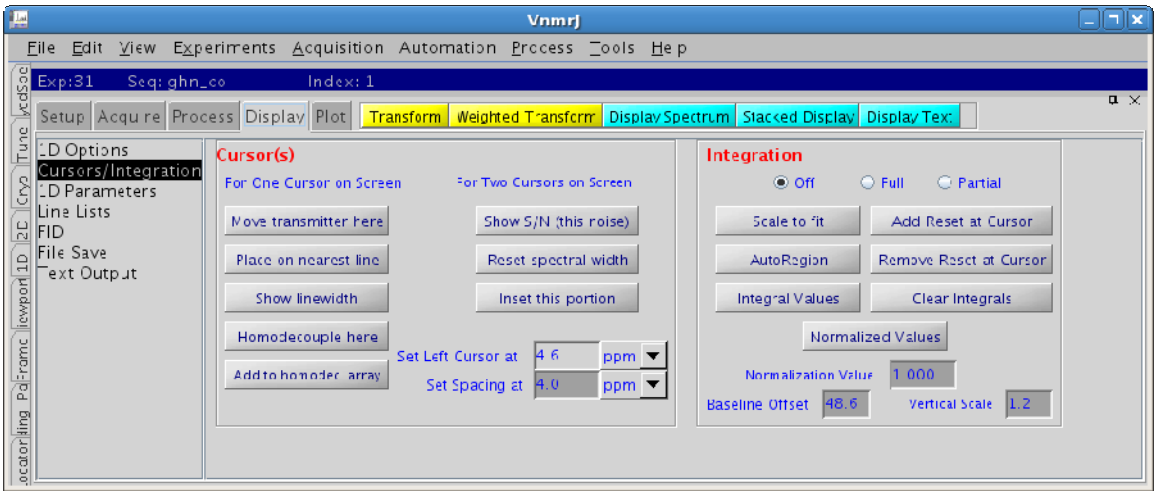


Figure 21 The Cursors/Integration Page

Figure 22 shows the 1D Parameters page which gives display and entry capability for a variety of parameters (in some lists this page is called “More”). Referencing at an arbitrary position is possible by placing the cursor on a spectral line, entering the chemical shift and clicking on the “Reference Now” button. Baseline correction using polynomials is selectable (integral regions need to be defined with alternating noise and spectral features so that the BC program may use the noise regions in the correction process). The “Save/Recall Displays” menus permit the saving of display parameters (up to nine separate displays) with the ability to recall and redisplay. These may be copied to another Experiment for use when the spectral widths are identical.

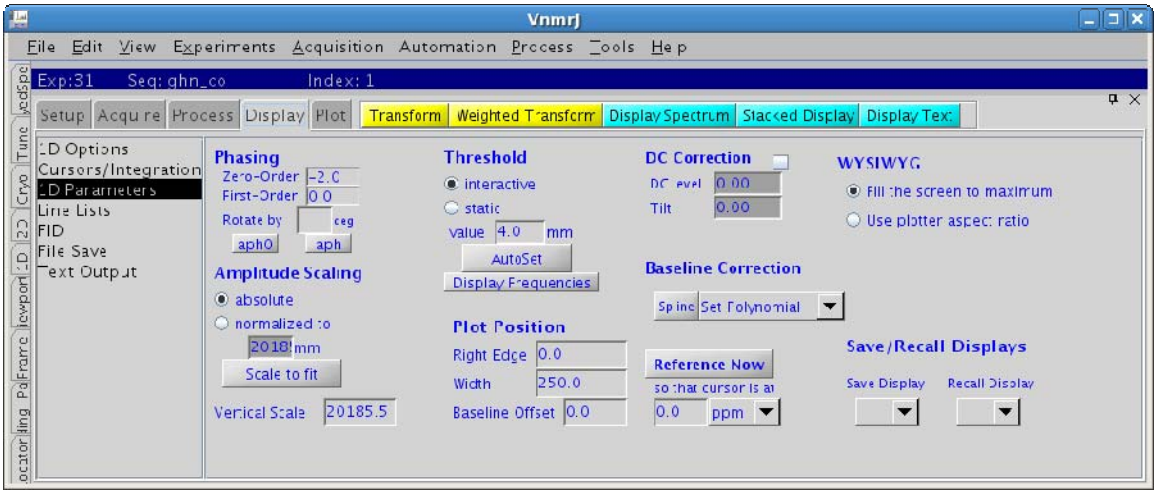


Figure 22 The 1D Parameters (or More) Page

Figure 23 shows the Line Lists page. This is used to display the results of line lists and integral lists.

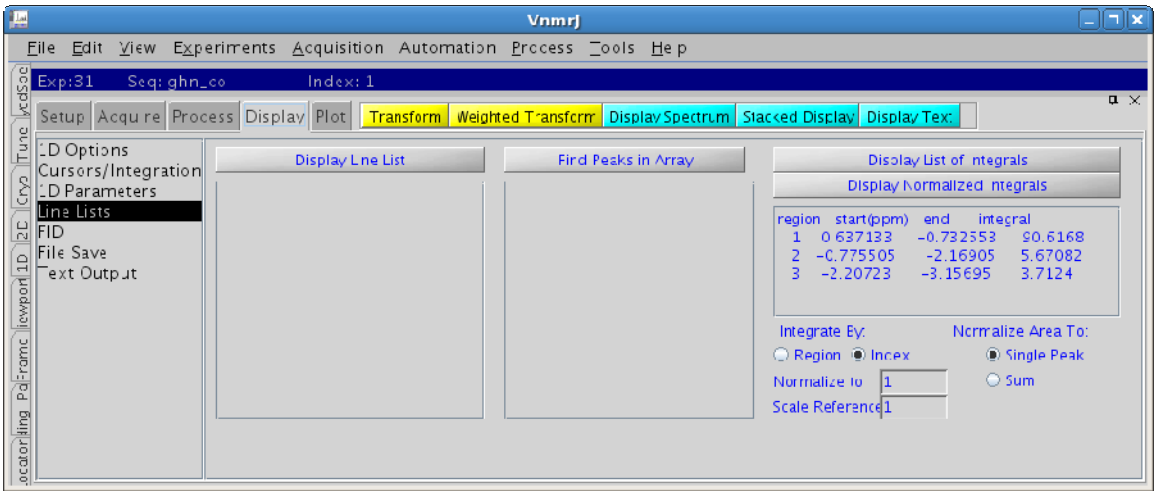


Figure 23 The Line Lists Page

While the primary method of displaying arrays is the ArrayedSpectra Vertical Panel, the Spectral Arrays page (in Figure 24) permits extensive user control of the display of arrayed data.

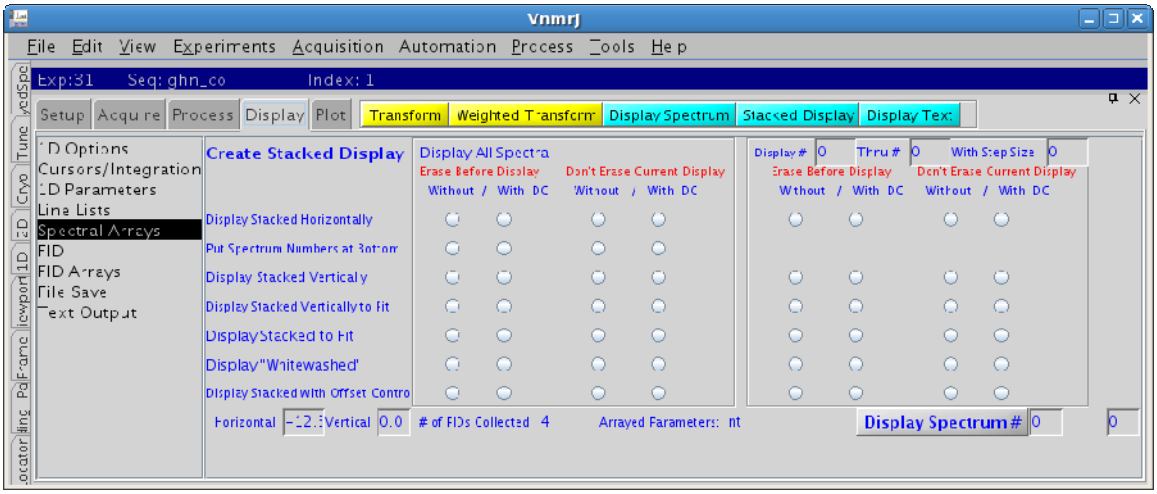


Figure 24 The Spectral Arrays Page

This page shows a vertical list of different types of displays. For these types there is a choice of showing all spectra (left group) or displaying starting with one spectra and ending with another, skipping some (right group). For example, in this case spectra 1, 3,5,7 and 9 could be displayed. There is also a “Display Spectrum #” button that displays the spectrum indicated by the index at the right.

Within these two groups there are two columns for the option of having or not having a “dc” correction performed on each spectrum before display (this “dc” correction is only for the purposes of display. The stored data is unaffected). Finally, there is another set of two columns that do the same thing, except there is the choice to erase or not erase the screen before doing the display.

The FID page has parameters for FID display. The File Save and Text Output pages are similar to those described above. Display of multi-dimensional data is typically governed by a panel names F1F3 or F2F3, for example (Figure 25).

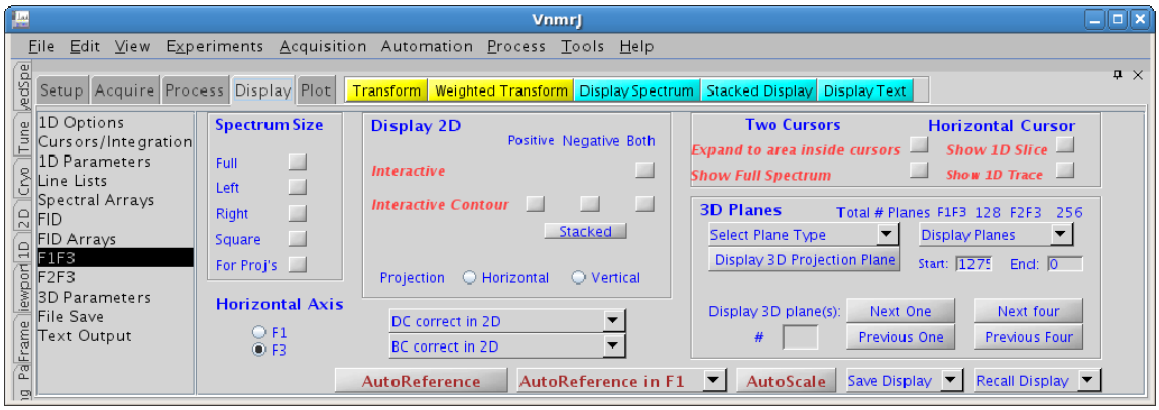


Figure 25 The F1F3 2D Display Page

The spectrum size can be set and displayed by checkboxes. Display of 2D data can be done for positive, negative or both kinds of peaks. Automatic referencing is available in both direct and indirect dimensions. If a 3D transform has been performed and planes extracted, the orthogonal projection planes can be displayed. Individual or multiple planes can be displayed.

Chapter 7 The Plot Folder

The Plot Folder contains pages for ploting 1D, 2D and 3D data. The pages displayed are governed by the default library specified by the sequence-specific layout file in templates/layout. The actions buttons on the Action Bar are appropriate to data processing and display:

Button	Description
“Show Time”	Runs the VnmrJ “time” command to show total experiment time. This button is useful when the total experiment time is needed for inclusion in the text file.
“Display Spectrum”	Runs the VnmrJ command “ds” which shows a 1D spectrum.
“Stacked Display”	Runs the VnmrJ command “dssh” which stacks spectra horizontally.
“Display Text”	Runs the VnmrJ command “dtext” which puts the text file in the graphics area.
“Cancel”	Aborts currently running commands or macros.

7.1 Plot Folder Pages

Figure 26 shows the panel used for plotting 1D data. This is the first panel displayed and serves as a reminder to annotate the spectrum text file with pertinent information. This should be done before the FID is saved using the File Save page.

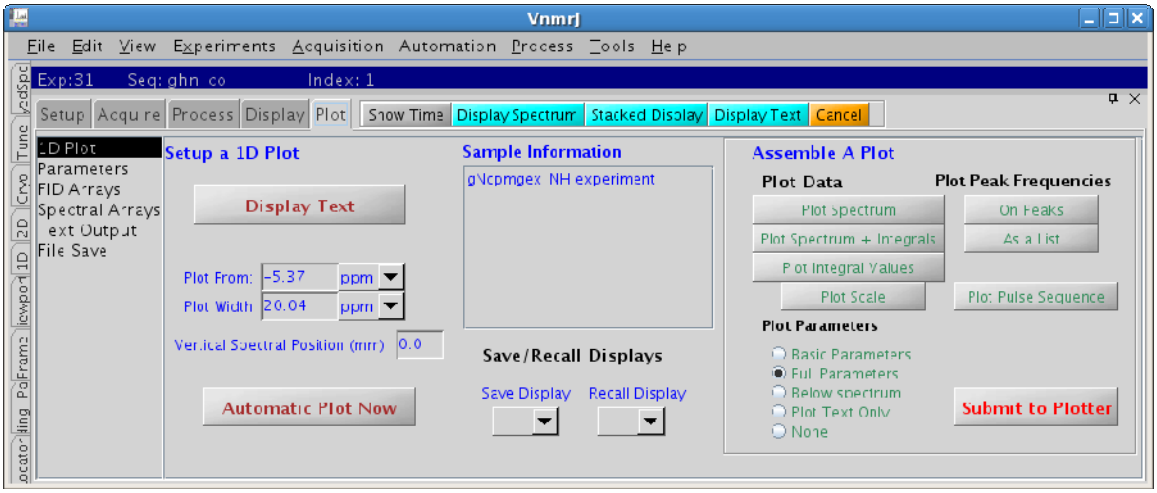


Figure 26 The 1D Plot Page

A 1D plot is typically constructed by using the (green) plot buttons, followed by selection of the type of parameter set plot type and submission to the plotter. The “Submit to Plotter” button will construct a “RAST” type of file using the printer given by the printer variable. If plotting to a file or other types of plotting is desired consult the VnmrJ manual.

The Save/Recall menus permit rapid recall of saved displays for plotting a series of expansions, for example.

Figure 27 shows the page for plotting 2D data from 2D or 3D experiments.

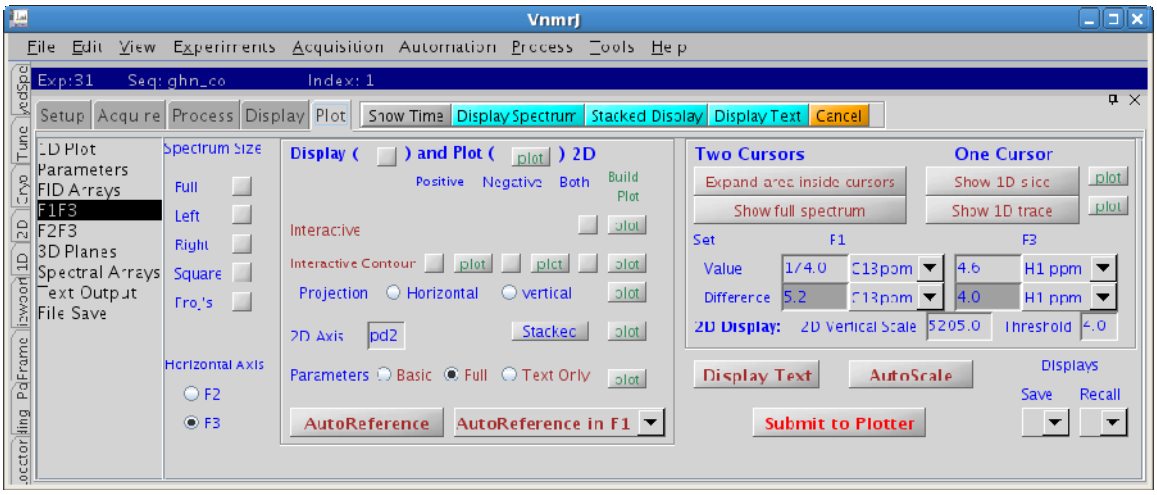


Figure 27 The F1F3 Plot Page

This page has similar functions as the Display/F1F3 2D page, but it also has (green) plotting buttons. Both display (empty buttons) and plotting (buttons with “plot” in them) are shown. The display may be created and examined on the screen and, if desired, plotted by using the associated plot button. A short form, full parameter set plot or text only choice is made by a radio button followed by the associated plot button to add it to the plot. If a slice of the 2D data is displayed it may also be plotted.

After all plotting choices are made; the “Submit to Plotter” button initiates the plotting.

A specific region of the spectrum can be displayed (and then plotted) by the entry boxes for F1 and F3.

The “Submit to Plotter” button will construct a “RAST” type of file using the printer given by the printer variable. If plotting to a file or other types of plotting is desired consult the VnmrJ manual.

The Save/Recall menus permit rapid recall of saved displays for plotting a series of expansions, for example.

Figure 28 shows the 3D Planes plot page. Again, the green buttons are involved with plotting. Display functions similar to above pages are also available but this page is only used for plotting 2D spectra from the extracted planes or projections of a 3D experiment.

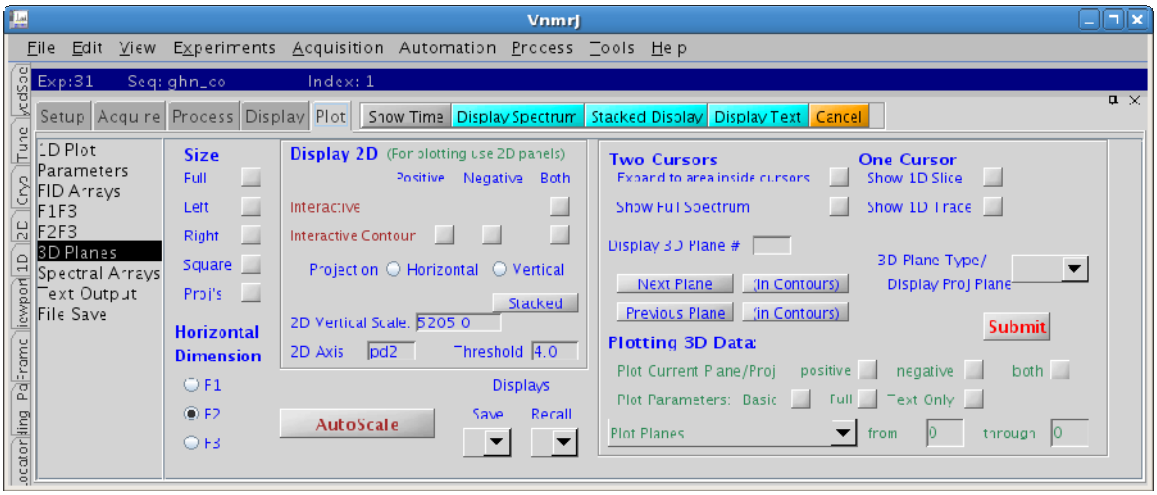


Figure 28 The 3D Planes page

The current displayed plane or projection is plotted using the checkboxes next to “Plot Current Plane/Proj” followed by the “Submit” button. Parameters can be plotted as well by selecting the type of plot before using “Submit”.

Multiple plane plotting (six to a page) is done via the menu. The entry boxes at its right ar for specifying a starting and ending plane. The “3D Plane Types” menu allows switching between plane types such as going from F1F3 to F2F3, for example.

The File Save and Text Output pages are similar to those described above.

Chapter 8 Experiment Setup Macros

Since macros are the software elements that actually do the work, it is instructive to look in detail at the macros involved in setting up an experiment. For this, consider the C13 NOESYHSQC experiment mentioned above. The menu used to select this experiment is present as a text file. In VnmrJ it is an XML submenu in /vnmr/biopack/templates/vnmj/interface for a /vnmr installation, or in ~/vnmrsys/biopack/templates/vnmrj/interface for a user installation. Once the selection is made, in either interface, the gnoesyChsqc macro is executed. The macro is a text file present in /vnmr/biopack/maclib and detailed in the following sections.

8.1 The gnoesyChsqc macro

"gnoesyChsqc macro, 1.1 06/29/00 Copyright (c) 1999-2009 Varian, Inc. All Rights Reserved."

BPrtppar('gnoesyChsqc')

getparam('dmf30','C13'):dmf30 “**sets parameters for STUD decoupling**”

getparam('rf30','C13'):rf30

getparam('dmf80','C13'):dmf80

getparam('rf80','C13'):rf80

sw2=80d

getparam('dmm2NH','N15'):$mode “**gets default value for N15 decoupling**”

substr($mode,3,1):$mode

dm2='nnnn' dm2='nnnn'

dmm2='ccc'+$mode

dof=dof-(174-43)*dfrq **“sets C13 offset to 43ppm”**

BPsetwurstparams “**sets parameters for adiabatic decoupling**”

“ BPsetwurstparams is described in section 10.4

if dmm<>'cccp' then “**sets parameters for non-adiabatic decoupling**”

dpwr=pwClvl-15 dmf=1e6/(2*2*1.4*pwC*compC)

BPcheckdpwr “**readjusts conditions if above optional limit**”

endif

getparam('pwC3db','C13'):pwClw “**power for simultaneous 15N/13C pulse**”

pwNlw=1.4*(compN*pwN) “**power for simultaneous 15N/13C pulse**”

CNrefoc='n' COrefoc='n'

getparam('gt5','N15'):gt5

getparam('gt0','N15'):gt0

getparam('gzlvl5','N15'):gzlvl5

getparam('gzlvl0','N15'):gzlvl0

getparam('NHgstab','N15'):gstab

dm='nnny' "**sets C13 decoupling in acquisition**"

pwC10 = 80.5*600.0/sfrq **"recalculates shaped pulse width"**

$pw=pwC10*5.0 $pw=2.0*trunc($pw/2.0)+1.0 pwC10=($pw + 1.0)/5.0

spin='n' sw2=80d "**sets up 13C indirect dimension spectral width**"

BPmake180COa

ni=0 ni2=0 phase=1 phase2=1 **"sets up for 1D spectral check"**

Let us now examine the elements of the gnoesyChsqc macro in detail. First, a general parameter set recall macro BPrtppar is run (see next section) specifying the gnoesyChsqc parameter set. Note that in most cases the parameter set has the same name as the pulse sequence. The parameter set can be present in either /vnmr/biopack/parlib, /vnmr/parlib, ~/vnmrsys/biopack/parlib or in the user's parlib, depending on user preference. The order of search for the parameter set is determined by the order in the applications directory (accessible via the MainMenu..Edit…Applications popup).

8.2 The BPrtppar macro

"@(#)BPrtppar 1.1 06/29/00 Copyright (c) 1999-2009 Varian, Inc. All Rights Reserved."

if $#>0.5 then

BPgetpars($) **"get parameter set"**

BPfixup **"creates any missing parameters"**

if (BPinstall=1) then **"only true after installation"**

BPupdate_from_probefile "**gets common calibrations from probefile**"

BPcheck "**checks powers against user-defined power limits**"

BPsetddrtc "**ddrtc has already been set by BPupdate_from_probefile**"

"this sets ddrtc=alfa for certain sequences ending in an echo"

BPflipbacks "**Get water flipback pulse calibrations**"

BPfixrefs **"fixup referencing"**

BPfixup2 **"final check on parameters"**

else

banner('BioPack not properly installed(BPinstall=0).\\Parameter set

has been loaded,\\but no values have been loaded from the

probefile.\\Check/change all parameter values manually')

endif

endif

man(seqfil)

The first macro to run is BPgetpars which finds the parameter set (depending on where and how BioPack was installed) and loads it.

8.3 The BPgetpars macro

"@(#)BPgetpars 1.1 06/29/00 Copyright (c) 1999-2009 Varian, Inc. All Rights Reserved."

```
$file='' $file=$1                                   "Figure out if .par is part of name"
length($file):$size
if $size>4 then
$t=''
substr($file,$size-3,4):$t
if $t<>'.par' then
$file=$file+'.par'
endif
else
$file=$file+'.par'
endif
$name='' $name='/vnmr/maclib/rtppar'                "Look for rtppar in maclib"
exists($name,'file'):$e
if $e=1 then                                "rtppar uses appdir path(VnmrJ2.2 or later)"
rtppar($file)                                                   "Recall parameter set"
else                                                "use either /vnmr or user path"
$fpath=userdir+'/parlib/'+$file
exists($fpath,'file'):$e
if $e<0.5 then
$fpath=systemdir+'/parlib/'+$file
exists($fpath,'file'):$f
if $f<0.5 then
write('error','%s: could not find \'%s\'',$0,$fpath)
return
endif
endif
rtp($fpath)                                         "Recall parameter set"
endif
exists('prescan','parameter'):$e
if($e>0.5) then destroy('prescan')endif
fixpar
```

The next macro to run is BPfixup which adds parameters if needed. This macro ensures that parameters used by following macros do exist.

8.4 The BPfixup macro

```
"@(#)BPfixup 1.1 06/29/00 Copyright (c) 1999-2009 Varian, Inc. All Rights Reserved."
                                                    "Add missing parameters to parameter set"
exists('probe','parameter'):$e
if($e>0.5) then destroy('probe') endif                          "probe should be a global"
exists('dimension','parameter'):$e
if $e=0 then create('dimension','string') endif
dimension=''
exists('compH','parameter'):$e
if $e=0 then create('compH','real') compH=1.000 endif
exists('pwC','parameter'):$e
if $e=0 then create('pwC','pulse') pwC=0.000 endif
exists('pwClvl','parameter'):$e
if $e=0 then create('pwClvl','real') pwClvl=0.000 endif
exists('pwN','parameter'):$e
if $e=0 then create('pwN','pulse') pwN=0.000 endif
exists('pwNlvl','parameter'):$e
if $e=0 then create('pwNlvl','real') pwNlvl=0.000 endif
exists('compC','parameter'):$e
if $e=0 then create('compC','real') compC=1.000 endif
exists('compN','parameter'):$e
if $e=0 then create('compN','real') compN=1.000 endif
exists('compD','parameter'):$e
if $e=0 then create('compD','real') compD=1.000 endif
exists('ref_pwr','parameter'):$e
if $e=0 then create('ref_pwr','pulse') ref_pwr=tpwr endif
exists('ref_pw90','parameter'):$e
if $e=0 then create('ref_pw90','real') ref_pwr=pw endif
exists('empty_noise','parameter'):$e
if $e=0 then create('empty_noise','real') empty_noise=0 endif
exists('sample_noise','parameter'):$e
if $e=0 then create('sample_noise','real') sample_noise=0 endif
exists('ddrtc','parameter'):$e
if $e=0 then create('ddrtc','pulse') endif
''ddrtc=0.0 in parameter set or probefile indicates non-vnmrs system''
if Console='vnmrs' then                                  “add parameters for homodecoupling”
```

```
exists('hdof','parameter'):$e
if $e=0 then create('hdof','frequency') hdof=0 endif
exists('hdpwr','parameter'):$e
if $e=0 then create('hdpwr','real') hdof=0 endif
exists('hdmf','parameter'):$e
if $e=0 then create('hdmf','real') hdmf=1000 endif
exists('hdpwrf','parameter'):$e
if $e=0 then create('hdpwrf','real') hdpwrf=4095 endif
exists('dutyc','parameter'):$e
if $e=0 then create('dutyc','real') dutyc=0.1 endif
exists('hdres','parameter'):$e
if $e=0 then create('hdres','real') hdres=0.9 endif
exists('rof3','parameter'):$e
if $e=0 then create('rof3','pulse') rof3=0 endif
exists('homorof1','parameter'):$e
if $e=0 then create('homorof1','pulse') homorof1=2 endif
exists('homorof2','parameter'):$e
if $e=0 then create('homorof2','pulse') homorof2=2 endif
exists('homorof3','parameter'):$e
if $e=0 then create('homorof3','pulse') homorof3=2 endif
exists('homo','parameter'):$e
if $e=0 then create('homo','flag') homo='n' endif
exists('hdseq','parameter'):$e
if $e=0 then create('hdseq','string') hdseq='' endif
exists('rof3','parameter'):$e
if $e=1 then destroy('rof3') endif
endif
exists('mag_flg','parameter'):$e
if $e=1 then
getparam('mag_flg','N15'):mag_flg
if (mag_flg='y') then BPcheckfortriax endif
endif
exists('gzsize','parameter'):$e
if $e=0 then create('gzsize','real') endif
exists('gzwin','parameter'):$e
if $e=0 then create('gzwin','real') endif
exists('nv2','parameter'):$e
```

```
if $e then destroy('nv2') endif
exists('nv','parameter'):$e
if $e then destroy('nv') endif
exists('ni','parameter'):$e if $e then fpmult1=0.5 endif
exists('ni2','parameter'):$e if $e then fpmult2=0.5 endif
exists('STUD','parameter'):$e
if $e=1 then STUD='n' endif
exists('SPARSE','parameter'):$e
if $e=0 then create('SPARSE','flag')
SPARSE='n'
endif
''set local value of samplename to global value''
exists('samplename_global','parameter','global'):$e
if $e=0 then create('samplename_global','string','global') endif
exists('samplename','parameter','global'):$e
if $e=1 then destroy('samplename','global') endif
exists('samplename','parameter'):$e
if $e=0 then create('samplename','string') endif
samplename=samplename_global
exists('SCT','parameter'):$e
if $e=1 then
SCT='y'
endif
exists('f1180','parameter'):$e
if $e=1 then
if f1180='y' then lp1=-180 rp1=90 else lp1=0 rp1=0 endif
endif
exists('f2180','parameter'):$e
if $e=1 then
if f2180='y' then lp2=-180 rp2=90 else lp2=0 rp2=0 endif
endif
cz setref
r1=0 r2=0 r3=0 r4=0 r5=0 r6=0                                                    "reset"
wnt='' wexp=''
pmode='full'
fpmult=0.5
sf=0 wf=np/sw/2
```

```
sp=rfp-rfl wp=sw
full ph
bs='n'                                          "makes sure bs='n' for 2D/3D"
dcg='cdc' cdc
sw=sw
lp=0
"set up reasonable values for typical np/fn, mainly for 3D"
if ((np>500) and (np<520)) then np=512 fn='n' endif
if ((np>1010) and (np<1040)) then np=1024 fn='n' endif
if ((np>2020) and (np<2060)) then np=2048 fn='n' endif
if ((np>4060) and (np<4120)) then np=4096 fn='n' endif
if ((np>8170) and (np<8210)) then np=8192 fn='n' endif
if ((np>16360) and (np<16400)) then np=16384 fn='n' endif
fn=2*fn
fn='y'
```

8.5 Use of the probefile

The next macro to run is BPupdate_from _probefile which obtains values for parameters from the active probefile (specified by the global variable ‘probe’). This is where the user stores values when a probefile update is performed, either manually or during the AutoCalibrate process. The BPupdate_from_probefile macro is detailed below. In many cases the retrieved parameters may not be used in the pulse sequence, but the information is useful and is displayed in a common format within the parameter panels. Thus, the 13C pw90 is always available in the “Channels” VnmrJ panel.

8.6 The BPupdate_from _probefile macro

```
"@(#)BPupdate_from_probefile 1.1 06/29/00 Copyright (c) 1999-2009 Varian, Inc. All Rights
Reserved."
                                        " Updates universal parameters values from probefile"
                                            /* Gets common acquisition parameter values */


getparam('rof1','H1'):rof1
getparam('rof2','H1'):rof2
getparam('alfa','H1'):alfa
getparam('BPd1','H1'):d1
getparam('BPgain','H1'):gain
getparam('sw','H1'):sw
getparam('at','H1'):at
fn=np if fn<np then fn=2*fn endif
```

/* **Gets common RF parameter values** */

```
getparam('tofH2O','H1'):tof
getparam('tpwr','H1'):tpwr
getparam('ref_pw90','H1'):ref_pw90
getparam('ref_pwr','H1'):ref_pwr
getparam('compH','H1'):compH
getparam('pw90','H1'):pw
getparam('pw90','H1'):pw90
getparam('pwC','C13'):pwC
getparam('pwClvl','C13'):pwClvl
getparam('compC','C13'):compC
getparam('dofCO','C13'):dof
getparam('pwN','N15'):pwN
getparam('pwNlvl','N15'):pwNlvl
getparam('compN','N15'):compN
getparam('dofN15','N15'):dof2
getparam('dpwr2NH','N15'):dpwr2
getparam('dmm2NH','N15'):dmm2
getparam('dseq2NH','N15'):dseq2
getparam('dres2NH','N15'):dres2
getparam('dmf2NH','N15'):dmf2
dseq2='wurst40N'
if Console='vnmrs' then
getparam('ddrtc','H1'):ddrtc
endif
BPcheckprobefile('empty_noise',tn):$e
if ($e=1) then
exists('empty_noise','parameter'):$e
if ($e=0) then
create('empty_noise','real')
endif
getparam('empty_noise',tn):empty_noise
endif
```

"**always start with dof=174ppm for 13C**"

/* **Gets common gradient parameter values** */

```
exists('gzcal','parameter'):$e
if $e then
 getparam('BPgzcal','N15'):gzcal
```

```
endif
exists('grecov','parameter'):$e
if $e then
getparam('NHgstab','N15'):grecov
endif
exists('gstab','parameter'):$e
if $e then
getparam('NHgstab','N15'):gstab
endif
exists('JNH','parameter'):$e
if $e then
getparam('JNH','N15'):JNH
endif
                                    /* Gets STUD Decoupling parameter values, if appropriate */
exists('STUD','parameter'):$e
if $e then
exists('dmf140','parameter'):$ex
if $ex<0.5 then
create('dmf140','real')
endif
exists('dmf80','parameter'):$ex
if $ex<0.5 then
create('dmf80','real')
endif
exists('dmf30','parameter'):$ex
if $ex<0.5 then
create('dmf30','real')
endif
exists('rf140','parameter'):$ex
if $ex<0.5 then
create('rf140','real')
endif
exists('rf80','parameter'):$ex
if $ex<0.5 then
create('rf80','real')
endif
exists('rf30','parameter'):$ex
```

```
if $ex<0.5 then
create('rf30','real')
endif
getparam('dmf140','C13'):dmf140
getparam('dmf80','C13'):dmf80
getparam('dmf30','C13'):dmf30
getparam('rf140','C13'):rf140
getparam('rf80','C13'):rf80
getparam('rf30','C13'):rf30
endif
```

Experiment setup macros and the BPrtppar macro all use the getparam macro. This macro is the fundamental tool for accessing the probefile. It requires the name of the parameter stored in the probefile, as well as the section (by nucleus) in which it is stored. The output of the macro is the value of the parameter that is placed in the parameter specified. A similar macro, BPaddparams, is used to add a parameter to the probefile. It accepts only text strings so it is common to prepare a text string using the format command prior to using addparams. If the parameter is already present as a text string, for example dmm, the formatting is not necessary.

As indicated above, most macros involved in experiment setup retrieve parameters values from information stored in a probefile. This file is located in /vnmr/probes or ~/vnmrsys/probes (the user file). Different probes directories can be stored here. While typically these are physically different probes, this is not required. The probefile names could be for different samples, for example, so that by changing probe “names” one can return to a desired set of conditions or calibrations. VnmrJ offers a probe popup utility in the hardware bar that has a menu widget showing all system or user probefiles. Just clicking the appropriate probefile name changes the value of “probe” in the user’s ~/vnmrsys/global.

The default name for the probefile for biomolecular experiments is “HCN”. Once this probefile has been created (see below) the name can be changed by using a terminal window and the “mv” command to any desired name. All macros use only the name contained in the global variable “probe”, rather than looking for a specific “HCN” file.

Once the BPupdate_from_probefile macro is run, the BPcheck macro is executed. This macro makes sure that the power levels obtained from the probefile are reconciled with the user-defined power limits set in the “Globals&Probefile” page in the Setup folder. If the power levels are too high and non-waveform modulation is used for the decoupling, the power levels are set to the user-defined power limit and the “dmf/dmf2” values are re-calculated to be consistent with the lower power.

8.7 The BPcheck macro

```
"This is a macro that will run before every BPsvp"
"and after every BPrtppar if BPinstall=1"
/* Make sure powers are within user-defined limits */
if BPpwrlimits=1 then                    "BPpwrlimits in use"
if (numrfch>3) then
if ((BPtpwrmax<40) or (BPdpwrmax<30) or (BPdpwr2max<30) or (BPdpwr3max<30)) then
```

```
BPpwrlimits=0                          "disable incorrect power limits"
banner('User-entered power limits too low. Power limits have been disabled')
write('line3','User-entered power limits too low. Power limits disabled')
endif
elseif (numrfch>2) then
if ((BPtpwrmax<40) or (BPdpwrmax<30) or (BPdpwr2max<30)) then
BPpwrlimits=0                          "disable incorrect power limits"
banner('User-entered power limits too low. Power limits have been disabled')
write('line3','User-entered power limits too low. Power limits disabled')
endif
else
if ((BPtpwrmax<40) or (BPdpwrmax<30)) then
BPpwrlimits=0                          "disable incorrect power limits"
banner('User-entered power limits too low. Power limits have been disabled')
write('line3','User-entered power limits too low. Power limits disabled')
endif
endif
if (tpwr>BPtpwrmax) then               "only for case of too much power"
ln(10):$ln10
$exp=$ln10/20*(tpwr-BPtpwrmax)
exp($exp):$factor
pw90=pw90*$factor
tpwr=BPtpwrmax
endif
if (dpwr>BPdpwrmax) then               "only for case of too much power"
if ((dmm<>'ccp')or(dmm<>'cccp')or(dmm<>'cpcp'))
then                                   "only for non-waveform modulation"
ln(10):$ln10
$exp=$ln10/20*(dpwr-BPdpwrmax)
exp($exp):$factor
dmf=dmf/$factor
dpwr=BPdpwrmax
endif
endif
if dpwr2>BPdpwr2max then          "only for case of too much power"
if ((dmm2<>'ccp')or(dmm2<>'cccp')or(dmm2<>'cpcp'))
then                                   "only for non-waveform modulation"
```

```
ln(10):$ln10
$exp=$ln10/20*(dpwr2-BPdpwr2max)
exp($exp):$factor
dmf2=dmf2/$factor
dpwr2=BPdpwr2max
endif
endif
if (numrfch>3) then
if dpwr3>BPdpwr3max then          "only for case of too much power"
ln(10):$ln10
$exp=$ln10/20*(dpwr3-BPdpwr3max)
exp($exp):$factor
dmf3=dmf3/$factor
dpwr3=BPdpwr3max
endif
endif
endif
```

The next macro makes sure that the parameter “ddrtc” is properly set (for VNMRS

systems only). This time is used by the acquisition hardware to properly time the sampling of the FID so that there is no frequency-dependent phase correction (lp) necessary, ensuring flat baselines.

The value is pulse-sequence specific. That is, if the sequence ends with an observe pulse the ddrtc value must account for any post-pulse delay (e.g. rof2 and alfa). Cold probes require a significant delay (e.g. 20-300 usec) to allow the high-Q probe to ring down before sampling. Room temperature probes can have shorter delays. A typical value of ddrtc is just the sum of all post-pulse delays. If the sequence end with a spin-echo, the sequence may be coded without a rof2 delay such that at the end of the echo the magnetization is perfectly refocused. In this case, the ddrtc value would be simply alfa.

Since the ddrtc value is sequence-specific, the BPsetddrtc macro checks seqfil (contains the name of the sequence) and sets ddrtc accordingly. Use of any sequence other than those checked by BPsetddrtc means that the user should set ddrtc manually according to the same considerations described above.

8.7.1 The BPsetddrtc macro

```
"BPsetddrtc- macro to set ddrtc for sequences"
“that do not use rof2 at end of sequence"
" ddrtc is typically near rof2+alfa if rof2 is used"
"therefore, ddrtc is set to alfa"
$SE="
```

```
exists('SE','parameter'):$e
if $e=1 then $SE=SE else $SE='n' endif

if (Console='vnmrs') then
substr(seqfil,1,2):$first2
substr(seqfil,1,3):$first3
substr(seqfil,1,4):$first4
substr(seqfil,1,5):$first5
substr(seqfil,1,8):$first8
substr(seqfil,1,14):$first14
if ($#>0) then
$e=0
if (((seqfil='gChsqc') or (seqfil='gChsqcA') and $SE='n'))
then $e=1 endif
if $first2='gc' then  $e=1 endif
if $first3='JIT' then  $e=1 endif
if $first3='ghn' then  $e=1 endif
if $first4='gChm' then  $e=1 endif
  if $first2='gN' then  $e=1 endif
if $first4='best' then  $e=1 endif
if $first5='ghcch' then  $e=1 endif
if seqfil='gnoesyChsqcSE' then  $e=1 endif
if seqfil='gnoesyChsqc_wg' then  $e=1 endif
if seqfil='rna_gCtrosy' then  $e=1 endif
if seqfil='rna_HNNcosy' then  $e=1 endif
if seqfil='rna_HCN' then  $e=1 endif
if seqfil='rna_HCP' then  $e=1 endif
if seqfil='crinept' then  $e=1 endif
if $e=1 then
return(0)
else
return(1)
endif
else
if $first2='gc' then  ddrtc=alfa endif
if (($first2='PR') and (seqfil<>'PR42_gChsqcnoesyChsqc')) then  ddrtc=alfa endif
if $first3='ghn' then ddrtc=alfa endif
```

```
if $first3='JIT' then  ddrtc=alfa endif
if $first4='gChm' then  ddrtc=alfa endif
if $first4='gCPM' then  ddrtc=alfa endif
if $first2='gN' then ddrtc=alfa endif
if $first4='best' then ddrtc=alfa  endif
if $first5='ghcch' then  ddrtc=alfa endif
if ((seqfil='gChsqc') or (seqfil='gChsqcA') or (seqfil='gChsqcP')) then
if $SE='y' then
ddrtc=alfa
else
getparam('ddrtc','H1'):ddrtc
endif
endif
if seqfil='wgcosy' then ddrtc=alfa endif
if seqfil='wgnoesy' then ddrtc=alfa endif
if seqfil='gnoesyChsqcSE' then ddrtc=alfa endif
if seqfil='groesyChsqcSM' then ddrtc=alfa endif
if seqfil='gnoesyChsqcSM' then ddrtc=alfa endif
if seqfil='gtocsyChsqcSM' then ddrtc=alfa endif
if seqfil='groesyNhsqcSM' then ddrtc=alfa endif
if seqfil='gnoesyNhsqcSM' then ddrtc=alfa endif
if seqfil='gtocsyNhsqcSM' then ddrtc=alfa endif
if seqfil='gnoesyNhsqc' then ddrtc=alfa endif
if seqfil='gtocsyChsqc_wg' then ddrtc=alfa endif
if seqfil='rna_gCtrosy' then ddrtc=alfa endif
if seqfil='rna_HNNcosy' then ddrtc=alfa endif
if seqfil='rna_HCN' then ddrtc=alfa endif
if seqfil='rna_HCP' then ddrtc=alfa endif
if seqfil='rna_11echo' then ddrtc=alfa endif
if seqfil='crinept' then ddrtc=alfa endif
if seqfil='sofastNhmqc' then ddrtc=alfa endif
if seqfil='C13observe' then ddrtc=alfa endif
if seqfil='CNfilnoesy' then ddrtc=alfa endif
if seqfil='gCNfilnoesyNfhsqcA' then ddrtc=alfa endif
if seqfil='gCNfilnoesyChsqcSE' then ddrtc=alfa endif
if seqfil='STS_SOFAST' then ddrtc=alfa endif
if seqfil='sSCT_CCR' then ddrtc=alfa endif
```

endif

else

return(0)

endif

Many BioPack pulse sequences use selective pulses on H2O. These pulses are generated and adjusted by the BPflipbacks macro (Table 8). The soft pulse is often named ‘pwHs’ and its power level is typically controlled by the coarse power tpwrs and the fine power tpwrsf*.

8.8 The BPflipbacks macro

"**BPflipbacks -a macro to handle H2O soft pulses**"

exists('pwHs','parameter'):$e

if $e=1 then

getparam('pwHs','H1'):pwHs

“use a default width based on 1.7msec at 500 MHz”

“remove the next line to use the probefile value”

pwHs=1700*(500/sfrq)

exists('grecov','parameter'):$e "rna sequence"

if $e then

$shape='rna_H2Osinc'

else

$shape='H2Osinc'

endif

exists(systemdir+'/shapelib/'+$shape+'.RF','file'):$es

if not($es) then

exists(userdir+'/shapelib/'+$shape+'.RF','file'):$es

endif

if $es then

pwsadj($shape,'pwHs')

endif

exists('tpwrsf','parameter'):$e

if $e=1 then getparam('tpwrsf','H1'):tpwrsf endif

endif

“create fine power parameters if not present”

exists('tpwrsf_t','parameter'):$e

if $e=0 then create('tpwrsf_t','real') tpwrsf_t=4095 endif

exists('tpwrsf_i','parameter'):$e

if $e=0 then create('tpwrsf_i','real') tpwrsf_i=4095 endif

exists('tpwrsf_u','parameter'):$e

```
if $e=0 then create('tpwrsf_u','real') tpwrsf_u=4095 endif
exists('tpwrsf_d','parameter'):$e
if $e=0 then create('tpwrsf_d','real') tpwrsf_d=4095 endif
exists('tpwrsf_n','parameter'):$e
if $e=0 then create('tpwrsf_n','real') tpwrsf_n=4095 endif
exists('tpwrsf_a','parameter'):$e
if $e=0 then create('tpwrsf_a','real') tpwrsf_a=4095 endif
$probe=probe
exists(userdir+'/probes/'+$probe,'file','r-x'):$e
if not($e) then
$sysadd = 1
$probedir = systemdir + '/probes'
else
$sysadd = 0
$probedir = userdir + '/probes'
endif
$result='' $numi=0 $numt=0 $numu=0 $numd=0 $numn=0
$probefile = $probedir + '/' + $probe + '/' + $probe
lookup('file',$probefile)
lookup('seek','H1tpwrsf_i','read'):$result,$numi
lookup('file',$probefile)
lookup('seek','H1tpwrsf_t','read'):$result,$numt
lookup('file',$probefile)
lookup('seek','H1tpwrsf_u','read'):$result,$numu
lookup('file',$probefile)
lookup('seek','H1tpwrsf_d','read'):$result,$numd
lookup('file',$probefile)
lookup('seek','H1tpwrsf_a','read'):$result,$numa
lookup('file',$probefile)
lookup('seek','H1tpwrsf_n','read'):$result,$numn
if $numi=0 then
$tpwrsf_i='' format(tpwrsf_i,4,0):$tpwrsf_i
addparams('tpwrsf_i',$tpwrsf_i,'H1')
else
getparam('tpwrsf_i','H1'):tpwrsf_i
endif
if $numt=0 then
```

```
$tpwrsf_t='' format(tpwrsf_t,4,0):$tpwrsf_t
addparams('tpwrsf_t',$tpwrsf_t,'H1')
else
getparam('tpwrsf_t','H1'):tpwrsf_t
endif
if $numu=0 then
$tpwrsf_u='' format(tpwrsf_u,4,0):$tpwrsf_u
addparams('tpwrsf_u',$tpwrsf_u,'H1')
else
getparam('tpwrsf_u','H1'):tpwrsf_u
endif
if $numd=0 then
$tpwrsf_d='' format(tpwrsf_d,4,0):$tpwrsf_d
addparams('tpwrsf_d',$tpwrsf_d,'H1')
else
getparam('tpwrsf_d','H1'):tpwrsf_d
endif
f $numa=0 then
$tpwrsf_a='' format(tpwrsf_a,4,0):$tpwrsf_a
addparams('tpwrsf_a',$tpwrsf_a,'H1')
else
getparam('tpwrsf_a','H1'):tpwrsf_a
endif
if $numn=0 then
$tpwrsf_n='' format(tpwrsf_n,4,0):$tpwrsf_n
addparams('tpwrsf_n',$tpwrsf_n,'H1')
else
getparam('tpwrsf_n','H1'):tpwrsf_n
endif
exists('pwHs2','parameter'):$e
if $e=1 then
getparam('pwHs2','H1'):pwHs2
exists('finepwrf','parameter'):$e
if $e=1 then getparam('finepwrf','H1'):finepwrf endif
endif
```

The BPfixrefs macro (see 8.9) takes care of referencing of 3D parameter sets.

8.9 The BPfixrefs macro

```
"Create 3D referencing parameters as needed. Set to defaults for HCN"
"Observe dimension"
exists('reffrq','parameter'):$e
if not($e) then
create('reffrq','real')
endif
setgroup('reffrq','display')
reffrq = sfrq - (sw/2 - rfl + rfp)/1e6
exists('refpos','parameter'):$e
if not($e) then
  create('refpos','real')
endif
setgroup('refpos','display')
refpos = 0
"F1 Dimension"
exists('refsource1','parameter'):$e
if not($e) then
create('refsource1','string')
endif
setgroup('refsource1','display')
refsource1 = 'dfrq'
exists('reffrq1','parameter'):$e
if not($e) then
create('reffrq1','real')
endif
setgroup('reffrq1','display')
exists('sw1','parameter'):$e
if not($e) then
reffrq1 = dfrq
else
reffrq1 = dfrq - (sw1/2 - rfl1 + rfp1)/1e6
endif
exists('refpos1','parameter'):$e
if not($e) then
create('refpos1','real')
```

```
endif
setgroup('refpos1','display')
refpos1 = 0
refpos1 = 'n'
"F2 Dimension"
exists('refsource2','parameter'):$e
if not($e) then
create('refsource2','string')
endif
setgroup('refsource2','display')
refsource2 = 'dfrq2'
exists('reffrq2','parameter'):$e
if not($e) then
create('reffrq2','real')
endif
setgroup('reffrq2','display')
exists('sw2','parameter'):$e
if not($e) then
reffrq2 = dfrq2
else
reffrq2 = dfrq2 - (sw2/2 - rfl2 + rfp2)/1e6
endif
exists('refpos2','parameter'):$e
if not($e) then
  create('refpos2','real')
endif
setgroup('refpos2','display')
refpos2 = 0
refpos2 = 'n'
```

The last macro to run as part of BPrtppar is BPfixup2 (see 8.10).

8.10 The BPfixup2 macro

```
"BPfixup2  - a macros for a final check of parameters"
if (Console='vnmrs') then
length(dmm):$length
substr(dmm,$length,1):$mode1                    "last character of dmm"
length(dmm2):$length
```

```
substr(dmm2,$length,1):$mode2                    "last character of dmm2"
length(dmm3):$length
substr(dmm3,$length,1):$mode3                    "last character of dmm3"
if ($mode1<>'p') then dres=90.0 endif            "force dres value of 90.0"
if ($mode2<>'p') then dres2=90.0 endif
if ($mode3<>'p') then dres3=90.0 endif
endif
exists('mag_flg','parameter'):$ "magic-angle gradients (triax probe)"
if $e=1 then
getparam('mag_flg','N15'):mag_flg
if (mag_flg='y') then
BPcheckfortriax                                   "macro to check xyz calibrations"
endif
endif
getparam('BPtemp','H1'):temp
"installs temp value used in autocalibrate or autoupdate"
""
sb=-at sbs=sb gf='n' awc='n' lb='n'              "set reasonable weighting"
parfidss                                         "set solvent suppression"
getparam('ssfilter','H1'):ssfilter
getparam('ssntaps','H1'):ssntaps
getparam('ssorder','H1'):ssorder
getparam('sslsfrq','H1'):sslsfrq
if (tn='H1') then
dn='C13'                                        "prevents 1H on channel 2 with obs 1H on channel 1"
endif
```

The BPcheckfortriax macro makes sure a gradtable is present in /vnmr/imaging/gradtables. The gradtable has the calibrations for the x, y and z gradients so that the pulse sequence statement “magradpulse” will work properly (used for magic-angle gradients). The probefile has an entry for mag_flg. A value of ‘y’ indicates that the user wants to use magic-angle gradients for those sequences in which they are optional (must have a triax probe and amplifier and pfgon=’yyy’). The value stored in the probefile is set when doing a full probefile update using the button on the “Globals&Probefile” page in the Setup folder when ghn_co.par is loaded into the current experiment.

Chapter 9 Modifying, Updating, and Usage of the Probefile

The probefile must be updated with calibrations appropriate for the current instrument, after the probefile is created. This is done first by having a current parameter set that has all the common parameters stored in the probefile (typically, this is true for the ghn_co.par parameter set). Therefore, to update the probefile use the drop-down VnmrJ menu to select Triple-Resonance/NH-Detected/HNCO, or use the macro “ghn_co” on the command line. In either case the ghn_co macro is executed.

If BioPack has not been “activated” (the global variable BPinstall=0), menu options are provided to “activate” the files (these set BPinstall=1). These options run the macro BPbiopack1a or BPbiopack1b which in turn runs Bpaddprobe, ending with setting up a ghn_co parameter set. These menu options allow a new probefile (BPbiopack1a) or, (a) a copy of the vnmr1 probefile or, (b) a copy of the vnmr probefile,or (c) a copy of the current probefile to be created (BPbiopack1b).

Once the calibrated or approximate calibrations are entered, the user must update the probefile using the button present in the “Globals&Probefile” panel in the “Setup” folder. This runs the macro BPbiopack2.

9.1 The ghn_co macro

BPrtppar('ghn_co')	
getparam('NHgrad1_time','N15'):gt1	**“Gets parameters for ghn_co”**
getparam('NHgrad1_lvl','N15'):gzlvl1	**“coherence transfer gradients”**
getparam('NHgrad2_lvl','N15'):gzlvl2	
getparam('NHgstab','N15'):gstab	“**field recovery delay after gradient**”
getparam('swN','N15'):sw2	**“typical 15N spectral window”**
getparam('waltzB1','H1'):waltzB1	**“proton decoupling field strength”**
dm2='nny'	“**sets up for 15N decoupling in acquisition**”
BPsetampmode	**“sets ampmode parameter value to match hardware**”
if (numrfch>3) then	**“gets 2H decoupling parameters”**
getparam('dof3D','H2'):dof3	
getparam('dmm3D','H2'):dmm3	
getparam('dmf3D','H2'):dmf3	
getparam('dpwr3D','H2'):dpwr3	
getparam('dres3D','H2'):dres3	
getparam('dseq3D','H2'):dseq3	
else	“**don’t need ampmode if <4 channels**”
exists('ampmode','parameter'):$e	

if ($e=1) then destroy('ampmode') endif

endif

getparam('gt5','N15'):gt5 “**get values for crusher gradients**”

getparam('gt0','N15'):gt0

getparam('gzlvl5','N15'):gzlvl5

getparam('gzlvl0','N15'):gzlvl0

/*__Recalculates pulse widths using same equations as used for__ */

/*__making shapes so no timing errors are generated at “go”__ */

pwC3 = 1.0e6*sqrt(3.0)/(2.0*118.0*dfrq)

$pw = pwC3*5.0 $pw = trunc($pw + 0.5)

pwC3 = $pw/5.0 pwC3a = pwC3

pwC4 = 1.0e6*sqrt(3.0)/(2.0*118.0*dfrq)

$pw = pwC4*5.0 $pw = trunc($pw + 0.5)

pwC4 = $pw/5.0

pwC5 = 88.8*600.0/sfrq

$pw = pwC5*5.0 $pw = 2.0*trunc($pw/2.0) + 1.0

pwC5 = ($pw + 1.0)/5.0

pwC6 = 88.8*600.0/sfrq

$pw = pwC6*5.0 $pw = 2.0*trunc($pw/2.0) + 1.0

pwC6 = ($pw + 1.0)/5.0

pwC7 = 80.5*600.0/sfrq

$pw = pwC7*5.0 $pw = 2.0*trunc($pw/2.0) + 1.0

pwC7 = ($pw + 1.0)/5.0

pwC7a = pwC7

pwC8 = 80.5*600.0/sfrq

$pw = pwC8*5.0 $pw = 2.0*trunc($pw/2.0) + 1.0

pwC8 = ($pw + 1.0)/5.0

pwC8a = pwC8

pwC9 = 80.5*600.0/sfrq

$pw = pwC9*5.0 $pw = 2.0*trunc($pw/2.0) + 1.0

pwC9 = ($pw + 1.0)/5.0

pwC9a = pwC9

BPmake180Ca_CO

BPmake90CO_CO

BPmake180CO_CO

spin='n' ni=0 ni2=0 phase=1 phase2=1

9.2 The BPbiopack1a macro

This macro runs relevant sections of the ghn_co macro. It is used as part of the “Activate” process, driven by menu. No probefile likely exists so no getparam statements are used. It is run only once, to setup ghn_co and create the probefile.

```
"BPbiopack1a(<y>)"
"BPbiopack1a('y') is run by VNMRJ to activate BioPack from MainMenu"
"BPbiopack1a with no options is run from VNMR 6.1C menu"
$tail='tail'
exists('/usr/xpg4/bin/tail','file','x'):$e
if $e then
$tail='/usr/xpg4/bin/tail'
endif
if (BPinstall=0) then                                   "only true at installation"
$gv='' $revfile=systemdir+'/BioPack.dir/BP_rev'
exists($revfile,'file'):$v1
if ($v1) then
shell('grep "version 20" <',$revfile,'|',$tail,'-n 1 | awk \'{print $NF}\'; cat'):$gv
$lv='' $revfile=userdir+'/BioPack.dir/BP_rev'
exists($revfile,'file'):$v2
if ($v2) then
shell('grep "version 20" <',$revfile,'|',$tail,'-n 1 | awk \'{print $NF}\'; cat'):$lv
endif
if ($lv<>'')and($gv<>'') then
$lastv=''
shell('(echo',$lv,'; echo',$gv,')| sort|',$tail,'-n 1; cat'):$lastv
if $lastv=$gv then
exists(systemdir+'/Biopack.dir/BP_files','file'):$e
if $e then
BPrmlocalfiles''remove any old files and store in backup file''
endif
endif
elseif $gv<>'' then
BPrmlocalfiles
endif
endif
endif
```

```
BPrtppar('ghn_co')                              "setup a ghn_co parameter set"
clear(2)
fn=np np=fn fn=2*np
pwC3 = 1.0e6*sqrt(3.0)/(2.0*118.0*dfrq)
$pw = pwC3*5.0      $pw = trunc($pw + 0.5)
pwC3 = $pw/5.0
pwC3a = pwC3
pwC4 = 1.0e6*sqrt(3.0)/(2.0*118.0*dfrq)
$pw = pwC4*5.0      $pw = trunc($pw + 0.5)
pwC4 = $pw/5.0
pwC5 = 88.8*600.0/sfrq
$pw = pwC5*5.0   $pw = 2.0*trunc($pw/2.0) + 1.0
pwC5 = ($pw + 1.0)/5.0
pwC6 = 88.8*600.0/sfrq
$pw = pwC6*5.0   $pw = 2.0*trunc($pw/2.0) + 1.0
pwC6 = ($pw + 1.0)/5.0
pwC7 = 80.5*600.0/sfrq
$pw = pwC7*5.0   $pw = 2.0*trunc($pw/2.0) + 1.0
pwC7 = ($pw + 1.0)/5.0
pwC7a = pwC7
pwC8 = 80.5*600.0/sfrq
$pw = pwC8*5.0   $pw = 2.0*trunc($pw/2.0) + 1.0
pwC8 = ($pw + 1.0)/5.0
pwC8a = pwC8
pwC9 = 80.5*600.0/sfrq
$pw = pwC9*5.0   $pw = 2.0*trunc($pw/2.0) + 1.0
pwC9 = ($pw + 1.0)/5.0
pwC9a = pwC9
spin='n' ni=0  ni2=0 phase=1 phase2=1
exists('waltzB1','parameter'):$e
if $e=0 then create('waltzB1','real') waltzB1=5p endif
exists('dmf30','parameter'):$e
if $e=0 then create('dmf30','real') endif
exists('rf30','parameter'):$e
if $e=0 then create('rf30','real') endif
exists('dmf80','parameter'):$e
if $e=0 then create('dmf80','real') endif
```

```
exists('rf80','parameter'):$e
if $e=0 then create('rf80','real') endif
exists('dmf140','parameter'):$e
if $e=0 then create('dmf140','real') endif
exists('rf140','parameter'):$e
if $e=0 then create('rf140','real') endif
tn=tn dn=dn dn2=dn2            "make sure proper frequencies are set"
setfrq
if (numrfch>3) then dn3=dn3 endif
exists('reffrq','parameter'):$e
if ($e=0) then create('reffrq','real') endif
reffrq=sfrq-(sw/2-rfl+rfp)/1e6
if (BPinstall=0) then                                    "only true at installation"
/* This section creates the HCN probefile with parameters          */
/* (The BPbiopack1b macro makes a copy of the system HCN probefile */
/* instead of making a new probefile) */

write('line3','New "HCN" Probefile being created')
BPaddprobe('HCN')
probe='HCN'
write('line3','BioPack parameters added to new Probefile')
"Set typical values for spinlock fieldstrengths"
"values of dof,dof2 and tof are ok for 800,750,600 and 500"
BP1Hspinlock=12p BPtpwrspinlock=12p
if (h1freq=900) then
sw=14500 np=2048
BPspinlock=8000 BPdpwrspinlock=8000
endif
if (h1freq=800) then
sw=13000 np=2048
BPspinlock=8000 BPdpwrspinlock=8000
endif
if (h1freq=750) then
sw=12000 np=2048
BPspinlock=8000 BPdpwrspinlock=8000
endif
if (h1freq=700) then
```

```
sw=11500 np=2048
BPspinlock=7500 BPdpwrspinlock=7500
endif
if (h1freq=600) then
BPspinlock=7000 BPdpwrspinlock=7000
endif
if (h1freq=500) then
sw=8000 np=1024
BPspinlock=6000 BPdpwrspinlock=6000
endif
if (h1freq=400) then
sw=7000 np=1024
BPspinlock=5000 BPdpwrspinlock=5000
endif
if (h1freq=300) then
sw=5000 np=512
BPspinlock=4000 BPdpwrspinlock=4000
endif
endif
"use setoffset statement to set offsets "
setoffset('H1',4.67):tof
setoffset('C13',174):dof
setoffset('N15',120):dof2
setoffset('H2',4):dof3
setref
pw=pw90                    "sets pw if pw was 0 at installation"
BPbiopack2          "makes sure shapes are created at least once"
BPinstall=1                                    "indicates installation is done"
printon man('BioPacklist') printoff
man('BioPack.update')
if ($#=0) then menu('BioPack1a')
 banner('A new user probefile has been created\\ \\Check printed list for important
 parameters\\Then update parameter values in this parameter set.\\ \\Click "Update Probefile with
 These Parameters"\\menu button to finish probefile update\\ \\Exit and Re-start VNMRJ after
 updating\\ \\THIS IS REQUIRED BEFORE ANY EXPERIMENT OR CALIBRATION IS DONE')
else
 banner('A new user probefile HCN has been created\\ \\Check printed list for important
```

```
parameters.\\Then update parameter values in this parameter set.\\   \\Click "Globals&Probefile"
tab in Setup Pages.\\ \\Click "Update Probefile From...." button to finish probefile update\\   \\Exit
and restart VNMRJ after Updating\\ \\THIS IS REQUIRED BEFORE ANY EXPERIMENT OR
CALIBRATION IS DONE ')
endif
```

9.3 The BPbiopack1b macro

```
"BPbiopack1b(option)- a version of BPbiopack1a to copy probefile"
"instead of making new probefile"
"option='0':  copies /vnmr/probes/HCN  probefile  "
"option='1':  copies .../vnmr1/vnmrsys/probes/HCN probefile  "
"option='2':  copies /vnmr/probes/HCN probefile  VJ only  "
"option='3':  copies .../vnmr1/vnmrsys/probes/HCN  probefile    VJ only"
"option='current':  copies ~/vnmrsys/probes current probefile    VJ only"
$tail='tail'
exists('/usr/xpg4/bin/tail','file','x'):$e
if $e then
 $tail='/usr/xpg4/bin/tail'
endif
if (BPinstall=0) then                    "only true at installation"
 $gv='' $revfile=systemdir+'/BioPack.dir/BP_rev'
 exists($revfile,'file'):$v1
 if ($v1) then
  shell('grep "version 20" <',$revfile,'|',$tail,'-n 1 | awk \'{print $NF}\'; cat'):$gv
  $lv='' $revfile=userdir+'/BioPack.dir/BP_rev'
  exists($revfile,'file'):$v2
  if ($v2) then
   shell('grep "version 20" <',$revfile,'|',$tail,'-n 1 | awk \'{print $NF}\'; cat'):$lv
  endif
  if ($lv<>'')and($gv<>'') then
   $lastv=''
   shell('(echo',$lv,'; echo',$gv,')| sort|',$tail,'-n 1; cat'):$lastv
   if $lastv=$gv then
    exists(systemdir+'/Biopack.dir/BP_files','file'):$e
    if $e then
     BPrmlocalfiles
    endif
```

```
    endif
   elseif $gv<>'' then
    BPrmlocalfiles
   endif
  endif

  $error=0
  BPcopyprobe('HCN',$1):$error
                    "probefile is copied into user probes file"
  if $error=1 then return endif                    "no source probefile present"
endif
probe='HCN'
BPinstall=1                                        "indicates installation is done"
BPrtppar('ghn_co')
clear(2)
fn=2*np
pwC3 = 1.0e6*sqrt(3.0)/(2.0*118.0*dfrq)
$pw = pwC3*5.0      $pw = trunc($pw + 0.5)    pwC3 = $pw/5.0
pwC3a = pwC3
pwC4 = 1.0e6*sqrt(3.0)/(2.0*118.0*dfrq)
$pw = pwC4*5.0      $pw = trunc($pw + 0.5)    pwC4 = $pw/5.0
pwC5 = 88.8*600.0/sfrq
$pw = pwC5*5.0   $pw = 2.0*trunc($pw/2.0) + 1.0
pwC5 = ($pw + 1.0)/5.0
pwC6 = 88.8*600.0/sfrq
$pw = pwC6*5.0   $pw = 2.0*trunc($pw/2.0) + 1.0
pwC6 = ($pw + 1.0)/5.0
pwC7 = 80.5*600.0/sfrq
$pw = pwC7*5.0   $pw = 2.0*trunc($pw/2.0) + 1.0
pwC7 = ($pw + 1.0)/5.0
pwC7a = pwC7
pwC8 = 80.5*600.0/sfrq
$pw = pwC8*5.0   $pw = 2.0*trunc($pw/2.0) + 1.0
pwC8 = ($pw + 1.0)/5.0
pwC8a = pwC8
pwC9 = 80.5*600.0/sfrq
$pw = pwC9*5.0   $pw = 2.0*trunc($pw/2.0) + 1.0
```

```
pwC9 = ($pw + 1.0)/5.0
pwC9a = pwC9
spin='n' ni=0  ni2=0 phase=1 phase2=1
exists('waltzB1','parameter'):$e
if $e=0 then create('waltzB1','real') waltzB1=5p endif
exists('dmf30','parameter'):$e
if $e=0 then create('dmf30','real') endif
exists('rf30','parameter'):$e
if $e=0 then create('rf30','real') endif
exists('dmf80','parameter'):$e
if $e=0 then create('dmf80','real') endif
exists('rf80','parameter'):$e
if $e=0 then create('rf80','real') endif
exists('dmf140','parameter'):$e
if $e=0 then create('dmf140','real') endif
exists('rf140','parameter'):$e
if $e=0 then create('rf140','real') endif
tn=tn dn=dn dn2=dn2                    "make sure proper frequencies are set"
setfrq
if (numrfch>3) then dn3=dn3 endif
exists('reffrq','parameter'):$e
if ($e=0) then create('reffrq','real') endif
reffrq=sfrq-(sw/2-rfl+rfp)/1e6
setref
printon man('BioPacklist') printoff
man('BioPack.update')
if ($1='0') then menu('BioPack1a')          "For VNMR only"
banner('/vnmr/probes/HCN probefile copied to user probes file\\  \\Check printed list for important parameters.\\   \\Then update parameter values in this parameter set.\\    \\Click "Update Probefile with These Parameters"\\   \\menu button to finish probefile update\\  \\THIS IS REQUIRED BEFORE ANY EXPERIMENT OR CALIBRATION IS DONE')
elseif ($1='1') then                        "For VNMR only"
banner('NMR Admin. vnmrsys/probes/HCN probefile copied to user probes file\\  \\Check printed list for important parameters.\\   \\Then update parameter values in this parameter set.\\    \\Click "Update Probefile with These Parameters"\\   \\menu button to finish probefile update\\  \\THIS IS REQUIRED BEFORE ANY EXPERIMENT OR CALIBRATION IS DONE')
elseif ($1='2') then                        "For VNMRJ only"
```

banner('/vnmr/probes/HCN probefile copied to user probes file\\ \\Check printed list for important parameters.\\ \\Then update parameter values in this parameter set.\\ \\Click "Globals&Probefile" tab in Setup Pages.\\ \\Click "Update Probefile ...." button to finish probefile update\\ \\Exit and restart VNMRJ after Updating\\ \\THIS IS REQUIRED BEFORE ANY EXPERIMENT OR CALIBRATION IS DONE ')

elseif ($1='current') then **"for VNMRJ only"**

banner('current user probefile is used\\ \\Check printed list for important parameters.\\ \\Then update parameter values in this parameter set.\\ \\Click "Globals&Probefile" tab in Setup Pages.\\ \\Click "Update Probefile ...." button to finish probefile update\\ \\Exit and restart VNMRJ after Updating\\ \\THIS IS REQUIRED BEFORE ANY EXPERIMENT OR CALIBRATION IS DONE ')

elseif ($1='3') then **"For VNMRJ only"**

banner('NMR Admin. /vnmrsys/probes/HCN probefile copied to user probes file\\ \\Check printed list for important parameters.\\ \\Then update parameter values in this parameter set.\\ \\Click "Globals&Probefile" tab in Setup Pages.\\ \\Click "Update Probefile ...." button to finish probefile update\\ \\Exit and restart VNMRJ after Updating\\ \\THIS IS REQUIRED BEFORE ANY EXPERIMENT OR CALIBRATION IS DONE ')

endif

9.4 The BPaddprobe macro

"**BPaddprobe(probename)- macro to add a Probefile**"

/***This section figures out the active probe directory***/

$probedir = userdir + '/probes'

exists($probedir,'file'):$e

if not($e) then

mkdir($probedir)

endif

$probe=$1

$date=''

BPgetdate:$date

$probename=$probedir+'/'+$probe

exists($probename,'file'):$e

/* **This section backs up existing probefile** */

if $e then

  mv($probename,$probename+'.bkup_'+$date)

  write('line3','existing user Probefile backed up as %s',$probe+'.bkup_'+$date)

endif

/* **This section makes the new probefile** */

mkdir($probedir+'/'+$probe)

$probefile = $probedir + '/' + $probe + '/' + $probe

```
write('reset',$probefile)
write('file',$probefile,'NAME:          %s',$probe)

“finds location of a template probefile and copies it”
$path=''
BPfindfile('probe.tmplt','manual'):$path
if $path<>'' then
shell('cat '+$path+' >> '+$probefile):$dum
else
shell('cat '+'/vnmr/probes/probe.tmplt >> '+$probefile):$dum
endif
“makes sure a probe parameter exists”
exists('probe','parameter','global'):$e
if $e<0.5 then
create('probe','string','global')
setprotect('probe','on',8,'global')
endif
probe = $probe_probe
$file=$probedir + '/' + $probe + '/safety_levels'
write('reset',$file)
write('file',$file,'# Probe protection parameters')
write('file',$file,'# High band power level for 2 Watt')
write('file',$file,'# Low  band power level for 2 Watt')
write('file',$file,'# High band coil energy limit')
write('file',$file,'# Low  band coil energy limit')
exists(systemdir+'acqqueue/acq.conf','file'):$e
if ($e > 0.5) then
shell('showconsole 1'):$amt
else
$amt=0
endif
if ($amt=12) then
write('file',$file,'# For 100W high band/300W low band amplifier')
write('file',$file,'46.0 42.0 15.0 15.0')
else
write('file',$file,'# For 50W high band/300W low band amplifier')
write('file',$file,'49.0 42.0 15.0 15.0')
```

```
endif
write('line3','user level Probefile %s is being updated. Please wait for end message',$probe)
/* This section writes out parameter values to probefile */
pw=trunc(10*pw + 0.5)/10
pw90=trunc(10*pw90 + 0.5)/10
pwC=trunc(10*pwC + 0.5)/10
pwN=trunc(10*pwN + 0.5)/10
ref_pw90=trunc(10*pw*compH + 0.5)/10 ref_pwr=tpwr
$gf='' format(gf,4,4):$gf
addparams('gf',$gf,'H1')
$gfs='' format(gfs,4,4):$gfs
addparams('gfs',$gfs,'H1')
$sb='' format(sb,4,4):$sb
addparams('sb',$sb,'H1')
$sbs='' format(sbs,4,4):$sbs
addparams('sbs',$sbs,'H1')
$lb='' format(lb,4,4):$lb
addparams('lb',$lb,'H1')
$ssfilter='' format(ssfilter,4,0):$ssfilter
addparams('ssfilter',$ssfilter,'H1')
$ssntaps='' format(ssntaps,4,0):$ssntaps
addparams('ssntaps',$ssntaps,'H1')
$sslsfrq='' format(sslsfrq,4,4):$sslsfrq
addparams('sslsfrq',$sslsfrq,'H1')
$ssorder='' format(ssorder,2,0):$ssorder
addparams('ssorder',$ssorder,'H1')
exists('ddrtc','parameter'):$e
if $e=1 then
$ddrtc='' format(ddrtc,4,2):$ddrtc
addparams('ddrtc',$ddrtc,'H1')
endif
$rof1='' format(rof1,4,2):$rof1
addparams('rof1',$rof1,'H1')
$rof2='' format(rof2,4,2):$rof2
addparams('rof2',$rof2,'H1')
$alfa='' format(alfa,4,2):$alfa
addparams('alfa',$alfa,'H1')
```

```
$sw='' format(sw,6,1):$sw
addparams('sw',$sw,'H1')
$at='' format(at,5,3):$at
addparams('at',$at,'H1')
$ref_pw90='' format(ref_pw90,2,1):$ref_pw90
addparams('ref_pw90',$ref_pw90,'H1')
$ref_pwr='' format(ref_pwr,2,0):$ref_pwr
addparams('ref_pwr',$ref_pwr,'H1')
$pw90='' format(pw90,2,1):$pw90
addparams('pw90',$pw90,'H1')
addparams('wetpw','6000','H1')
$BPd1='' format(d1,4,4):$BPd1
addparams('BPd1',$BPd1,'H1')
$tpwr='' format(tpwr,2,0):$tpwr
addparams('tpwr',$tpwr,'H1')
addparams('wetpwr','12.0','H1')
addparams('wetshape','gauss','H1')
$tpwrsf='' format(tpwrsf,4,0):$tpwrsf
addparams('tpwrsf',$tpwrsf,'H1')
$phincr_i='' format(0.0,2,2):$phincr_i
addparams('phincr_i',$phincr_i,'H1')
$phincr_t='' format(0.0,2,2):$phincr_t
addparams('phincr_t',$phincr_t,'H1')
$phincr_u='' format(0.0,2,2):$phincr_u
addparams('phincr_u',$phincr_u,'H1')
$phincr_d='' format(0.0,2,2):$phincr_d
addparams('phincr_d',$phincr_d,'H1')
$phincr_n='' format(0.0,2,2):$phincr_n
 addparams('phincr_n',$phincr_n,'H1')
 $tpwrsf_i='' format(tpwrsf,4,0):$tpwrsf_i
 addparams('tpwrsf_i',$tpwrsf_i,'H1')
 $tpwrsf_t='' format(tpwrsf,4,0):$tpwrsf_t
 addparams('tpwrsf_t',$tpwrsf_t,'H1')
 $tpwrsf_u='' format(tpwrsf,4,0):$tpwrsf_u
 addparams('tpwrsf_u',$tpwrsf_u,'H1')
 $tpwrsf_d='' format(tpwrsf,4,0):$tpwrsf_d
 addparams('tpwrsf_d',$tpwrsf_d,'H1')
```

```
$tpwrsf_n='' format(tpwrsf,4,0):$tpwrsf_n
addparams('tpwrsf_n',$tpwrsf_n,'H1')
addparams('finepwrf',$tpwrsf,'H1')
$pwHs='' format(pwHs,2,1):$pwHs
addparams('pwHs',$pwHs,'H1')
addparams('pwHs2',$pwHs,'H1')
$waltzB1='' format(waltzB1,6,0):$waltzB1
addparams('waltzB1',$waltzB1,'H1')
$tofH2O='' format(tof,4,1):$tofH2O
addparams('tofH2O',$tofH2O,'H1')
$compH='' format(compH,4,3):$compH
addparams('compH',$compH,'H1')
$BPgain='' format(gain,2,0):$BPgain
addparams('BPgain',$BPgain,'H1')
$rna_gain='' format(gain,2,0):$rna_gain
addparams('rna_gain',$rna_gain,'H1')
$BPtemp='' format(temp,3,1):$BPtemp
addparams('BPtemp',$BPtemp,'H1')
$empty_noise='' format(empty_noise,3,1):$empty_noise
addparams('empty_noise',$empty_noise,'H1')
$empty_noise='' format(empty_noise,3,1):$empty_noise
addparams('empty_noise',$empty_noise,'C13')
$empty_noise='' format(empty_noise,3,1):$empty_noise
addparams('empty_noise',$empty_noise,'N15')
$sample_noise='' format(sample_noise,3,1):$sample_noise
addparams('sample_noise',$sample_noise,'H1')
$sample_noise='' format(sample_noise,3,1):$sample_noise
addparams('sample_noise',$sample_noise,'C13')
$sample_noise='' format(sample_noise,3,1):$sample_noise
addparams('sample_noise',$sample_noise,'N15')
$pwC='' format(pwC,2,1):$pwC
addparams('pwC',$pwC,'C13')
$pwC3db='' format(1.4*pwC,2,1):$pwC3db
addparams('pwC3db',$pwC3db,'C13')
$pwClvl='' format(pwClvl,2,0):$pwClvl
addparams('pwClvl',$pwClvl,'C13')
$compC='' format(compC,4,3):$compC
```

```
 addparams('compC',$compC,'C13')
 $compC3db='' format(compC,4,3):$compC3db
 addparams('compC3db',$compC3db,'C13')
 $dofCO='' format(dof,4,1):$dofCO
 addparams('dofCO',$dofCO,'C13')
 $swCO='' format(sw1,4,1):$swCO
 addparams('swCO',$swCO,'C13')
 addparams('phi7cal','0.0','C13')
 addparams('phi7calP','60.0','C13')
 addparams('phi_CO','0.0','C13')
 addparams('phi_Ca','0.0','C13')
 addparams('phshift3','O.0','C13')
''next three will be set by autocalibrate of gChsqc''
 $CHgrad1_time='' format(gt1,8,6):$CHgrad1_time
 addparams('CHgrad1_time',$CHgrad1_time,'C13')
 $CHgrad1_lvl='' format(gzlvl1,6,0):$CHgrad1_lvl
 addparams('CHgrad1_lvl',$CHgrad1_lvl,'C13')
 $CHgrad2_lvl='' format(gzlvl2,6,0):$CHgrad2_lvl
 addparams('CHgrad2_lvl',$CHgrad2_lvl,'C13')
 $CHgstab='' format(gstab,6,6):$CHgstab
 addparams('CHgstab',$CHgstab,'C13')
''next three will be used by ghcch_tocsy''
 $hcch_gtime='' format(0.0008,8,6):$hcch_gtime
 addparams('hcch_gtime',$hcch_gtime,'C13')
 $hcch_gzlvl1='' format(gzlvl1,6,0):$hcch_gzlvl1
 addparams('hcch_gzlvl1',$hcch_gzlvl1,'C13')
 $hcch_gzlvl2='' format(gzlvl2,6,0):$hcch_gzlvl2
 addparams('hcch_gzlvl2',$hcch_gzlvl2,'C13')
$spinlock='' format(BPspinlock,4,1):$spinlock
 addparams('spinlock',$spinlock,'C13')
 $dmf30='' format(dmf30,4,1):$dmf30
 addparams('dmf30',$dmf30,'C13')
 $dmf80='' format(dmf80,4,1):$dmf80
 addparams('dmf80',$dmf80,'C13')
 $dmf140='' format(dmf140,4,1):$dmf140
 addparams('dmf140',$dmf140,'C13')
 $rf30='' format(rf30,4,1):$rf30
```

```
 addparams('rf30',$rf30,'C13')
 $rf80='' format(rf80,4,1):$rf80
 addparams('rf80',$rf80,'C13')
 $rf140='' format(rf140,4,1):$rf140
 addparams('rf140',$rf140,'C13')
$pwN='' format(pwN,2,1):$pwN
 addparams('pwN',$pwN,'N15')
 $pwNlvl='' format(pwNlvl,2,0):$pwNlvl
 addparams('pwNlvl',$pwNlvl,'N15')
 $compN='' format(compN,4,3):$compN
 addparams('compN',$compN,'N15')
 $dofN15='' format(dof2,4,1):$dofN15
 addparams('dofN15',$dofN15,'N15')
 $swN='' format(sw2,4,1):$swN
 addparams('swN',$swN,'N15')
 $dpwr2NH='' format(dpwr2,2,0):$dpwr2NH
 addparams('dpwr2NH',$dpwr2NH,'N15')
 addparams('dmm2NH',dmm2,'N15')
 addparams('dseq2NH',dseq2,'N15')
 $dres2NH='' format(dres2,5,1):$dres2NH
 addparams('dres2NH',$dres2NH,'N15')
 $dmf2NH='' format(dmf2,3,1):$dmf2NH
 addparams('dmf2NH',$dmf2NH,'N15')
 $JNH='' format(JNH,3,1):$JNH
 addparams('JNH',$JNH,'N15')
$gt0='' format(gt0,1,6):$gt0
 addparams('gt0',$gt0,'N15')
 $gt5='' format(gt5,1,6):$gt5
 addparams('gt5',$gt5,'N15')
 $gzlvl5='' format(gzlvl5,6,0):$gzlvl5
addparams('gzlvl5',$gzlvl5,'N15')
$gzlvl0='' format(gzlvl0,6,0):$gzlvl0
addparams('gzlvl0',$gzlvl0,'N15')
$gzlvl5='' format(gzlvl5,6,0):$gzlvl5
addparams('gzlvl5',$gzlvl5,'N15')
$NHgrad1_time='' format(gt1,8,6):$NHgrad1_time
addparams('NHgrad1_time',$NHgrad1_time,'N15')
```

```
$NHgrad1_lvl='' format(gzlvl1,4,0):$NHgrad1_lvl
addparams('NHgrad1_lvl',$NHgrad1_lvl,'N15')
$NHgrad2_lvl='' format(gzlvl2,4,0):$NHgrad2_lvl
addparams('NHgrad2_lvl',$NHgrad2_lvl,'N15')
$BPgzcal='' format(gzcal,6,6):$BPgzcal
addparams('BPgzcal',$BPgzcal,'N15')
$NHgstab='' format(gstab,6,6):$NHgstab
addparams('NHgstab',$NHgstab,'N15')
addparams('mag_flg',mag_flg,'N15')
$dres3D='' format(dres3,2,1):$dres3D
addparams('dres3D',$dres3D,'H2')
$dpwr3D='' format(dpwr3,2,0):$dpwr3D
addparams('dpwr3D',$dpwr3D,'H2')
$dmf3D='' format(dmf3,5,1):$dmf3D
addparams('dmf3D',$dmf3D,'H2')
$dof3D='' format(dof3,5,1):$dof3D
addparams('dof3D',$dof3D,'H2')
addparams('dseq3D',dseq3,'H2')
addparams('dmm3D',dmm3,'H2')
exists('probconnect','parameter','global'):$e
if ($e>0) then
addparams('probeConnect',probeConnect,'H1')
endif
''Biopack power limits are active if BPpwrlimits=1 ''
exists('BPpwrlimits','parameter','global'):$e
if ($e>0) then
$BPpwrlimits='' format(BPpwrlimits,1,0):$BPpwrlimits
addparams('BPpwrlimits',$BPpwrlimits,'H1')
endif
''1H and 13C spinlock upper limits if BPpwrlimits=1 ''
exists('BPtpwrspinlock','parameter','global'):$e
if ($e>0) then
$BPtpwrspinlock='' format(BPtpwrspinlock,5,0):$BPtpwrspinlock
addparams('BPtpwrspinlock',$BPtpwrspinlock,'H1')
endif
exists('BPdpwrspinlock','parameter','global'):$e
if ($e>0) then
```

```
$BPdpwrspinlock='' format(BPdpwrspinlock,5,0):$BPdpwrspinlock
addparams('BPdpwrspinlock',$BPdpwrspinlock,'C13')
endif
''default values for 13C and 1H spinlock fields''
exists('BPspinlock','parameter','global'):$e
if ($e>0) then
$BPspinlock='' format(BPspinlock,6,0):$BPspinlock
addparams('BPspinlock',$BPspinlock,'C13')
endif
exists('BP1Hspinlock','parameter','global'):$e
if ($e>0) then
 $BP1Hspinlock='' format(BP1Hspinlock,6,0):$BP1Hspinlock
 addparams('BP1Hspinlock',$BP1Hspinlock,'H1')
endif

''1H and 13C decoupling power upper limits if BPpwrlimits=1 ''
exists('BPtpwrmax','parameter','global'):$e
if ($e>0) then
 $BPtpwrmax='' format(BPtpwrmax,2,0):$BPtpwrmax
 addparams('BPtpwrmax',$BPtpwrmax,'H1')
endif

exists('BPdpwrmax','parameter','global'):$e
if ($e>0) then
 $BPdpwrmax='' format(BPdpwrmax,2,0):$BPdpwrmax
 addparams('BPdpwrmax',$BPdpwrmax,'C13')
endif

exists('BPdpwr2max','parameter','global'):$e
if ($e>0) then
 $BPdpwr2max='' format(BPdpwr2max,2,0):$BPdpwr2max
 addparams('BPdpwr2max',$BPdpwr2max,'N15')
endif
exists('BPdpwr3max','parameter','global'):$e
if ($e>0) then
 $BPdpwr3max='' format(BPdpwr3max,2,0):$BPdpwr3max
 addparams('BPdpwr3max',$BPdpwr3max,'H2')
```

```
endif
"System power limits"
exists('maxattench1','parameter','global'):$e1
if ($e1>0) then
 $maxattench1='' format(maxattench1,2,0):$maxattench1
 addparams('maxattench1',$maxattench1,'H1')
endif
exists('maxattench2','parameter','global'):$e1
if ($e1>0) then
 $maxattench2='' format(maxattench2,2,0):$maxattench2
 addparams('maxattench2',$maxattench2,'C13')
endif
exists('maxattench3','parameter','global'):$e1
if ($e1>0) then
 $maxattench3='' format(maxattench3,2,0):$maxattench3
 addparams('maxattench3',$maxattench3,'N15')
endif
exists('maxattench4','parameter','global'):$e1
if ($e1>0) then
 $maxattench4='' format(maxattench4,2,0):$maxattench4
 addparams('maxattench4',$maxattench4,'H2')
endif
banner('probefile setup finished.')
```

9.5 The BPbiopack2 macro

```
"BPbiopack2   updates parameters in probefile using current values"
/* This section makes sure a probefile exists */
$probename=probe
if ($probename='') then
  banner('No Probe is specified. New Probefile "HCN" is being created')
  BPaddprobe('HCN')
endif

/* This section checks if power limits are reasonable */
if (numrfch>3) then
 if (BPpwrlimits=1) then
  if ((BPtpwrmax<30) or (BPdpwrmax<30) or (BPspinlock<2000) or (BPdpwr2max<30) or
```

```
(BPdpwr3max<30)) then
   banner('BioPack Power Limits need to be set on all four channels, or unset BioPack Power Limits. Update aborting')
   return
  endif
  BPcheck "adjusts tpwr/dpwr/dpwr2/dpwr3 if BPpwrlimits=1"
 endif
elseif (numrfch>2) then
 if (BPpwrlimits=1) then
  if ((BPtpwrmax<30) or (BPdpwrmax<30) or (BPspinlock<2000) or (BPdpwr2max<30)) then
   banner('BioPack Power Limits need to be set on all three channels, or unset BioPack Power Limits. Update aborting')
   return
  endif
  BPcheck "adjusts tpwr/dpwr/dpwr2/dpwr3 if BPpwrlimits=1"
 endif
else
 if (BPpwrlimits=1) then
  if ((BPtpwrmax<30) or (BPdpwrmax<30) or (BPspinlock<2000)) then
   banner('BioPack Power Limits need to be set on all channels, or unset BioPack Power Limits. Update aborting')
   return
  endif
  BPcheck "adjusts tpwr/dpwr/dpwr2 if BPpwrlimits=1"
 endif
endif
banner('Probefile Is Being Updated with These Calibrations')

/* This section makes sure pulse widths are multiple of 0.1 usec */ pw=trunc(10*pw + 0.5)/10
pw90=trunc(10*pw + 0.5)/10
pwC=trunc(10*pwC + 0.5)/10
pwN=trunc(10*pwN + 0.5)/10
ref_pw90=trunc(10*pw*compH + 0.5)/10 ref_pwr=tpwr
fn=np np=fn fn=fn*2

BPN15updatepw90   “updates N15 section of probefile”

/* This section creates parameters if missing */
```

```
exists('autocal','parameter'):$e
if $e=0 then create('autocal','string') autocal='y' endif
exists('checkofs','parameter'):$e
if $e=0 then create('checkofs','string') checkofs='n' endif
exists('JNH','parameter'):$e
if $e=0 then create('JNH','real') JNH=93 endif
exists('waltzB1','parameter'):$e
if $e=0 then create('waltzB1','real') waltzB1=5p endif
exists('dmf30','parameter'):$e
if $e=0 then create('dmf30','real') dmf30=0 endif
exists('dmf80','parameter'):$e
 if $e=0 then create('dmf80','real') dmf80=0 endif
exists('dmf140','parameter'):$e
 if $e=0 then create('dmf140','real') dmf140=0 endif
exists('rf30','parameter'):$e
 if $e=0 then create('rf30','real') rf30=0 endif
exists('rf80','parameter'):$e
 if $e=0 then create('rf80','real') rf80=0 endif
exists('rf140','parameter'):$e
 if $e=0 then create('rf140','real') rf140=0 endif

/* Saves ghn_co parameter set in user (or vnmr) parlib */
if seqfil='ghn_co' then
 ai f full setref BPsvp('ghn_co')
endif

/* This section updates processing parameters. BPsetparams is a
   version of setparams that creates and sets a parameter if it
   is not present in the probefile  */
$lb='' format(lb,4,4):$lb
BPsetparams('lb',$lb,'H1')
$gf='' format(gf,4,4):$gf
BPsetparams('gf',$gf,'H1')
$gfs='' format(gfs,4,4):$gfs
BPsetparams('gfs',$gfs,'H1')
$sb='' format(sb,4,4):$sb
```

```
BPsetparams('sb',$sb,'H1')
$sbs='' format(sbs,4,4):$sbs
BPsetparams('sbs',$sbs,'H1')

$ssfilter='' format(ssfilter,4,0):$ssfilter
BPsetparams('ssfilter',$ssfilter,'H1')
$ssntaps='' format(ssntaps,4,0):$ssntaps
BPsetparams('ssntaps',$ssntaps,'H1')
$sslsfrq='' format(sslsfrq,4,4):$sslsfrq
BPsetparams('sslsfrq',$sslsfrq,'H1')
$ssorder='' format(ssorder,2,0):$ssorder
BPsetparams('ssorder',$ssorder,'H1')

/* This section updates acquisition parameters */
$rof1='' format(rof1,4,2):$rof1
BPsetparams('rof1',$rof1,'H1')
$rof2='' format(rof2,4,2):$rof2
BPsetparams('rof2',$rof2,'H1')
$alfa='' format(alfa,4,2):$alfa
BPsetparams('alfa',$alfa,'H1')
if (sw<14p) then sw=14p endif
$sw='' format(sw,6,1):$sw
BPsetparams('sw',$sw,'H1')
$at='' format(at,5,4):$at
BPsetparams('at',$at,'H1')
$BPd1='' format(d1,2,3):$BPd1
BPsetparams('BPd1',$BPd1,'H1')
$BPgain='' format(gain,2,0):$BPgain
BPsetparams('BPgain',$BPgain,'H1')
$BPtemp='' format(temp,3,1):$BPtemp
BPsetparams('BPtemp',$BPtemp,'H1')

/* This section updates RF parameters */

$ref_pw90='' format(ref_pw90,2,1):$ref_pw90
BPsetparams('ref_pw90',$ref_pw90,'H1')
$ref_pwr='' format(ref_pwr,2,0):$ref_pwr
```

```
BPsetparams('ref_pwr',$ref_pwr,'H1')
$pw90='' format(pw90,2,1):$pw90
BPsetparams('pp',$pw90,'H1')
BPsetparams('pw90',$pw90,'H1')
$waltzB1='' format(waltzB1,6,0):$waltzB1
BPsetparams('waltzB1',$waltzB1,'H1')
$tpwr='' format(tpwr,2,0):$tpwr
BPsetparams('tpwr',$tpwr,'H1')
BPsetparams('pplvl',$tpwr,'H1')
$dmf=1e6/(pw*10*compH) $dpwr=tpwr-20
$dmfH1='' format($dmf,5,0):$dmfH1
$dpwrH1='' format($dpwr,2,0):$dpwrH1
BPsetparams('dmf',$dmfH1,'H1')
BPsetparams('dpwr',$dpwrH1,'H1')
$tofH2O='' format(tof,4,1):$tofH2O
BPsetparams('tofH2O',$tofH2O,'H1')
$compH='' format(compH,4,3):$compH
BPsetparams('compH',$compH,'H1')
BPsetparams('tpwr_cf',$compH,'H1')

$pwC='' format(pwC,2,1):$pwC
BPsetparams('pwC',$pwC,'C13')
BPsetparams('pw90',$pwC,'C13')
BPsetparams('pwx',$pwC,'C13')
$pwClvl='' format(pwClvl,2,0):$pwClvl
BPsetparams('pwClvl',$pwClvl,'C13')
BPsetparams('tpwr',$pwClvl,'C13')
BPsetparams('pwxlvl',$pwClvl,'C13')
$compC='' format(compC,4,3):$compC
BPsetparams('compC',$compC,'C13')
BPsetparams('pwxlvl_cf',$compC,'C13')

$BPgzcal='' format(gzcal,8,6):$BPgzcal
BPsetparams('gcal',$BPgzcal,'Probe')
BPsetparams('gradient','y','Probe')
BPsetparams('BPgzcal',$BPgzcal,'N15')
BPsetparams('mag_flg',mag_flg,'N15')
```

```
if (seqfil='ghn_co') then
$pwHs='' format(pwHs,2,1):$pwHs
BPsetparams('pwHs',$pwHs,'H1')
$tpwrsf='' format(tpwrsf,4,0):$tpwrsf
BPsetparams('tpwrsf',$tpwrsf,'H1')
$dofCO='' format(dof,4,1):$dofCO
BPsetparams('dofCO',$dofCO,'C13')
$swCO='' format(sw1,4,1):$swCO
BPsetparams('swCO',$swCO,'C13')
$spinlock='' format(BPspinlock,4,1):$spinlock
  BPsetparams('spinlock',$spinlock,'C13')
  $dofN15='' format(dof2,4,1):$dofN15
  BPsetparams('dofN15',$dofN15,'N15')
  $swN='' format(sw2,4,1):$swN
  BPsetparams('swN',$swN,'N15')
  $JNH='' format(JNH,3,1):$JNH
  BPsetparams('JNH',$JNH,'N15')
  $NHgrad1_time='' format(gt1,8,6):$NHgrad1_time
  BPsetparams('NHgrad1_time',$NHgrad1_time,'N15')
  $NHgrad1_lvl='' format(gzlvl1,6,0):$NHgrad1_lvl
  BPsetparams('NHgrad1_lvl',$NHgrad1_lvl,'N15')
  $NHgrad2_lvl='' format(gzlvl2,6,0):$NHgrad2_lvl
  BPsetparams('NHgrad2_lvl',$NHgrad2_lvl,'N15')
  $NHgstab='' format(gstab,6,4):$NHgstab
  BPsetparams('NHgstab',$NHgstab,'N15')

  $gt5='' format(gt5,1,6):$gt5
  BPsetparams('gt5',$gt5,'N15')
  $gt0='' format(gt0,1,6):$gt0
  BPsetparams('gt0',$gt0,'N15')
  $gzlvl5='' format(gzlvl5,6,0):$gzlvl5
  BPsetparams('gzlvl5',$gzlvl5,'N15')
  $gzlvl0='' format(gzlvl0,6,0):$gzlvl0
  BPsetparams('gzlvl0',$gzlvl0,'N15')
  $dres3D='' format(dres3,2,1):$dres3D
  BPsetparams('dres3D',$dres3D,'H2')
```

“sequence-specific parameters”

```
  $dpwr3D='' format(dpwr3,2,0):$dpwr3D
  BPsetparams('dpwr3D',$dpwr3D,'H2')
  $dmf3D='' format(dmf3,5,1):$dmf3D
  BPsetparams('dmf3D',$dmf3D,'H2')

$dof3D='' format(dof3,5,1):$dof3D
  BPsetparams('dof3D',$dof3D,'H2')
  if dseq3='' then dseq3='garp1' endif
  BPsetparams('dseq3D',dseq3,'H2')
  BPsetparams('dmm3D',dmm3,'H2')
  BPsetupshapes                                    "makes Pbox waveforms"
  BPcal                                            "makes rf waveforms with macros"
  BPmakeSTUDpp    "make STUD waveforms and set dmf30, rf30, etc"

  $dmf30='' format(dmf30,4,1):$dmf30
  BPsetparams('dmf30',$dmf30,'C13')
  $dmf80='' format(dmf80,4,1):$dmf80
  BPsetparams('dmf80',$dmf80,'C13')
  $dmf140='' format(dmf140,4,1):$dmf140
  BPsetparams('dmf140',$dmf140,'C13')
  $rf30='' format(rf30,4,1):$rf30
  BPsetparams('rf30',$rf30,'C13')
  $rf80='' format(rf80,4,1):$rf80
  BPsetparams('rf80',$rf80,'C13')
  $rf140='' format(rf140,4,1):$rf140
  BPsetparams('rf140',$rf140,'C13')
endif

if (seqfil='rna_gNhsqc') then
  $NHgrad1_time='' format(gt1,8,6):$NHgrad1_time
  BPsetparams('NHgrad1_time',$NHgrad1_time,'N15')
  $NHgrad1_lvl='' format(gzlvl1,6,0):$NHgrad1_lvl
  BPsetparams('NHgrad1_lvl',$NHgrad1_lvl,'N15')
  $NHgrad2_lvl='' format(gzlvl2,6,0):$NHgrad2_lvl
  BPsetparams('NHgrad2_lvl',$NHgrad2_lvl,'N15')
  $NHgstab='' format(grecov,6,4):$NHgstab
  BPsetparams('NHgstab',$NHgstab,'N15')
```

```
endif

if (seqfil='rna_gChsqc') then
$CHgrad1_time='' format(gt1,8,6):$CHgrad1_time
BPsetparams('CHgrad1_time',$CHgrad1_time,'C13')
$CHgrad1_lvl='' format(gzlvl1,6,0):$CHgrad1_lvl
BPsetparams('CHgrad1_lvl',$CHgrad1_lvl,'C13')
$CHgrad2_lvl='' format(gzlvl2,6,0):$CHgrad2_lvl
BPsetparams('CHgrad2_lvl',$CHgrad2_lvl,'C13')
$CHgstab='' format(grecov,6,4):$CHgstab
BPsetparams('CHgstab',$CHgstab,'C13')
BPcal                                          "makes rf waveforms"
BPmakeSTUDpp  "makes STUD waveforms and sets dmf30, rf30, etc"
$dmf30='' format(dmf30,4,1):$dmf30
BPsetparams('dmf30',$dmf30,'C13')
$dmf80='' format(dmf80,4,1):$dmf80
BPsetparams('dmf80',$dmf80,'C13')
$dmf140='' format(dmf140,4,1):$dmf140
BPsetparams('dmf140',$dmf140,'C13')
$rf30='' format(rf30,4,1):$rf30
BPsetparams('rf30',$rf30,'C13')
$rf80='' format(rf80,4,1):$rf80
BPsetparams('rf80',$rf80,'C13')
$rf140='' format(rf140,4,1):$rf140
BPsetparams('rf140',$rf140,'C13')
endif

exists('probeConnect','parameter','global'):$e
if ($e>0) then
 BPsetparams('probeConnect',probeConnect,'H1')
endif
''Biopack power limits are active if BPpwrlimits=1 ''
exists('BPpwrlimits','parameter','global'):$e
if ($e>0) then
$BPpwrlimits='' format(BPpwrlimits,1,0):$BPpwrlimits
BPsetparams('BPpwrlimits',$BPpwrlimits,'H1')
endif
```

```
"1H and 13C spinlock upper limits if BPpwrlimits=1 "
exists('BPtpwrspinlock','parameter','global'):$e
if ($e>0) then
 $BPtpwrspinlock='' format(BPtpwrspinlock,5,0):$BPtpwrspinlock
 BPsetparams('BPtpwrspinlock',$BPtpwrspinlock,'H1')
endif

exists('BPdpwrspinlock','parameter','global'):$e
if ($e>0) then
$BPdpwrspinlock='' format(BPdpwrspinlock,5,0):$BPdpwrspinlock
BPsetparams('BPdpwrspinlock',$BPdpwrspinlock,'C13')
endif
"default values for 13C and 1H spinlock fields"
exists('BPspinlock','parameter','global'):$e
if ($e>0) then
$BPspinlock='' format(BPspinlock,6,0):$BPspinlock
BPsetparams('BPspinlock',$BPspinlock,'C13')
endif

exists('BP1Hspinlock','parameter','global'):$e
if ($e>0) then
 $BP1Hspinlock='' format(BP1Hspinlock,6,0):$BP1Hspinlock
 BPsetparams('BP1Hspinlock',$BP1Hspinlock,'H1')
endif

"1H and 13C decoupling power upper limits if BPpwrlimits=1 "
exists('BPtpwrmax','parameter','global'):$e
if ($e>0) then
 $BPtpwrmax='' format(BPtpwrmax,2,0):$BPtpwrmax
 BPsetparams('BPtpwrmax',$BPtpwrmax,'H1')
endif

exists('BPdpwrmax','parameter','global'):$e
if ($e>0) then
 $BPdpwrmax='' format(BPdpwrmax,2,0):$BPdpwrmax
 BPsetparams('BPdpwrmax',$BPdpwrmax,'C13')
```

```
endif

exists('BPdpwr2max','parameter','global'):$e
if ($e>0) then
 $BPdpwr2max='' format(BPdpwr2max,2,0):$BPdpwr2max
 BPsetparams('BPdpwr2max',$BPdpwr2max,'N15')
endif

exists('BPdpwr3max','parameter','global'):$e
if ($e>0) then
 $BPdpwr3max='' format(BPdpwr3max,2,0):$BPdpwr3max
 BPsetparams('BPdpwr3max',$BPdpwr3max,'H2')
endif

''System power limits''
exists('maxattench1','parameter','global'):$e
if ($e>0) then
 $maxattench1='' format(maxattench1,2,0):$maxattench1
 BPsetparams('maxattench1',$maxattench1,'H1')
endif

exists('maxattench2','parameter','global'):$e
if ($e>0) then
 $maxattench2='' format(maxattench2,2,0):$maxattench2
 BPsetparams('maxattench2',$maxattench2,'C13')
endif

exists('maxattench3','parameter','global'):$e
if ($e>0) then
 $maxattench3='' format(maxattench3,2,0):$maxattench3
 BPsetparams('maxattench3',$maxattench3,'N15')
endif
exists('maxattench4','parameter','global'):$e
if ($e>0) then
 $maxattench4='' format(maxattench4,2,0):$maxattench4
 BPsetparams('maxattench4',$maxattench4,'H2')
endif
```

```
write('line3','Probefile Update Finished')
banner('Probefile Updated with These Calibrations')
```

Chapter 10 Making Shapelib Files: RF and Decoupling Waveforms

Several macros are used within the probefile update process. BPsetupshapes and BPcal are macros that create waveforms for decoupling and band-selective excitation. They are executed during the probefile update because this is when (possibly) new calibrations have been determined. Since decoupling waveforms depend on reference pw90's, power levels and compression factors, having the waveforms recreated during BPbiopack2 execution guarantees that the waveforms and their internal parameters are consistent with probe calibrations. Once the waveforms have been created, the BPsetwurstparams macro can be used in experiment setup macros (e.g. "gChsqc") to install calibrated parameter values for decoupling and spinlocks. Note that probefile updates using BPbiopack2 are initiated by buttons on the Globals&Probefile page in the Setup folder.

10.1 The BPsetupshapes macro

/* The macro prepares band-selective 13C decoupling waveforms. It then stores each in the appropriate shapelib. if BPpwrlimits=1, power levels may reduced, but bandwidths remain, at the expense of increased sidebands */

/* The macro also prepares solvent suppression 1H waveforms. It then stores each in the appropriate shapelibs */

```
$bw = ''
$wave = ''
$name = ''
$n3=n3
format(BPdpwrmax,2,0):n3
n3=n3+'d'
$attn=n3
exists('BPspinlock','parameter','global'):$e2
if $e2=1 then
if BPspinlock=0 then
BPspinlock=50d                    "set C-13 spinlock bandwidth to default value"
endif
else
create('BPspinlock','real','global')
BPspinlock=50d                    "set C-13 spinlock bandwidth to default value"
endif
```

"check BPspinlock relative to user-defined maximum BPdpwrspinlock"
if BPpwrlimits=1 then
if BPdpwrspinlock<BPspinlock then
BPspinlock=BPdpwrspinlock
endif
endif
" **set C-13 spinlock bandwidth to global stored value**"
format(BPspinlock,9,1):$bw
$wave = 'WURST2m' + $bw + '/0.08ms'
opx **" setup CC spinlock "**
setwave($wave)
pbox_rst
pboxpar('name', 'cctocsy.DEC')
pboxpar('stepsize', '1.0')
pboxpar('sfrq', dfrq)
pboxpar('sucyc', 't9,t5,m4')
pboxpar('ref_pwr', pwClvl)
pboxpar('ref_pw90', pwC*compC)
" pboxpar('reps', '0') - to suppress the Pbox output "
shell('Pbox')
if BPpwrlimits=1 then
BPpboxget('cctocsy.DEC'):$name,$par,$dpwr,$par,$par,$par
if $dpwr>BPdpwrmax then
pboxpar('attn',$attn)
echo($attn)
shell('Pbox')
endif
endif
BPmoveshape('cctocsy.DEC')
format(90.0*sfrq/4.0,9,1):$ " set C-13 decoupling bandwidth to 80 ppm "
if BPpwrlimits=1 then
 $wave = 'WURST40' + $bw + '/1.4ms'
else
 $wave = 'WURST2' + $bw + '/1.01ms'
endif
opx
setwave($wave)

```
pbox_rst
if BPpwrlimits=0 then
 pboxpar('$1 adb','3.0')
"this can be omitted for 3db less power, but > sideband"
endif
pboxpar('name', 'wurst80.DEC')
pboxpar('ref_pwr', pwClvl)
pboxpar('ref_pw90', pwC*compC)
" pboxpar('reps', '0')  - to suppress the Pbox output "
shell('Pbox')
if BPpwrlimits=1 then
BPpboxget('wurst80.DEC'):$name,$par,$dpwr,$par,$par,$par
if $dpwr>BPdpwrmax then
pboxpar('attn',$attn)
echo($attn)
shell('Pbox')
endif
endif
BPmoveshape('wurst80.DEC')
" set C-13 decoupling bandwidth to 110 ppm "
format(130.0*sfrq/4.0,9,1):$bw
if BPpwrlimits=1 then
$wave = 'WURST40' + $bw + '/1.4ms'
else
$wave = 'WURST2' + $bw + '/1.01ms'
endif
opx
setwave($wave)
pbox_rst
pboxpar('name', 'wurst110.DEC')
pboxpar('ref_pwr', pwClvl)
pboxpar('ref_pw90', pwC*compC)
" pboxpar('reps', '0')  - to suppress the Pbox output "
shell('Pbox')
if BPpwrlimits=1 then
 BPpboxget('wurst110.DEC'):$name,$par,$dpwr,$par,$par,$par
 if $dpwr>BPdpwrmax then
```

```
    pboxpar('attn',$attn)
    echo($attn)
    shell('Pbox')
 endif
endif
BPmoveshape('wurst110.DEC')

" set C-13 decoupling bandwidth to 50 ppm "
format(60.0*sfrq/4.0,9,1):$bw
if BPpwrlimits=1 then
$wave = 'WURST40' + $bw + '/1.4ms'
else
$wave = 'WURST2' + $bw + '/1.01ms'
endif
opx
setwave($wave)
pbox_rst
if BPpwrlimits=0 then
pboxpar('$1 adb','3.0') "this can be omitted for 3db less power, but > sideband"
endif
pboxpar('name', 'wurst50.DEC')
pboxpar('ref_pwr', pwClvl)
pboxpar('ref_pw90', pwC*compC)
"pboxpar('reps', '0')  - to suppress the Pbox output "
shell('Pbox')
if BPpwrlimits=1 then
BPpboxget('wurst50.DEC'):$name,$par,$dpwr,$par,$par,$par
if $dpwr>BPdpwrmax then
pboxpar('attn',$attn)
echo($attn)
shell('Pbox')
endif
endif
BPmoveshape('wurst50.DEC')

" set C-13 decoupling bandwidth to 40 ppm "
format(45.0*sfrq/4.0,9,1):$bw
```

```
if BPpwrlimits=1 then
$wave = 'WURST40' + $bw + '/1.4ms'
else
$wave = 'WURST2' + $bw + '/1.01ms'
endif
opx
setwave($wave)
pbox_rst
if BPpwrlimits=0 then
pboxpar('$1 adb','3.0') "this can be omitted for 3db less power, but > sideband"
endif
pboxpar('name', 'wurst40.DEC')
pboxpar('ref_pwr', pwClvl)
pboxpar('ref_pw90', pwC*compC)
"pboxpar('reps', '0')  - to suppress the Pbox output "
shell('Pbox')
if BPpwrlimits=1 then
BPpboxget('wurst40.DEC'):$name,$par,$dpwr,$par,$par,$par
if $dpwr>BPdpwrmax then
pboxpar('attn',$attn)
echo($attn)
shell('Pbox')
endif
endif
BPmoveshape('wurst40.DEC')
" set C-13 decoupling bandwidth to 140 ppm "
format(160.0*sfrq/4.0,9,1):$bw
if BPpwrlimits=1 then
 $wave = 'WURST40' + $bw + '/1.4ms'
else
 $wave = 'WURST2' + $bw + '/1.01ms'
endif
opx
setwave($wave)
pbox_rst
if BPpwrlimits=0 then
 pboxpar('$1 adb','3.0') "this can be omitted for 3db less power, but > sideband"
```

```
endif
pboxpar('name', 'wurst140.DEC')
pboxpar('ref_pwr', pwClvl)
pboxpar('ref_pw90', pwC*compC)
" pboxpar('reps', '0')  - to suppress the Pbox output "
shell('Pbox')
if BPpwrlimits=1 then
 BPpboxget('wurst140.DEC'):$name,$par,$dpwr,$par,$par,$par
 if $dpwr>BPdpwrmax then
    pboxpar('attn',$attn)
    echo($attn)
    shell('Pbox')
 endif
endif
BPmoveshape('wurst140.DEC')

opx                                    " set up for Cbeta decoupling in separate bands "
setwave('WURST2 22p/5m 36p')   " main Cb band "
setwave('WURST2  6p/5m 71p')   " Threonines "
setwave('WURST2  6p/5m 15p')   " Alanines "
pbox_rst
pboxpar('name', 'cbdec.DEC')
pboxpar('stepsize', '4.0')
pboxpar('sfrq', dfrq)
pboxpar('refofs', '56p')
pboxpar('sucyc', 't5')
pboxpar('ref_pwr', pwClvl)
pboxpar('ref_pw90', pwC*compC)
" pboxpar('reps', '0')  - to suppress the Pbox output "
pboxpar('$1 adb', '2.0') "more power for CO dec as it is further away"
pboxpar('$2 adb', '1.2')
pboxpar('$3 adb', '1.2')
pboxpar('$4 adb', '1.2')
shell('Pbox')
if BPpwrlimits=1 then
 BPpboxget('cbdec.DEC'):$name,$par,$dpwr,$par,$par,$par
 if $dpwr>BPdpwrmax then
```

```
   pboxpar('attn',$attn)
   echo($attn)
   shell('Pbox')
 endif
endif
BPmoveshape('cbdec.DEC')

Opx                                  “set Cb decoupling”
setwave('WURST2 22p/5m 36p')          " main Cb band "
setwave('WURST2  6p/5m 71p')          " Threonines "
setwave('WURST2  6p/5m 15p')          " Alanines "
pbox_rst
pboxpar('name', 'cb43dec.DEC')        "c13 will be at 43ppm for this"
pboxpar('stepsize', '4.0')
pboxpar('sfrq', dfrq)
pboxpar('refofs', '43p')
pboxpar('sucyc', 't5')
pboxpar('ref_pwr', pwClvl)
pboxpar('ref_pw90', pwC*compC)
" pboxpar('reps', '0')  - to suppress the Pbox output "
pboxpar('$1 adb', '2.0') "more power for CO dec as it is further away"
pboxpar('$2 adb', '1.2')
pboxpar('$3 adb', '1.2')
pboxpar('$4 adb', '1.2')
shell('Pbox')
if BPpwrlimits=1 then
 BPpboxget('cb43dec.DEC'):$name,$par,$dpwr,$par,$par,$par
 if $dpwr>BPdpwrmax then
   pboxpar('attn',$attn)
   echo($attn)
   shell('Pbox')
 endif
endif
BPmoveshape('cb43dec.DEC')

opx                                  " set CO  and Cbeta decoupling "
setwave('WURST2 20p/5m 175p')        " CO decoupling "
```

```
setwave('WURST2 22p/5m 36p')          " main Cb band "
setwave('WURST2  6p/5m 71p')          " Threonines "
setwave('WURST2  6p/5m 15p')          " Alanines "
pbox_rst
pboxpar('name', 'cocbdec.DEC')
pboxpar('stepsize', '4.0')
pboxpar('sfrq', dfrq)
pboxpar('refofs', '56p')
pboxpar('sucyc', 't5')
pboxpar('ref_pwr', pwClvl)
pboxpar('ref_pw90', pwC*compC)
" pboxpar('reps', '0')  - to suppress the Pbox output "
pboxpar('$1 adb', '2.0') "more power for CO dec as it is further away"
pboxpar('$2 adb', '1.2')
pboxpar('$3 adb', '1.2')
pboxpar('$4 adb', '1.2')
shell('Pbox')
if BPpwrlimits=1 then
 BPpboxget('cocbdec.DEC'):$name,$par,$dpwr,$par,$par,$par
 if $dpwr>BPdpwrmax then
    pboxpar('attn',$attn)
    echo($attn)
    shell('Pbox')
 endif
endif
BPmoveshape('cocbdec.DEC')

opx                                   " set CO and Ca decoupling "
setwave('WURST2 20p/5m 174p')         " CO decoupling "
setwave('WURST2 22p/5m 30p')          " Ca decoupling "
pbox_rst
pboxpar('name', 'cocadec.DEC')
pboxpar('stepsize', '4.0')
pboxpar('sfrq', dfrq)
pboxpar('refofs', '56p')
pboxpar('sucyc', 't5')
pboxpar('ref_pwr', pwClvl)
```

```
pboxpar('ref_pw90', pwC*compC)
pboxpar('$1 adb', '2.0') "more power for CO dec as it is further away"
pboxpar('$2 adb', '1.2')
shell('Pbox')
if BPpwrlimits=1 then
 BPpboxget('cocadec.DEC'):$name,$par,$dpwr,$par,$par,$par
 if $dpwr>BPdpwrmax then
    pboxpar('attn',$attn)
    echo($attn)
    shell('Pbox')
 endif
endif
BPmoveshape('cocadec.DEC')
opx          " set CO decoupling "
setwave('WURST2 20p/5m 174p')                    " CO decoupling "
pbox_rst
pboxpar('name', 'codec.DEC')
pboxpar('stepsize', '4.0')
pboxpar('sfrq', dfrq)
pboxpar('refofs', '35p')
pboxpar('sucyc', 't5')
pboxpar('ref_pwr', pwClvl)
pboxpar('ref_pw90', pwC*compC)
pboxpar('$1 adb', '2.0') "more power for CO dec as it is further away"
shell('Pbox')
if BPpwrlimits=1 then
 BPpboxget('codec.DEC'):$name,$par,$dpwr,$par,$par,$par
 if $dpwr>BPdpwrmax then
    pboxpar('attn',$attn)
    echo($attn)
    shell('Pbox')
 endif
endif
BPmoveshape('codec.DEC')

opx               " CO decoupling when dof is at alpha C  "
setwave('WURST2 20p/5m 174p')                    " CO decoupling "
```

```
pbox_rst
pboxpar('name', 'codec2.DEC')
pboxpar('stepsize', '4.0')
pboxpar('sfrq', dfrq)
pboxpar('refofs', '56p')
pboxpar('sucyc', 't5')
pboxpar('ref_pwr', pwClvl)
pboxpar('ref_pw90', pwC*compC)
pboxpar('$1 adb', '2.0') "more power for CO dec as it is further away"
shell('Pbox')
if BPpwrlimits=1 then
 BPpboxget('codec2.DEC'):$name,$par,$dpwr,$par,$par,$par
 if $dpwr>BPdpwrmax then
    pboxpar('attn',$attn)
    echo($attn)
    shell('Pbox')
 endif
endif
BPmoveshape('codec2.DEC')
$bw1 = ''
$wave1 = ''
$offset = ''
$dmmyshp=''
$freq = (150-46)*dfrq
" distance from Cab to the middle of CO and aromatic region"
format(80.0*dfrq,9,1):$bw1   " set decoupling bandwidth to 90 ppm "
format($freq,9,1):$offset
$wave1 = 'g3' + $bw1 + '/0.005' + $offset +'0 0' "ph=0 st=0"
opx
setwave($wave1)
pbox_rst
pboxpar('name', 'CgCO1.RF')
pboxpar('ref_pwr', pwClvl)
pboxpar('ref_pw90', pwC*compC)
" pboxpar('reps', '0')  - to suppress the Pbox output "
shell('Pbox')
$bw1 = ''
```

```
$wave1 = ''
$offset = ''
$freq = (150-46)*dfrq
'' distance from Cab to the middle of CO and aromatic region''
format(80.0*dfrq,9,1):$bw1   '' set decoupling bandwidth to 90 ppm ''
format($freq,9,1):$offset
$wave1 = 'g3' + $bw1 + '/0.005' + $offset +'0 1' ''ph=0 st=1 time-inversed''
opx
setwave($wave1)
pbox_rst
pboxpar('name', 'CgCO2.RF')
pboxpar('ref_pwr', pwClvl)
pboxpar('ref_pw90', pwC*compC)
'' pboxpar('reps', '0')  - to suppress the Pbox output ''
shell('Pbox')
BPmoveshape('CgCO1.RF')
BPmoveshape('CgCO2.RF')

''make CaCO.DEC''
$bw1 = ''
$bw2 = ''
$wave1 = ''
$wave2 = ''
$offset = ''

$freq = -(174-56)*dfrq              '' distance to alpha carbon region''
format(40.0*dfrq,9,1):$bw1          '' set Ca decoupling bandwidth to 40 ppm ''
format(20.0*dfrq,9,1):$bw2          '' set CO decoupling bandwidth to 20 ppm ''
format($freq,9,1):$offset
$wave1 = 'WURST2' + $bw1 + '/1.01ms' + $offset
$wave2 = 'WURST2' + $bw2 + '/1.01ms'
opx
setwave($wave1)
setwave($wave2)
pbox_rst
pboxpar('name', 'CaCO.DEC')
pboxpar('ref_pwr', pwClvl)
```

pboxpar('ref_pw90', pwC*compC)

" pboxpar('reps', '0') - to suppress the Pbox output "

shell('Pbox')

n3=$n3

BPmoveshape(‘CaCo.DEC’)

10.2 The BPcal macro

This macro runs other macros, each of which produces an RF shaped pulse file that is stored in the appropriate shapelib. The basis of this is that sfrq is known to the macro so that phase-ramped shaped pulses may be created with proper pulse widths and phase ramps to achieve the desired excitation. Pulse widths can be calculated in each macro for the desired effect. The same equations are used in the experiment setup macros. Shapes are first created for the maximum number of steps possible (200nsec per step) and these are usually rectangular or sinc-shaped. If off-resonance excitation is desired, a phase ramp is imposed on the shape using the BPconvolute macro. The end results are shapes that are used with specific pulse widths, but the amplitudes are not determined. These are determined by the pulse sequence code itself (except for those sequences directly using Pbox to produce the shaped pulses).

" BPcal"

"For version 5.1 software and later"

" Calculates all the shaped pulses for the BioPack pulse sequences. "

" Requires no prior calibration"

" written by Robin Bendall, Varian, March 95, updated January 1997"

" modified for BioPack, G. Gray, August 2002"

exists(userdir+'/shapelib','directory'):$exists

if $exists<1 then mkdir(userdir+'/shapelib') endif

/* 180 degree inversion pulse across 200ppm, centered on dof, to invert all C13 nuclei when dof=70ppm */

BPmakestC200

/*180 degree inversion pulse across 140ppm, centered on dof, to invert all C13 nuclei when dof=70ppm */

BPmakestC140

/* 180 degree inversion pulse across 80ppm, centered on dof, to invert all aliphatic C13 nuclei without perturbing protein aromatic or CO C13 nuclei when dof=35ppm*/

BPmakestC80

/* 180 degree inversion pulse across 30ppm, centered on dof, to invert all aromatic C13 nuclei without perturbing protein aliphatic C13 nuclei when dof=125ppm, OR to invert Ca's without perturbing protein aromatic or CO C13 nuclei when dof=56ppm */

BPmakestC30

/* 3 ms 180 degree pulse across 50ppm, with the 10% left-hand edge of the inversion at 50 ppm when dof=56ppm, to invert Cb's to the right of the Ca's */

BPmakestC50_3r

/* 5 ms 180 degree pulse across 50ppm, with the 3% left-hand edge of the inversion at 50 ppm when dof=56ppm, to invert Cb's to the right of the Ca's */

PmakestC50_5r

/* 3 ms 180 degree pulse across 50ppm, with the 10% right-hand edge of inversion at 63 ppm when dof=56ppm, to invert Cb's to the left of the Ca's A narrower inversion can be used, but it is handy for all other aspects to be the same as stC50r */

BPmakestC50_3l

/* 5 ms 180 degree pulse across 50ppm, with the 3% right-hand edge of the inversion at 63 ppm when dof=56ppm, to invert Cb's to the left of the Ca's a narrower inversion can be used, but it is handy for all other aspects to be the same as stC50r */

BPmakestC50_5l

/* 10 ms 180 degree pulse across 12.5ppm, from 50ppm to 62.5ppm relative to dof at 35ppm, to invert Ca's and not touch almost all Cb's/

BPmakestC12_10

/* 180 degree inversion pulse across 30ppm, centered on dof, to invert all RNA aromatic C2/C6/C8 (dof=150ppm) C13 nuclei without perturbing RNA pyrimidine C5 and ribose C13 nuclei */

rna_makestC30

/* **180 degree inversion pulse across 50ppm, centered on dof, to invert all RNA ribose and pyrimidine C5 (dof=85ppm) C13 nuclei without perturbing RNA aromatic C2/C6/C8 C13 nuclei** */

rna_makestC50

/* 180 degree inversion pulse across 80ppm, centered on dof, to invert all RNA aromatic C5/C6/C8 (dof=125ppm) C13 nuclei perturbing RNA ribose nuclei */

rna_makestC80

/* 180 degree inversion pulse across 60ppm, centered on dof, to invert all RNA ribose C1' and aromatic C6/C8 (dof=115ppm) C13 nuclei without perturbing RNA ribose C2' and aromatic C2 C13 nuclei */

rna_makestC60

/* 180 degree inversion pulse across 140ppm, centered on dof, to invert all C13 nuclei when dof=110ppm */

rna_makestC140

/* 180 degree inversion pulse across 50ppm, centered on dof2, to invert all aromatic N1/N9 N15 nuclei when dof2=155ppm */

rna_makestN50

/* **180 degree inversion pulse across 100ppm, centered on dof2** */

rna_makestN100

"**90 degree selective one-lobe pulse on H2O (1.2 ppm), < 1% excitation off resonance**"

rna_makeH2Osinc

"**90 degree selective one-lobe pulse on H2O (0.5 ppm), < 1% excitation off resonance**"

rna_makeH2Osinc1

"flopsy8 decoupling pattern will be made only if not already present"

rna_makeflopsy8

"**dipsi2 decoupling pattern will be made only if not already present**"

rna_makedipsi2

"**90 degree square pulse on Ca, null at CO, 118ppm away. C13 frequency at CO. Generates offC13.RF**"

BPmake90Ca_CO

"**180 degree square pulse on Ca, null at CO, 118ppm away. C13 frequency at CO. Generates offC3.RF**"

BPmake180Ca_CO

"**180 degree square pulse on Ca, null at CO, 118ppm away. C13 frequency at Cab. Generates offC4.RF**"

BPmake180Ca

"**90 degree selective sinc one-lobe pulse on CO, first null at Ca. C13 frequency at Ca. Generates offC2.RF**"

BPmake90CO_Ca

"**90 degree selective sinc one-lobe pulse on CO, first null at Ca C13 frequency at Cab. Generates offC5.RF**"

BPmake90CO

"**90 degree selective sinc one-lobe pulse on CO, first null at Ca C13 frequency at CO. Generates offC6.RF**"

BPmake90CO_CO

"180 degree selective sinc one-lobe pulse on Cab.C13 frequency at CO. Generates offC27.RF"

BPmake180Cab_CO

"**180 degree selective sinc one-lobe pulse on CO, first null at Ca. C13 frequency at Cab. Generates offC7.RF**"

BPmake180CO

"**180 degree selective sinc one-lobe pulse on Ca, first null at CO. C13 frequency at CO. Generates offC17.RF**"

BPmake180Ca_COs

"**180 degree selective sinc one-lobe pulse on CO, first null at Ca. C13 frequency at CO. Generates offC8.RF**"

BPmake180CO_CO

"**180 degree selective sinc one-lobe pulse on CO, first null at Ca. C13 frequency at Ca. Generates offC9.RF**"

BPmake180CO_Ca

"**180 degree selective sinc one-lobe pulse on CO, first null at Ca. C13 frequency at 35ppm. Generates offC10.RF**"

BPmake180COa

"**90 degree selective one-lobe pulse on H2O, < 1% excitation off resonance**"

if (wexp<>'BP10')

"wexp=BP10 for autocalibrate process where H2Osinc.RF is made"

then BPmakeH2Osinc endif

BPmakeH2Osinc1

"**dipsi2 decoupling pattern will be made only if not already present**"

BPmakedipsi2

"**180-degree cosine-modulated pulse on NH, leaving H2O unaffected (180's at +4ppm and -4ppm from H2O**"

BPmakecosinemodulate

10.3 The BPmake90CO_Ca macro

The BPmake90CO_Ca macro makes a base waveform which is then convolted to give the frequency shift.

"Carrier on Ca, 90 degree sinc pulse on CO 118ppm away, null at Ca."

"The carrier should be set to 56ppm."

$pwC2 = 80.5*600.0/sfrq “**proper pulse width for this excitation**”

“**make maximum number of 200nsec steps**”

$steps2 = $pwC2*5.0 $steps2 = 2.0*trunc($steps2/2.0) + 1.0

$pwC2 = ($steps2 + 1.0)/5.0 “find actual pulse width that fits”

$rf2 = 1023.0 **“maximum power in waveform”**

BPmakesinc($steps2, $rf2) “makes a sinc pulse with correct # steps”

$stps2 = ''

format($steps2,0,0):$stps2 “make a text piece with correct numbers”

$shape2 = 'sincs' + $stps2 “make a name using number of steps”

$freq2 = dfrq*118.0 “desired off-resonance shift in excitation”

BPconvolute($shape2,'offC2',$pwC2,$freq2) **“make phase-ramped shape”**

BPmoveshape('offC2.RF') “store shape in appropriate shapelib”

shell('rm -f',userdir+'/shapelib/'+$shape2+'.RF'):$dum

write('line3','The SLP pulse "shapelib/offC2.RF" has been created')

10.4 The BPsetwurstparams macro

The BPsetwurstparams macro obtains decoupling channel parameters directly from the waveform specified, depending on the name of the pulse sequence. The macro BPpboxget looks in the header of the shape file to find values and installs them in a specific order. Variables preceded by $ are not set. BPsetwurstparams is used in experiment setup macros.

if ((seqfil='hbcbcgcdceheA') or (seqfil='hbcbcgcdhdA')) then

BPpboxget('wurst40.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf

endif

if ((seqfil='hbcbcacocahaA') or (seqfil='hacahbA') or (seqfil='noesyCA')) then

BPpboxget('wurst80.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf

endif

```
if ((seqfil='gChsqc') or (seqfil='gChsqcP') or (seqfil='gChsqcA')) then
if arom='y' then
BPpboxget('wurst40.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif
if alphaC='y' then
BPpboxget('wurst50.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif
if allC='y' then
BPpboxget('wurst140.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif
if aliph='y' then
BPpboxget('wurst80.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif
endif
if (seqfil='gChmqc') then
if arom='y' then
BPpboxget('wurst40.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif
if aliph='y' then
BPpboxget('wurst80.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif
endif
if ((seqfil='gCfhsqc') or (seqfil='gCfhsqcA') or (seqfil='CNnoesy_interintraA')) then
BPpboxget('wurst140.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif
if (seqfil='CTgChmqc') then
if arom='y' then
BPpboxget('wurst40.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif
if aliph='y' then
BPpboxget('wurst80.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif
if allC='y' then
BPpboxget('wurst140.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif
endif
if ((seqfil='ghca_co') or (seqfil='ghca_co_n'))  then
```

```
BPpboxget('wurst50.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif

if ((seqfil='ghca_coA') or (seqfil='ghca_co_nA'))  then
BPpboxget('wurst50.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif

if (seqfil='ghn_ca') then
BPpboxget('cocbdec.DEC'):Cbdseq,$dmmy,Cbdpwr,$dmmy,Cbdres,Cbdmf
endif

if (seqfil='gChmqcnoesyNhsqc') then
BPpboxget('codec.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif

if (seqfil='gNhmqcnoesyNhsqc') then
BPpboxget('cocadec.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif
if ((seqfil='gnoesyChsqcA') or (seqfil='gnoesyCNhsqcA') or
(seqfil='gChsqcnoesyA') or (seqfil='gNhsqctocsyA') or
(seqfil='gChsqctocsyA') or (seqfil='gtocsyChsqcA')) then
BPpboxget('wurst80.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif

if ((seqfil='gnoesyChsqc') or (seqfil='gnoesyCNhsqc') or (seqfil='gLRCC') or
(seqfil='gChsqcnoesy') or (seqfil='gLRCH') or
(seqfil='gNhsqctocsy') or (seqfil='gNhsqctocsyA') or
(seqfil='gNhsqcnoesy') or (seqfil='gNhsqcnoesyA') or
(seqfil='rna_hmqc_tocsy') or
(seqfil='gChsqctocsy') or (seqfil='gtocsyChsqc')) then
BPpboxget('wurst80.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif

if (seqfil='ghcch_tocsy') or (seqfil='ghcch_tocsySP') or (seqfil='hcch_tocsy') or
(seqfil='hcch_tocsySP') then
BPpboxget('wurst80.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif
```

```
if ((seqfil='hcch_tocsyA') or (seqfil='hcch_cosyA')) then
BPpboxget('wurst80.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif

if ((seqfil='tnnoesy') or (seqfil='SSnoesy') or
(seqfil='tntocsy') or (seqfil='tnroesy') or
(seqfil='qwnoesy') or (seqfil='qwnoesyA') or
(seqfil='tndqcosy') or (seqfil='tnmqcosy')) then
 BPpboxget('wurst140.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif

if ((seqfil='wgnoesy') or (seqfil='zdipsitocsy') or
(seqfil='wroesy') or (seqfil='wnoesy') or
seqfil='qwnoesy') or (seqfil='wgtocsy') or (seqfil='water')) then
BPpboxget('wurst140.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif
if (seqfil='hcch_tocsyP') then
BPpboxget('cctocsy.DEC'):mixpat,pwmix,mixpwr,mixpwrf,mixres,mixdmf
endif

if (seqfil='ghn_coca_cb') then
BPpboxget('CgCO1.RF'):$dmmyshp,pwCgCO,pwCgCO_lvl
endif

if ((seqfil='rna_gChsqc') or (seqfil='rna_gCtrosy')) then
if aromatic='y' then
BPpboxget('wurst40.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
else
if allC='y' then
BPpboxget('wurst110.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
else
BPpboxget('wurst80.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif endif endif

if (seqfil='rna_gCtrosy') then
if (aromatic_C5='y') then
```

```
BPpboxget('wurst80.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif
endif

if ((seqfil='rna_gnoesyChsqc') or (seqfil='rna_gnoesyChsqcA')) then
BPpboxget('wurst140.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif

if (seqfil='rna_hcch_tocsy') then
if AH2H8='y' then
BPpboxget('wurst40.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
else
BPpboxget('wurst80.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif endif

if ((seqfil='rna_gChmqc') or (seqfil='rna_hcch_cosy')) then
if aromatic='y' then
BPpboxget('wurst40.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
else
BPpboxget('wurst80.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif endif

if ((seqfil='rna_Ahnctocsyc') or (seqfil='rna_HCN') or (seqfil='rna_HCP'))  then
BPpboxget('wurst80.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif

if ((seqfil='rna_CUhnccch') or (seqfil='rna_Ghnctocsyc'))  then
BPpboxget('wurst40.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif

if (seqfil='rna_ghcch_tocsy')  then
BPpboxget('wurst80.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf
endif

if ((seqfil='rna_tnnoesy') or (seqfil='rna_wetnoesy') or
(seqfil='rna_wettntocsy') or (seqfil='tnroesy') or
(seqfil='rna_tndqcosy') or (seqfil='rna_wroesy') or
```

(seqfil='tnmqcosy')) then

BPpboxget('wurst110.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf

endif

if ((seqfil='wgnoesy') or (seqfil='rna_WGnoesy') or (seqfil='wnoesy')

or (seqfil='qwnoesy') or (seqfil='wgtocsy') or (seqfil='rna_water'))

then

BPpboxget('wurst110.DEC'):dseq,$dmmy,dpwr,$dmmy,dres,dmf

endif

10.5 Updating the Probefile after a new PW90 is Determined

It is possible to update the probefile after determining a new RF calibration without doing a full update. The BP*updatepw90 macros, (where *=H1,C13,N15 or H2) do an update of all parameters depending on the specified nucleus calibration, including re-creation of waveforms. These macros are detailed below. These macros are run by specific nucleus buttons on the Globals&Probefile page in the Setup folder (see Figure 6).

10.5.1 The BPH1updatepw90 macro

/* BPH1updatepw90: updates proton pulse parameters values in probefile.

Updates probefile with tof,tpwr,compH values in this parameter set. pw90 is updated to pw value and stored in probefile. finepower values in probefile adjusted. ref_pw90 and ref_pwr values in probefile updated*/

"read old H1pw and store in p1"

getparam('pw90','H1'):$pw90_probefile

"ratio of new and old pw"

$ratio=pw/$pw90_probefile

pw=trunc(100*pw)/100

pw90=trunc(100*pw)/100

ref_pw90=trunc(100*pw*compH)/100 ref_pwr=tpwr

"load H1tpwrsf values into $tpwrsf values"

getparam('tpwrsf','H1'):$tpwrsf_probefile

getparam('tpwrsf_u','H1'):$tpwrsf_u_probefile

getparam('tpwrsf_d','H1'):$tpwrsf_d_probefile

getparam('tpwrsf_i','H1'):$tpwrsf_i_probefile

getparam('tpwrsf_t','H1'):$tpwrsf_t_probefile

getparam('tpwrsf_n','H1'):$tpwrsf_n_probefile

```
"Calculate updated H1tpwrsf"
$tpwrsf_probefile=trunc($tpwrsf_probefile*$ratio)
$tpwrsf_u_probefile=trunc($tpwrsf_u_probefile*$ratio)
$tpwrsf_d_probefile=trunc($tpwrsf_d_probefile*$ratio)
$tpwrsf_i_probefile=trunc($tpwrsf_i_probefile*$ratio)
$tpwrsf_t_probefile=trunc($tpwrsf_t_probefile*$ratio)
$tpwrsf_n_probefile=trunc($tpwrsf_n_probefile*$ratio)

getparam('finepwrf','H1'):$finepwrf_probefile

"calculate updated H1finepwrf"
$finepwrf_new=trunc($finepwrf_probefile*$ratio)

$tpwrsf='' format($tpwrsf_probefile,4,1):$tpwrsf
BPsetparams('tpwrsf',$tpwrsf,'H1')
$tpwrsf_u='' format($tpwrsf_u_probefile,4,1):$tpwrsf_u
BPsetparams('tpwrsf_u',$tpwrsf_u,'H1')
$tpwrsf_d='' format($tpwrsf_d_probefile,4,1):$tpwrsf_d
BPsetparams('tpwrsf_d',$tpwrsf_d,'H1')
$tpwrsf_i='' format($tpwrsf_i_probefile,4,1):$tpwrsf_i
BPsetparams('tpwrsf_i',$tpwrsf_i,'H1')
$tpwrsf_t='' format($tpwrsf_t_probefile,4,1):$tpwrsf_t
BPsetparams('tpwrsf_t',$tpwrsf_t,'H1')
$tpwrsf_n='' format($tpwrsf_n_probefile,4,1):$tpwrsf_n
BPsetparams('tpwrsf_n',$tpwrsf_n,'H1')
$finepwrf='' format($finepwrf_new,4,1):$finepwrf
BPsetparams('finepwrf',$finepwrf,'H1')
$ref_pw90='' format(ref_pw90,2,2):$ref_pw90
BPsetparams('ref_pw90',$ref_pw90,'H1')
$ref_pwr='' format(ref_pwr,2,0):$ref_pwr
BPsetparams('ref_pwr',$ref_pwr,'H1')
$pw90='' format(pw90,2,2):$pw90
BPsetparams('pp',$pw90,'H1')
BPsetparams('pw90',$pw90,'H1')
$tpwr='' format(tpwr,2,0):$tpwr
BPsetparams('tpwr',$tpwr,'H1')
BPsetparams('pplvl',$tpwr,'H1')
```

```
$dmf=trunc(1e6/(pw*10*compH)) $dpwr=tpwr-20
$dmfH1='' format($dmf,5,0):$dmfH1
$dpwrH1='' format($dpwr,2,0):$dpwrH1
BPsetparams('dmf',$dmfH1,'H1')
BPsetparams('dpwr',$dpwrH1,'H1')
$tofH2O='' format(tof,4,1):$tofH2O
BPsetparams('tofH2O',$tofH2O,'H1')
$compH='' format(compH,4,3):$compH
BPsetparams('compH',$compH,'H1')
BPsetparams('tpwr_cf',$compH,'H1')
write('line3','H1 Probefile Update Finished')
banner('H1 Probefile Parameters Updated using This pw90')
```

10.5.2 The BPN15updatepw90 macro

```
/* BPN15updatepw90: updates nitrogen pulse parameters values in probefile based on
 pwN/pwNlvl/compN. Updates decoupling parameters in probefile to local values*/
“calculates wurst40N.DEC and wurst80N.DEC for adiabatic decoupling.”

pwN=trunc(10*pwN + 0.5)/10
$dmm2=dmm2
length(dmm2):$length
substr(dmm2,$length,1):$mode  "last character shows type of modulation"

$bw = ''
$wave = ''
$name = ''
$n3=n3
format(BPdpwr2max,2,0):n3
n3=n3+'d'
$attn=n3
format(40.0*dfrq2,9,1):$bw    "set N15 decoupling bandwidth to 40 ppm "
if BPpwrlimits=1 then
$wave = 'WURST40' + $bw + '/2.1ms'
else
$wave = 'WURST2' + $bw + '/2.1ms'
endif
```

```
opx
setwave($wave)
pbox_rst
pboxpar('name', 'wurst40N.DEC')
pboxpar('ref_pwr', pwNlvl)
pboxpar('ref_pw90', pwN*compN)
shell('Pbox')
if BPpwrlimits=1 then
BPpboxget('wurst40N.DEC'):$name,$par,$dpwr,$par,$par,$par
if ($dpwr)>BPdpwr2max then
pboxpar('attn',$attn)
echo($attn)
shell('Pbox')
endif
endif
if $mode='p' then
BPpboxget('wurst40N.DEC'):dseq2,$par,dpwr2,$par,dres2,dmf2
endif
BPmoveshape('wurst40N.DEC')
n3=$n3
 “calculate parameters for waltz, garp modulation”

$pwNdec=1/(4*18*dfrq2)  "pw90 for 18ppm 15N decoupling field "
ln((1e6*$pwNdec)/(compN*pwN)):$ln
$dpwr2=pwNlvl - 20.0*$ln/2.303
$dpwr2=trunc($dpwr2+0.5)
$Dmf2=trunc(1/$pwNdec)

"if BPpwrlimits=1, limit dpwr2 to BPdpwr2max and adjust dmf2 "
if BPpwrlimits=1 then
if BPdpwr2max<$dpwr2 then
ln(10):$ln10
$exp=$ln10/20*($dpwr2-BPdpwr2max)
exp($exp):$factor
$Dmf2=trunc($Dmf2/$factor)
$dpwr2=BPdpwr2max
endif
```

```
endif
dmm2='cc'+$mode  dm2='nny'

if $mode<>'p' then
dpwr2=$dpwr2 dmf2=$Dmf2 dres2=1.0
endif

$compN='' format(compN,4,3):$compN
$pwN='' format(pwN,3,1):$pwN
$dmf2a='' format(dmf2,5,0):$dmf2a
$dpwr2a='' format(dpwr2,5,0):$dpwr2a
$pwNlvl='' format(pwNlvl,2,0):$pwNlvl
$dres2='' format(dres2,5,1):$dres2

BPsetparams('compN',$compN,'N15')
BPsetparams('pw90',$pwN,'N15')
BPsetparams('pwx',$pwN,'N15')
BPsetparams('pwN',$pwN,'N15')
BPsetparams('pwNlvl',$pwNlvl,'N15')
BPsetparams('tpwr',$pwNlvl,'N15')
BPsetparams('pwxlvl',$pwNlvl,'N15')
BPsetparams('pwxlvl_cf',$compN,'N15')
BPsetparams('dpwr2NH',$dpwr2a,'N15')
BPsetparams('dpwr',$dpwr2a,'N15')
BPsetparams('dmm',dmm2,'N15')
BPsetparams('dmm2NH',dmm2,'N15')
BPsetparams('dseq2',dseq2,'N15')
BPsetparams('dres2NH',$dres2,'N15')
BPsetparams('dres',$dres2,'N15')
BPsetparams('dmf2NH',$dmf2a,'N15')
BPsetparams('dmf',$dmf2a,'N15')

“create 80ppm decoupling waveform”
$bw = ''
$wave = ''
$name = ''
$n3=n3
```

```
format(BPdpwr2max,2,0):n3
n3=n3+'d'
$attn=n3
format(80.0*dfrq2,9,1):$bw   "set N15 decoupling bandwidth to 80 ppm "
if BPpwrlimits=1 then
$wave = 'WURST40' + $bw + '/2.1ms'
else
$wave = 'WURST2' + $bw + '/2.1ms'
endif
opx
setwave($wave)
pbox_rst
pboxpar('name', 'wurst80N.DEC')
pboxpar('ref_pwr', pwNlvl)
pboxpar('ref_pw90', pwN*compN)
shell('Pbox')
if BPpwrlimits=1 then
BPpboxget('wurst80N.DEC'):$name,$par,$dpwr,$par,$par,$par
if ($dpwr)>BPdpwr2max then
pboxpar('attn',$attn)
echo($attn)
shell('Pbox')
endif
endif
BPmoveshape('wurst80N.DEC')
n3=$n3
dmm2=$dmm2
if BPplot='off' then BPplot='' endif                    "reset plotter"
banner('N15 Pulse and Decoupling Probefile Parameters Updated ')
```

10.5.3 The BPC13updatepw90 macro

/* BPC13updatepw90: updates carbon pulse parameters values in probefile based on pwNCpwClvl/compCN. Updates decoupling parameters in probefile to local values*/

“re-calculates shapes for adiabatic decoupling.”

```
pwC=trunc(10*pwC + 0.5)/10
```

```
$pwC='' format(pwC,3,1):$pwC
$pwClvl='' format(pwClvl,2,0):$pwClvl

BPsetparams('pw90',$pwC,'C13')
BPsetparams('pwx',$pwC,'C13')
BPsetparams('pwC',$pwC,'C13')
BPsetparams('pwClvl',$pwClvl,'C13')
BPsetparams('tpwr',$pwClvl,'C13')
BPsetparams('pwxlvl',$pwClvl,'C13')

BPsetupshapes ''recreate *.DEC shapes with new pwC''
banner('C13 Pulse Probefile Parameters Updated and Decoupling Shapes Re-created')
```

10.5.4 The BPH2updatepw90 macro

```
/* BPH2updatepw90: updates deuterium pulse parameters values in probefile based on current
dpwr3/dmf3/dof3 . */

$dpwr3D='' format(dpwr3,2,0):$dpwr3D
BPsetparams('dpwr3D',$dpwr3D,'H2')
$dmf3D='' format(dmf3,5,1):$dmf3D
BPsetparams('dmf3D',$dmf3D,'H2')
$dof3D='' format(dof3,5,1):$dof3D
BPsetparams('dof3D',$dof3D,'H2')
```

Chapter 11 AutoCalibration

The above sections describe how a probefile is created and updated. It describes how experiments are setup using macros, and the steps involved along the way. If a user had to manually perform all the RF and gradient calibrations, as well as any small-angle phase corrections necessary for some pulse sequences, the potential for error is large, not to consider the time necessary for the activity. For this reason, several macros are provided to perform automatic RF and gradient calibrations. In addition, any small-angle phase shifts necessary are determined. All results are stored in the probefile and these results are accessed by the experiment setup macros. Since these calibrations and settings are used by the experiment setup macros, a full autocalibration should be done on a doubly-labeled (13C and 15N) protein or RNA. If such a sample is not available, calibrations may be performed on a 13C- or 15N-labeled compound giving a mechanism for obtaining all the RF and gradient calibrations. The full autocalibration method should be used on a doubly-labeled compound before any triple-resonance experiments utilizing specific small-angle phase corrections. If there is any doubt, examine the setup macro (same name as the pulse sequence) for any small-angle phase correction statements.

Autocalibration for a doubly-labeled compound is performed by the BPAutoProteinCal. There are several options available and these are passed to the macro as arguments ($1). The possible options are listed below. For reference, the Calibrations Page is reproduced in Figure 29.

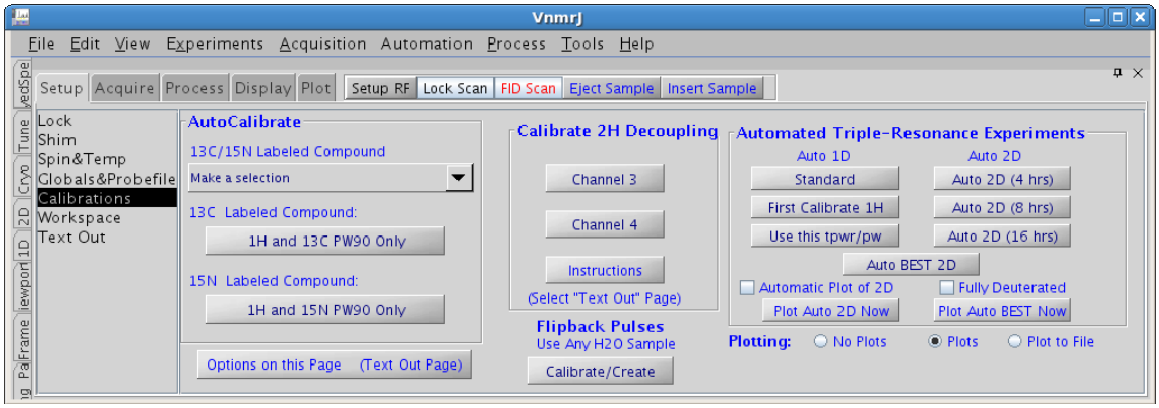


Figure 29 The Calibrations Page (See Section 4.0 for descriptions of options)

11.1 BPAutoProteinCal Options in the 13C/15N Labeled Compound Menu

Option	Argument($1) specified by menu selection
Full calibration (use probefile values)	0
Full calibration (use current tof)	2
Full calibration (use current tof,pw)	3
Fast Option (RF only)	4

Fastest Option (90's and gradients only)	9
1H only with out determining compH	5
N15 only	6
C13 only	7

Full calibration is possible using all starting values obtained from the probefile ($1=0). Sometimes a user might have determined a new value for the H2O frequency and in this case this new value of tof (in the current parameter set would be used ($1=2). Similarly, if the user had also determined a new pw90 for 1H, the $1=3 option permits the calibration to use this information as the starting point.

Full calibration results in all the RF calibrations for 1H, 13C and 15N (pw90's and amplifier compression factors), fine power values for various H2O selective pulses, small-angle phase shifts optimization for these selective pulses, gradient values for coherent transfer gradients in sensitivity-enhanced experiments for 13C and 15N, Bloch-Seigert phase corrections for several sequences involving shaped pulses, 1D first increment spectra for a large number of double- and triple-resonance experiments and comparisons of intrinsic sensitivities of different pulse sequences relative to HSQC for the appropriate nucleus.

Once a full calibration has been performed there is usually no need for small-angle phase corrections and gradient values to be redetermined since these are independent of sample. Therefore, the Full RF (Faster) option can be used to characterize a new sample (perhaps after the spectrometer has been used by another operator). No 1D spectra are collected in this mode. A faster mode is the RF and Gradients (Fastest) option.

If there is only a sample change and probe (1H) retuning, the 1H-only option ($1=5) is a fast way to redetermine the 1H pw90 and update the probefile for all parameters depending on the 1H pw90. Of course, the single-transient method using the "Find PW90" button on the Basic Page is the most convenient method, and fastest.

Options for 13C and 15N ($1=6 and 7) permit recalibration of these nuclei and updating of relevant probefile values.

There are two options for pulse sequence for the 1H and 13C calibrations: gCfhsqc and hcch_tocsy. The gCfhsqc has the virture of being a simple sequence, using Watergate 3919 for water suppression and having no necessary gradient calibration. The hcch_tocsy is a longer sequence, but since the spinlock step is omitted during the calibration it is essentially an INEPT sequence. Both have the 1H and 13C transmitters on-resonance so there is minimal offset effect. The hcch_tocsy more closely represents the multi-pulse sequences used in BioPack and is the default sequence used. The gCfhsqc sequence is useful for first-time installations. No decoupling is used in either calibration steps, so only "hard-pulse" RF events are used. The gCfhsqc option is used if the global parameter BPquick=1. This variable is selected by a check box in the "Globals&Probefile" page of the Setup folder in VnmrJ.

11.1.1 The BPAutoProteinCal macro

"BPAutoProteinCal(<option>)"

" uses either hcch_tocsy (no spinlock or decoupling-essentially INEPT) or gCfhsqc"

$date=''

BPgetdate:$date **"gets current date"**

BPdeuterated=0 **"not used for autocalibrates (is used in 2D)"**

cd(userdir)

```
exists('AutoTripRes','file'):$e                    “backs up existing directory”
if ($e=1) then
exists('BioPack.dir/backups','directory'):$e
if not($e) then
shell('mkdir -p BioPack.dir/backups'):$dum
endif
mv('AutoTripRes','BioPack.dir/backups/AutoTripRes_'+$date)
endif
mkdir('AutoTripRes')
$tof=tof $pw=pw
$vtset=''
shell('showstat > ',curexp+'/stat'):$dummy     “get current temperature”
lookup('file',curexp+'/stat','VT','active:','read'):$vtset,$ret
delete(curexp+'/stat')
if $ret then
format($vtset,0,1):temp
endif
if BPquick=1 then gCfhsqc else hcch_tocsy ncyc=0 endif
if ($#>0) then  "No argument uses probefile values, full calibrate"
if $1=2 then tof=$tof endif
if $1=3 then tof=$tof pw=$pw endif
if $1=4 then r7=1 endif                    "For Fast RF-Only Calibration(stops at BP5d)"
if $1=5 then r7=2 endif                    "For 1H pw90    Calibration only (not compH)"
if $1=7 then r7=7 N15refoc='n' endif       "Calibration (13C-labelled cmpd)"
if $1=6 then                               "Calibration (15N-labelled cmpd)”
gNfhsqc dm2='nnn' r7=6
endif
endif
if ((BPplot='plot')) or (BPplot='on')) then
if ($#>0) then  "Option calibrate"          “Select printout”
if $1=4 then                               "Fast Option"
printon man('proteinpreface4') printoff
else                                       "other option"
if ($1=5 or $1=15) then                    "1H pw90-only Option"
printon man('proteinpreface5') printoff
elseif ($1=6 or $1=16) then
printon man('proteinpreface6') printoff
```

```
elseif ($1=7 or $1=17) then
printon man('proteinpreface7') printoff
elseif ($1=9 or $1=19) then
printon man('proteinpreface9') printoff
else
printon man('proteinpreface') printoff
endif
endif
else   "Full calibration"
printon man('proteinpreface') printoff
endif
endif

if (seqfil='gNfhsqc') then
wnt='wft select(1) setref sp=6p wp=4p vsadj(50) dssh'
else
dm='nnn'
wnt='wft select(1) setref sp=-1p wp=4p vsadj(50) dssh'
endif

ni=0 phase=1
pw=trunc(10*pw + 0.5)/10 "ensures pw is multiple of 100nsec"
setlimit('pw',1000,0,0.1)                    "Limit parameter entry"
setlimit('pwC',1000,0,0.1)
nt=2 array('pw',15,pw-0.35*pw,0.05*pw)
f ss=4 ssfilter=500 ssntaps=200 ssorder='n' full av
vp=0 cdc f fn=4*np
wexp='BP1'
spin='n'
au
```

The full AutoCalibration process is performed at least once immediately after installation and a probefile update. This ensures that all necessary calibrations are performed. It does require a doubly-labelled sample. If one is not available, the 13C or 15N options above should be used. No triple-resonance experiments involving small-angle phase corrections should be done until this full calibration is performed.

11.2 2H AutoCalibration

Calibration via Direct 2H Detection

2H decoupling during a pulse sequence is often desirable, particularly for large deuterated proteins. The hardware necessary to do this is typically a broadband channel 4. In some cases channel 3 might be used. There are macros provided for each of these hardware cases. In each case, the 2H present in the solvent (typically D2O) provides a strong 2H signal via direct observe using the decoupling channel. Here, a single pulse is delivered via the decoupling channel using the same RF path used for decoupling. The signal is detected using the same RF path used for 2H observe for gradient shimming. No cable change is necessary for the Varian NMR System. Fortunately, the double-tuned 1H/2H coil is so heavily weighted toward 1H that the 2H calibration is insensitive to sample so the calibration need not be done as often as for 1H.

In VnmrJ the user selects the “Calibrations” page within the Setup folder. The “Instructions” button is displayed within the 2H AutoCalibration group.

The process is initiated by using the proper button for the RF channel to be used for decoupling (use the relevant button in the “Calibrations” page within the Setup folder.

The autocalibration consists of an experiment to find the 2H signal, and then for a fixed pulse width, the power level is varied. After the signal goes through a maximum the experiment is automatically stopped and a pulse width array spanning the expected pw360 is set up at the power giving the maximum signal. The null spectrum in this new experiment gives pw360. Then a macro updates the probefile with dpwr3 and dmf3 based on an appropriate decoupling bandwidth. In addition, the dof3 probefile value is updated and set to 2.5ppm in the 2H spectum. Henceforth, any experiment setup macro for an experiment potentially involving 2H decoupling obtains calibrated values for these parameters so 2H decoupling is obtainable by just setting dm3=’nyn’ (see pulse sequences for details. dm3 is used only as a flag and is not using the normal “status” control. Most sequences do not have 2H decoupling coded, however the Channels page does show a proper calibration for decoupling).

Calibration via Indirect 2H Detection using 13C Observe

A pulse sequence (*ddec_pwxcal.c) is supplied that can be used on a C6D6 or CDCl3 sample. Cable the probe for C13 observe and connect the channel 4 output to the 2H Decoupling Accessory (diplexer box). Connect the lock cable normally connected to the probe to the same box, and connect the box to the probe at the lock BNC. Use the 2H reject filter and low-pass filter at the BB preamp.

Use the dropdown menu (VnmrJ) for parameters. Adjust the C13 pw to the value of a 90 degree pulse and any other appropriate parameters. The C6D6 or CDCl3 triplet exhibits a nulling with increasing pw(2H), but the outer lines null at a longer time than the inner lines. Use the outer lines to find a null and this occurs at pw90 (2H). Use this value to determine dmf3 (dmf3=1e6/pw90) at the power dpwr3.

While it may seem that this method is more suitable (since the purpose of 2H decoupling is to decouple 2H from 13C) the 2H observe method is more reliable and far easier to use. It has the advantage of calibrating on the protein sample itself, rather than using an organic solvent.

Control of 2H Decoupling in Pulse Sequences

Deuterium decoupling is optional for spectrometers having 4 channels and may be applied in triple-resonance experiments while CaCb magnetization is transverse, and during any 1H shift evolution, for many BioPack sequences (ghn_co which does not generate transverse CaCb magnetization).

Note that 2H decoupling is done via dm3='nyn'. No status(B) statement is present in the pulse sequences but dm3 is just used as a flag to call the setstatus statement. All the normal decoupling parameters such as dpwr3, dof3, dmm3, dmf3 etc. are active, however, and should be set properly. The parameter "ampmode" should be set to 'dddp' when 2H decoupling is used with a BB 4th channel. This parameter is present and is set by the macro BPsetampmode for all parameter sets having 2H decoupling as an option.

The 2H magnetization is preserved in all sequences except hcch_tocsy and ghcch_tocsy. In the latter cases disturbance of the lock signal may be observed. In sequences where preservation of 2H magnetization is achieved, trim gradients around refocusing pulses are switched off during the 2H decoupling, so additional artifacts are possible.

Testing 2H Decoupling

The parameter set *ddec_s2pul.par uses the pulse sequence *ddec_s2pul.c (s2pul.c with amplifier unblanking and lock sample/hold for channel 4's amp). You can verify the proper dof3 value by adjusting the dof3 array values so that the results are symmetric with respect to dof3. This means that the center value of dof3 corresponds to ~7ppm in the 2H spectrum(shift of CDCl3 or C6D6). The dpwr3 and corresponding dmf3 should be large enough to cover +4ppm experiments, with dof3 set at 3ppm in the 2H spectrum. (It is not necessary to decouple over the full 10ppm range since 2H decoupling is usually used to decouple only aliphatic deuterons). Once you have set the 2H calibrations you can then check the decoupling efficiency using your values of dpwr3, dmf3 and dof3.

If the 2H channel on the probe has been retuned, or if calibrations are done manually, the probefile should be updated using the BPbiopack2 or BPH2updatepw90 macros (via buttons) to update the channel 4 parameters. No change in parameters should be necessary from sample to sample.

2H broadband decoupling is handled internally within the pulse sequences by the "setstatus" psg command, not by the usual "status" approach. Thus, "dps" and "pps" show only status A and C. The dm3 parameter is only used to determine that decoupling is desired during relevant portions of the pulse sequence and is requested by setting dm3='nyn'. Use of the normal status approach will result in other channels being reset to their status values, since all channels are set at the same time by any change in status on any channel. Hence, C13 decoupling that is initiated by waveform psg commands (decprgon) at an earlier time would be turned off since dm='n' when there would be a normal status change on any other channel.

11.3 The BPcheck macro

Power Limits and the ampmode parameter

The BPpwrlimits parameter (a global variable) has possible values of 0 and 1. If the value is 1 several macros will act accordingly, checking for possible power limit violations. These can arise if a decoupling waveform generated that requires a power in excess of the limit or if a power level is explicitly set. The power limits are not encoded into system-level software or pulse sequences and are only checked when the probefile is updated or an experiment setup macro is run. Each of the first three channels has a global parameter: BPtpwrmax, BPdpwrmax and BPdpwr2max and these are displayed in the “Globals and Probefile” page within the VnmrJ Setup folder. The first macro, BPcheck, is called when an experiment setup macro is run (at the end of BPrtppar).

" BPcheck "

"This is a macro that will run before every BPsvp"

"and after every BPrtppar"

```
if BPpwrlimits=1 then                    "Power limits are desired"
if (tpwr>BPtpwrmax) then                 "only for case of too much power"
ln(10):$ln10    “this section figures out new pw90 for lower power”
$exp=$ln10/20*(tpwr-BPtpwrmax)
exp($exp):$factor
pw90=pw90*$factor
tpwr=BPtpwrmax              “tpwr is now set to user-defined power limit”
endif
if (dpwr>BPdpwrmax) then                 "only for case of too much power"
if ((dmm<>'ccp')or(dmm<>'cccp')or(dmm<>'cpcp'))
"only for non-waveform modulation (waveform operation handled by
BPsetupshapes"
then
ln(10):$ln10
$exp=$ln10/20*(dpwr-BPdpwrmax)
exp($exp):$factor
dmf=dmf/$factor
dpwr=BPdpwrmax
endif
endif
if dpwr2>BPdpwr2max then          "only for case of too much power"
if ((dmm2<>'ccp')or(dmm2<>'cccp')or(dmm2<>'cpcp'))
"only for non-waveform modulation"
then
ln(10):$ln10
$exp=$ln10/20*(dpwr2-BPdpwr2max)
exp($exp):$factor
dmf2=dmf2/$factor
dpwr2=BPdpwr2max
endif
endif
endif
```

The macros BPcheckdpwr and BPcheckdpwr2 perform similar operations and are used within experiment setup macros for different decoupling conditions.

Amplifiers can operate in either “pulsed” (blanked) mode or “cw” (unblanked) mode. The ampmode parameter, if existing, allows overriding of default conditions. Normally this is unnecessary for 1H, 13C and 15N, but 2H must be treated differently because of potential interference with lock performance. Normally the ampmode parameter does not exist in a parameter set. If must be created and set properly for 2H decoupling. The BPsetampmode macro is used for this purpose. It is called in the experiment setup macros for those sequences having programmed 2H decoupling. In the VNMRS system, amplifiers are automatically blanked if unused.

11.4 The BPsetampmode macro

```
if (numrfch>3) then                 “only relevant for 4 channels”
exists('ampmode','parameter'):$e
if $e=0 then create('ampmode','string') endif
substr(rftype,4,1):$ddec            “figure out type of channel”
if $ddec='l' then                   "Channel 4 is lock/decoupler"
dn3='H2'                            "Channel 4 is set for lock normally"
ampmode='dddd'                      “use default mode for lock/decoupler”
else
dn3=''                              "Disable Channel 4 except when needed"
ampmode='dddp'                      “use pulsed mode for normal channel”
endif
else
exists('ampmode','parameter'):$e          “ampmode not needed”
if $e=1 then destroy('ampmode') endif
endif
exists('reffrq','parameter'):$e
if ($e=0) then create('reffrq','real') endif
reffrq=sfrq
```

A broadband channel needs to have the amplifier in the “pulsed” or “blanked” mode to prevent noise from being picked up in the lock preamp. The name “pulsed” does not mean that only pulses can be applied, it just refers to the blanking mode.

If a “normal” broadband channel is used for 2H decoupling, the user must set dn3=’H2’ and dm3=’nyn’ after running the experiment setup macro. Use “dps” to verify activity on channel 4. If the 2H AutoCalibration was performed previously, the dpwr/dmf/dres parameters will be automatically set, leaving only dn3 and dm3 to be set. If the sequence does not include hard-coded 2H decoupling as an option, the dm3 parameter follows the normal “status” control and use of 2H decoupling may adversely affect the lock.

11.5 Understanding Automated Operations

The above sections describe many of the important macros used in seting up of experiments, their execution and the use of the probefile. Autocalibration, Auto1D, Auto2D and sequence-specific automated operations all use the same basic construct: the experiment is begun with the “au” command. This command is an alias of “go” but will execute any text string present in the wexp, wnt, wbs and werr parameters. These parameters are visible in the “Future Actions” page within the Acquire folder.

All automated experiments have setup macros that set the value of wexp. Often, wnt is set to show updated spectral displays during the acquisition of an array. Experiments can be halted during an acquisition and another Experiment joined without any problem, as long as the the original experiment is restarted by “au” (the wexp/wnt values are still the same). If there is a problem with something failing it can be diagnosed by determining when the failure occurred. If the wexp macro fails, the data is still present and a new “wft dssh” will show the data. Since the data is present the failure must be within the execution of the macro specified by wexp. Sometimes, just looking at the data manually will reveal the source of the problem. If the spectral intensities should rise and fall as a function of the arrayed parameter, but actually just rise or fall, the wexp may fail or determine an incorrect value within its execution. The proper strategy is to examine the failed data set, determine the macro to be executed by wexp and then look at that macro. Examine its operation step-by-step. Sometimes parts of the macro will execute and evidence of this gives a clue to the failure point. Error messages are often given to help in the diagnosis.

Autocalibration operations usually generate plots containing parameter tables. These can be examined for the values of wexp so that the operative macro is known. If a failure occurs, examine that last plotted spectrum. Look for the value of wexp and read that macro. It will have a line containing BPsvf which is the macro saving the data. This gives the name of the fid that then can be recalled. The data can then be processed manually. If a value of “wnt” is specified in the recalled data, the command “{wnt}” will execute the contents of wnt. If the data look normal, execute the command “{wexp}” which will execute the contents of wexp. This should allow you to view the error condition and take appropriate steps.

Automatic plotting can be disabled by setting the global parameter BPplot to ‘off’.

Plots may be also made directly to a file (BPplot=’file’), as described in the BPpage macro below.

11.5.1 The BPpage macro

```
" BPpage - utility macro for plotting / saving plots to file(s) "
exists('BPplot','parameter','global'):$eplot
if not($eplot) then
page
else
if BPplot='off' then
page('clear')
elseif BPplot='file' then
$plotdir=''
exists('BPplotdir','parameter','global'):$eplotdir
if $eplotdir then
```

```
$ch1=''
substr(BPplotdir,1,1):$ch1
if $ch1='/' then
plotdir=BPplotdir
elseif BPplotdir<>'' then
shell('dirname',userdir):$plotdir
$plotdir=$plotdir+'/'+BPplotdir
endif
endif
if $plotdir='' then
shell('dirname',userdir):$plotdir
$plotdir=$plotdir+'/BPplots'
endif
exists($plotdir,'directory'):$e
if not($e) then
shell('mkdir -p',$plotdir):$dummy
endif
exists($plotdir,'file','rwx'):$e
if not($e) then
write('error','Unable to create plot dir %s',$plotdir)
shell('dirname',userdir):$plotdir
$plotdir=$plotdir+'/BPplots'
shell('mkdir -p',$plotdir):$dummy
write('line3','Saving BioPack plots in %s',$plotdir)
endif
$numfile=$plotdir+'/BPplotnum'
$plotnumstr='' $plotnum=0
exists($numfile,'file'):$e
if $e then
shell('cat',$numfile):$plotnumstr
format($plotnumstr,'isreal'):$r
if not($r) then
$plotnumstr=''
else
format($plotnumstr,'lower'):$plotnum
$plotnum=$plotnum+1
format($plotnum,'04',0):$plotnumstr
```

```
length($plotnumstr):$len
while $len<4 do
$plotnumstr='0'+$plotnumstr
$len=$len+1
endwhile
shell('rm -f',$numfile):$dum
write('file',$numfile,'%d',$plotnum)
endif
endif
if $plotnumstr='' then
$plotnum=1
$plotnumstr='0001'
shell('echo 1 >',$numfile):$dummy
endif
" determine plot output language "
$pltt=''
plotinfo(plotter):$pltt
lookup('file','/vnmr/devicetable','seek','Example','readline',2)
$devtype='' $res=1
while ($devtype<>$pltt) and ($res=1) do
lookup('seek','PrinterType','read'):$devtype,$res
endwhile
$ext=''
if $devtype=$pltt then
$raster=''
lookup('seek','raster','read'):$raster
if $raster='0' then
ext='.pgl'
elseif ($raster='3')or($raster='4') then
$ext='.ps'
endif
endif

" For now we use an automatic name template '%date%_%%_%seqfil%' - "
" we may add parameter / name template control later "
$plotfile=''
shell('date "+%Y-%m-%d"'):$plotfile
```

```
$plotfile=$plotfile+'_'+$plotnumstr+'_'+seqfil+$ext
$plotfile=$plotdir+'/'+$plotfile
page($plotfile)
write('line3','Plot written  to file %s',$plotfile)
else
page
endif
endif
```

11.6 Experiments Performed During AutoCalibration

The above sections describe in general the experiment setup philosophy and important macros used to manage the probefile. In this section we will cover the actual experiments used for AutoCalibration.

Since a doubly-labeled compound (typ. ~1mM in 90% H_2O) will be used, pulse sequences are selected that produce single-transient spectra that do not need to be phase-cycled. This means that, apart from the residual water signal, larger signal intensity indicates a more suitable value for a parameter when the parameter is arrayed. This will be the case for all experiments used in this process. Data are stored in the AutoTripRes directory in the user's vnmrsys (any existing directory is automatically moved to the BioPack.dir/backups directory). Each experiment is stored for future examination. Data are processed and plotted, if specified, as part of the AutoCalibration.

11.6.1 1H, 13C and 15N RF Calibrations

The first experiment set up by default is hcch_tocsy. While normally this involves a 13C spinlock, this is disabled and 13C decoupling is turned off. Therefore, the experiment is essentially a 13C HSQC. This pulse sequence is used because the 13C transmitter is set in the aliphatic region and therefore resonance offset effects are small (calibrating 13C pulses that are off-resonance will lead to errors, particularly at high field or when pwC values are >10-15usec). The 1H resonance offset effect is small enough to be ignored for pw<10usec. An alternative sequence, gCfhsqc (a 3919 watergate 13C HSQC), is used if BPquick=1 (selected by a checkbox in the "Globals&Probefile" page in the Setup folder.

The BPAutoProteinCal macro sets up the experiment using the calibrations and settings found in the active probefile and then sets up a pw (1H) array of 15 values centered around the probefile pw value. It then sets wexp='BP1' and begins the experiment with the "au" command. At the completion of the experiment, the BP1 macro executes and the data plotted (see Figure 30). The macro measures the peak height of the tallest peak upfield from the water and uses this pw to set up a new array with smaller step sizes. After this new experiment runs, the BP2 macro does the same analysis and stores the value in the pw90 parameter. The power is then dropped 12dB and a new 90-degree pulse is determined (BP2a and BP2b) in the same manner (see Figure 31). The ratio of this new pw90 and the one at higher power permits calculation of compH, the amplifier compression factor for 1H.

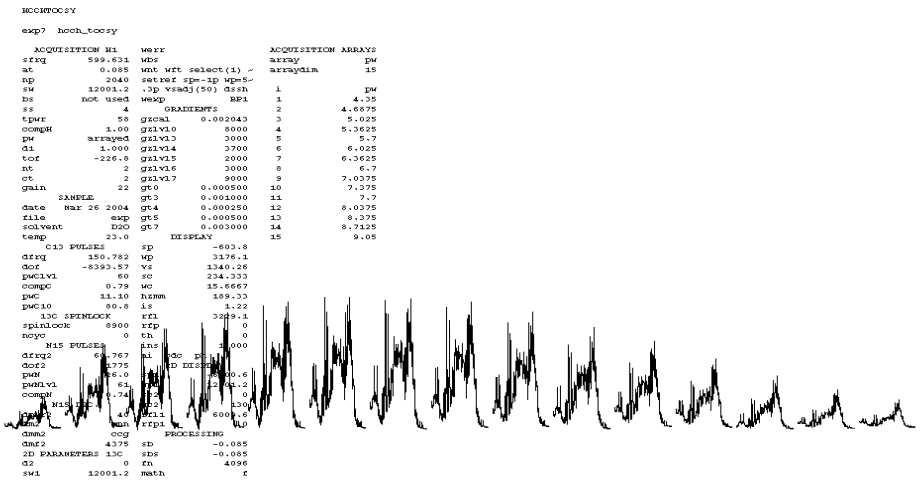


Figure 30 Automatic 1H pw90 Calibration

This factor should be 1.00 under ideal conditions. However, if the tpwr value is near the top of the range, depending on the amplifier, there may be some “compression” of the amplitude, leading to a longer pw90 at the higher power than would be calculated (12dB should mean a factor of 4 in pulse width). For example, if the pw90 at higher power was 6usec and at lower power it was 20, the compression factor would be 20/(4*6), or 0.83. A perfect amplifier would have pw90 at high power be 5usec (the lower power is usually a safer and more accurate power for calibration purposes). For any channel, once compression factors are determined they do not change as a function of sample if the same power values are used.

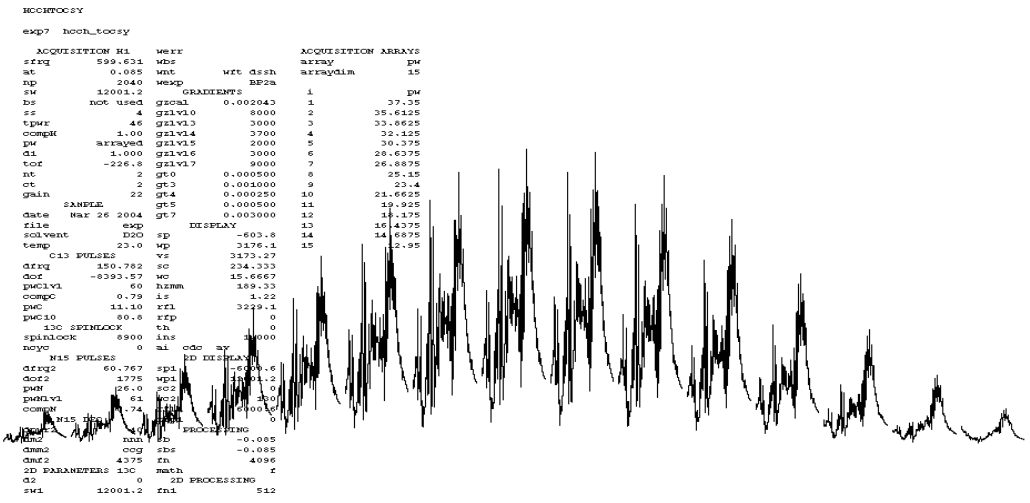


Figure 31 Automatic 1H pw90 Calibration at Lower Power

Following the above experiments, pwC (the 13C pw90) is arrayed around the probefile value over a large range (for example, from 6-28usec). A usuable probe will have pwC fall within this range, and a signal maximum will occur near the correct pwC value. Again, this new value is used as the center of a new pwC array with smaller step size and the best value in this new experiment is “memorized” and stored in the parameter set (macros BP3 and BP4, see Figure 32).

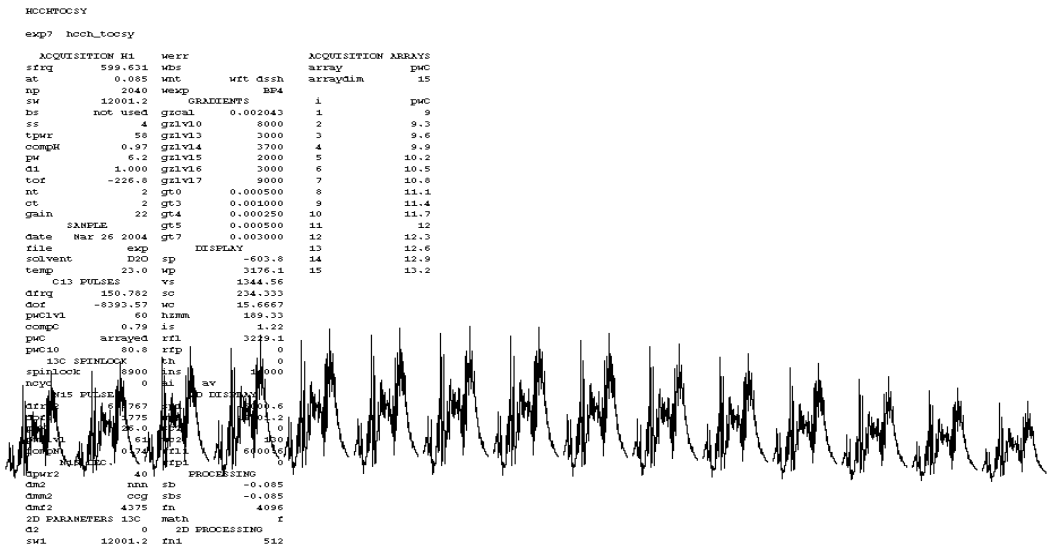


Figure 32 .AutoCalibration of 13C pw90

Some pulse sequences involving simultaneous 13C and 15N pulses automatically drop powers on these channels 3dB for probe protection. Therefore, pwC is then arrayed at this new power, a maximum found, and a new array with smaller step size performed (macros BP4aa and BP4aaa). An estimate of the 13C amplifier compression factor, compC , is also determined.

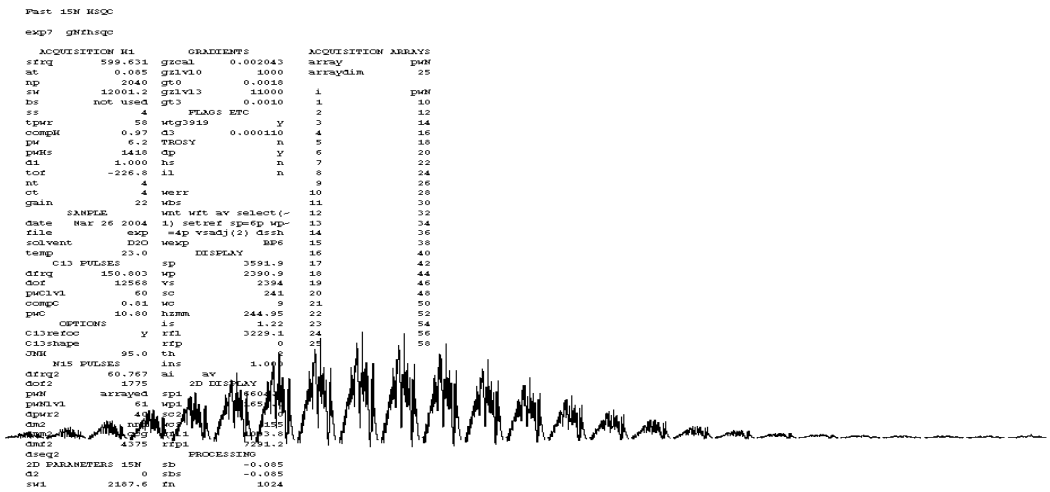


Figure 33 AutoCalibration of 15N pw90

At this point the gNfhsqc experiment is set up and a pwN array set up around the probefile value for pwN. This pulse sequence is used because it only depends on pw and pwN to give signal and does not depend on a gradient calibration. It uses watergate 3919 which uses “hard” pulses so no additional calibration is necessary. Macros BP6 and BP6a are used to determine pwN and the probefile pwNlvl value (Figure 35), and also at a power 6dB lower in order to calculate the compression factor for the 15N amplifier, compN. This value of compN is used for accurate calculation of decoupling parameters for 15N.

11.6.2 Selective Pulses on H_2O

Many experiments use selective 90-degree pulses on the H2O. These are used to preserve water magnetization along the +Z-axis so that exchangeable NH protons are not partially saturated. No presaturation is used typically so water magnetization is aligned along +Z at the end of the relaxation delay. While pulses within the pulse sequence do affect the water magnetization, in most cases (a “water-friendly” sequence) the water magnetization is restored to +Z as soon as possible, and certainly before acquisition. This not only preserves the magnetization but minimizes the detected water signal.

Different pulse sequences use different selective pulses, because the type of phase control used is different (examine the pulse sequence code or use “dps” to see how the selective pulses are used). For this reason, five different selective pulse types are used, and five different shaped pulse files must be created. This is done in the AutoCalibrate process. The pulse sequence (flipback_cal) has different options for different cases (governed by the satmode parameter). Each shaped pulse can be phase-corrected within the sequence and fine power adjustment is also possible. A satmode value of “i”,”t” or “n” produces a soft pulse immediately following an 1H-15N INEPT transfer (often used in the ghn* and gNhsqc sequences) followed by acquisition. A satmode value of “u” produces a “hard” pw90 followed by a “flip-up” selective pulse. A satmode value of “d” produces a “flip-down” selective pulse followed by a “hard” pw90. In all cases the water signal is monitored. Minimum water signal indicates the most +Z H2O magnetization. The pulses are listed in in the following section:

Selective Pulses on H_2O in the "flipback_cal" Sequence

Satmode	Nature of the Pulse	Shape Created
i	following an INEPT	H2Osinc_i.RF
t	in ghn* TROSY's, gNhsqc	H2Osinc_t.RF
u	following an hard 90	H_2Osinc_u.RF
d	prior to a hard 90	H2Osinc_d
n	in gNhsqc TROSY	H2Osinc_n.RF

The macro BP6a sets up the flipback_cal experiments as a triple array: satmode, phincr and tpwrsf, and uses the H2Osinc.RF shape which is a "standard" shape already created by the BPcal macro in the probefile update. This pulse has no phase correction within the shape. For each value of satmode (pulse sequence type), the fine power is varied around the value of 2000 (the coarse attenuator value is calculated within the pulse sequence from tpwr/pw/compH). The phincr phase correction parameter is set to a constant value.

At the end of the experiment the BP4a macro determines the optimal fine powers by looking for the minimum water signals (Figure 33).It then sets up a new array for each type of selective pulse, using the just-determined fine powers for each, but varying the phincr parameter from +60 to –60 degrees (Figure 34). After this data set is acquired, the BP4a1 macro determines the best phincr values for each shape and calls Pbox to create new versions of the shapes (listed in section that incorporate the phase shift into the pulse shape itself. This means that the shapes can be used in any sequence without needing small-angle phase correction statements.

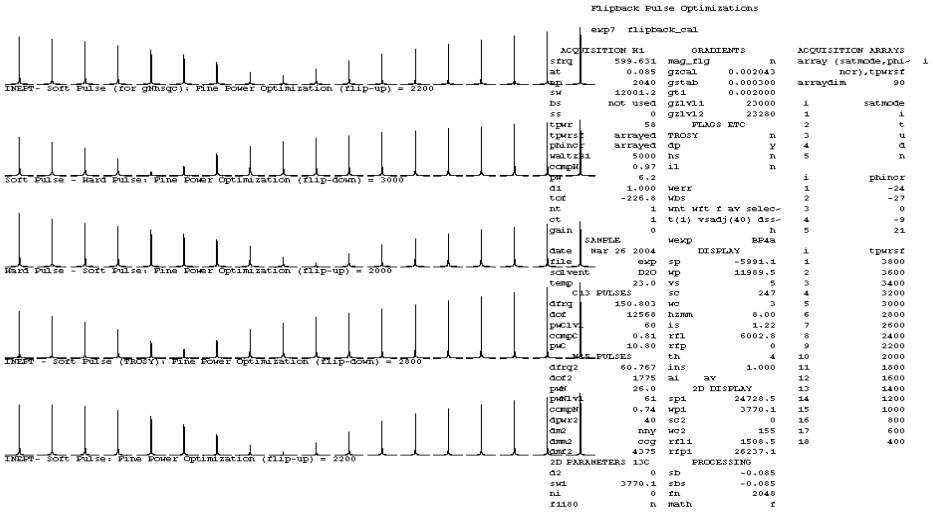


Figure 34 .Fine-Power Optimization of H2O-selective Pulses.

As a final optimization, the fine power is re-arrayed for each of the newly-created shape files. At the end of this experiment the BP4b2 macro determines the best fine power for each of the shapes and updates the probefile with these values.

At low field and/or with a low-Q probe, these power values will be very similar because it is radiation damping that is responsible for the variation. Radiation damping will tend to restore the water magnetization to +Z and can operate in milliseconds at very high field. Use of a high-Q probe also enhances this effect. Thus, it takes more power to do a “flip-down” pulse than predicted. Conversely, it takes less power to do a “flip-up” pulse. Therefore, the fine power optimization can produce values differing by 50% and this calibration is very important in these cases (data in Figures are from 600 MHz HCN Cold Probe).

Separate flipback pulse optimization is also available. A panel button on the “Calibrations” page in the “Setup” folder can run just this section, calibrating the flipback pulses and creating shapes, even on an unlabeled sample (in H_2O).

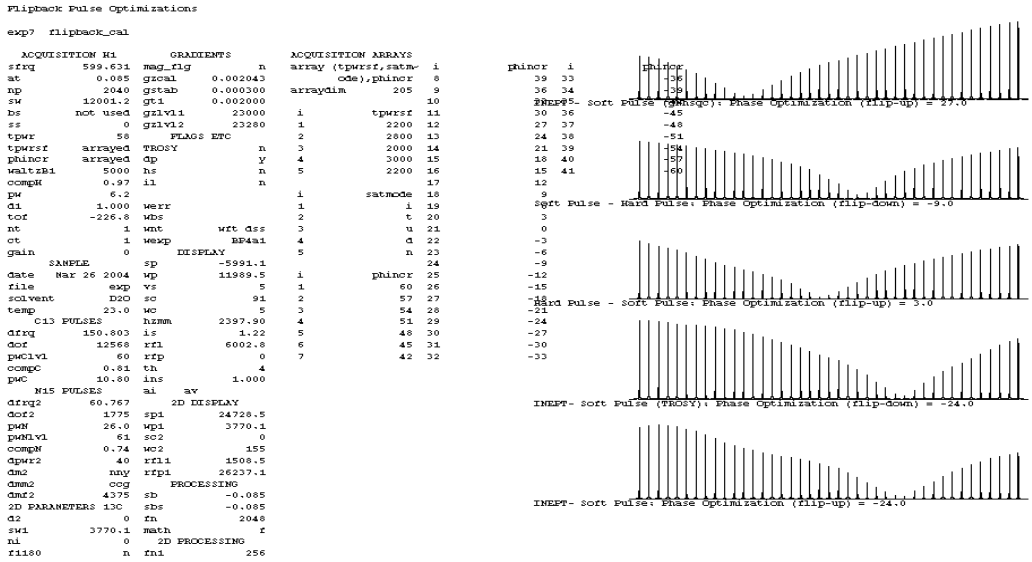


Figure 35 Optimization of Phase Correction for H2O-selective Pulses

11.6.3 NH Coherence Transfer Gradients

Many experiments use gradients in lieu of phase-cycling to select specific magnetization. The experiments that detect NH protons can use 1H and 15N pulses along with a gradient pair having amplitude/time ratios set to refocus NH magnetization and diphase all others.

The next experiment is set up by the BP4b2 macro and uses the ghn_co pulse sequence. All information necessary to do this experiment has now been determined, except for the gradient amplitude ratio. So a gzlvl2 array is set up around the probefile value and the BP5 macro analyzes the acquired data to determine the optimal value of gzlvl2 (see Figure 36 Optimization of NH Coherence Transfer Gradient.. It is refined in a new array with smaller steps and the BP5a macro obtains the best gzlvl2 value. Following this, a compC array is set up around the probefile value. All 13C pulses in this pulse sequence have compC involved in determining their amplitudes, so varying compC directly produces a simple means for optimizing this parameter (macros BP5c and BP5d).

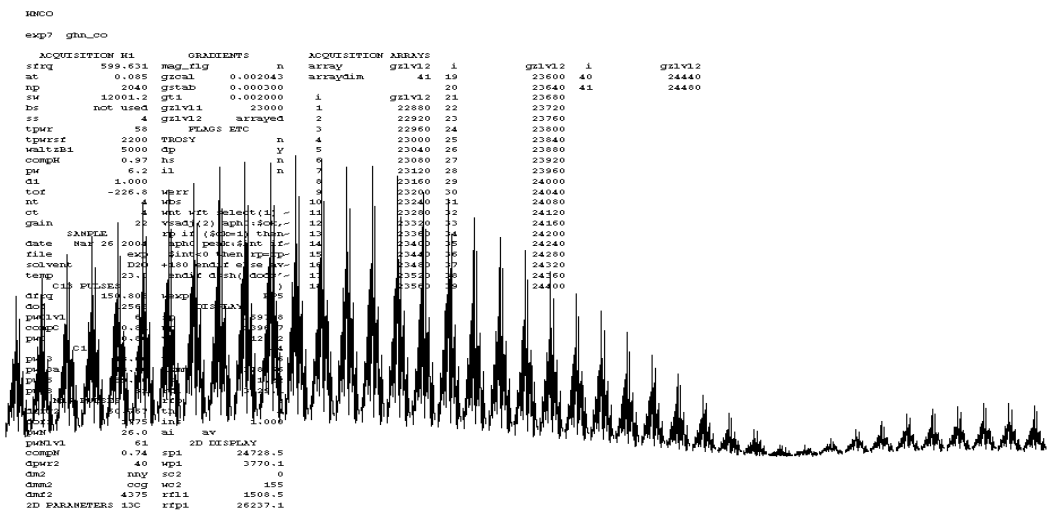


Figure 36 Optimization of NH Coherence Transfer Gradient.

11.6.4 HSQC's

Following the above, a gNhsqc experiment is set up, now using the gzlvl2 value determined above (since the last part of the ghn_co pulse sequence is identical to that of the gNhsqc sequence), and a double array of d1 and nt is set. Three spectra are obtained for d1=.4 nt=16, d1=1 nt=8 and d1=2.1 nt=4. Each experiment takes the same time so the BP7a macro can analyze for best s/n to determine the best d1 value to use in subsequent experiments.

The gNhsqc pulse sequence has an option for NH2-only, so the BP7a macro sets up NH2only='y','n' and the resulting data is processed by the BP8 macro, showing NH2-only data as well as NH /NH2 1D data.

At this point the BP8 macro sets up the gChsqc experiment for the option SE='y' (sensitivity enhanced). This option uses a gradient pair for refocusing CH magnetization and this gradient level is arrayed. The resulting spectra are analyzed by the BP8a macro for the best gzlvl2 value and the probefile updated. (This same optimization is later done using the ghcch_tocsy pulse sequence). Then, six 1D spectra are taken for the cases of detecting protons attached to "all carbons", "aliphatic carbons only", "aliphatic CH2's", "alpha carbons only", "alpha carbon CH2's" and "aromatic carbons" (Figure 37).

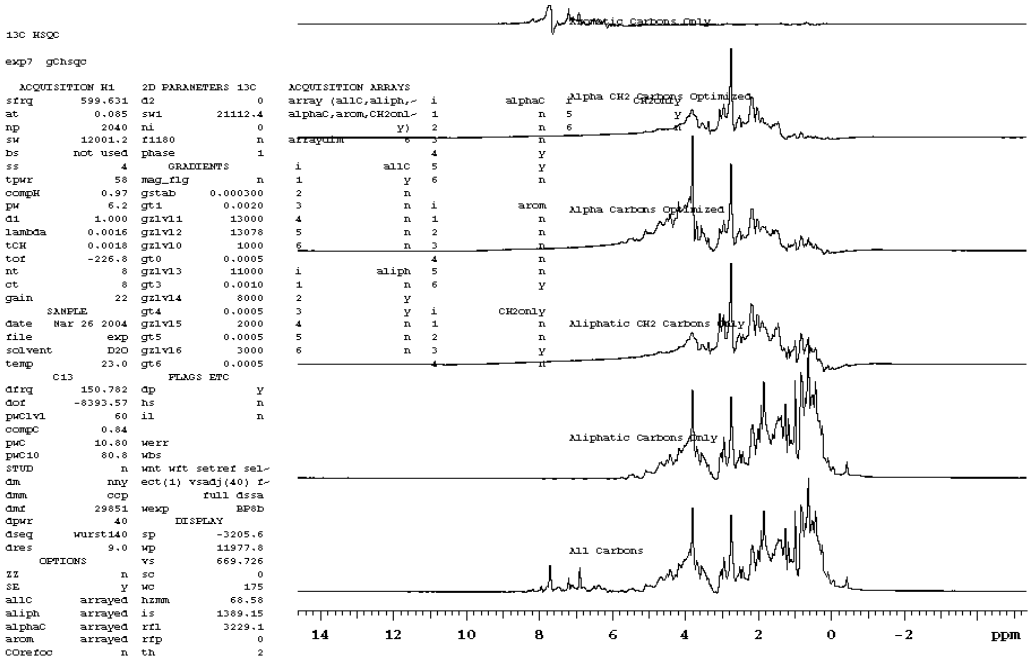


Figure 37 Edited 13C gChsqc Spectra for Carbon Type

11.6.5 Probefile Update

Following the above calibrations, the ghn_co experiment is set up with the new calibrations and then the macro BPbiopack2 is executed (see above). The probefile is fully updated and all waveforms are recalculated using the new pulse calibrations. All decoupling power levels etc. are updated at the same time, for both 13C and 15N.

11.6.6 First Increment Spectra from Triple-Resonance Experiments

The next part of the process is to collect a variety of first increment (ni=1 ni2=1) spectra under identical conditions (gain, sw, nt, at, ….). The spectra are plotted with parameter sets, pulse sequences and scales (for example, see Figure 38).These data provide the input for the side-by-side comparison plotted at the end of the process.

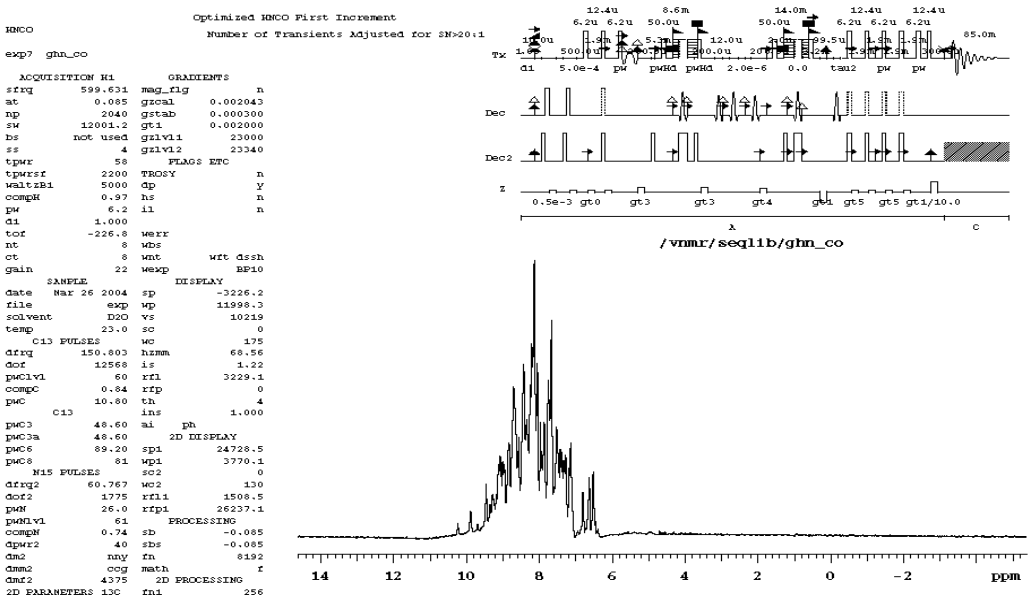


Figure 38 First Increment HNCO Data

11.6.7 Small-Angle Phase Shifts

Several sequences use small-angle phase shifts prior to certain pulses on 13C. This is to accommodate a “Bloch-Seigert” phase shift induced by the use of band-selective pulses. This correction may be avoided by using pairs of band-selective pulses, but this adds complexity and length to a pulse sequence. The amount of correction is determined by the frequency (magnet) and is, to first-order, taken care of in the pulse sequnce code. However, a user-determined shift is provided, and this is calibrated by varying the phase shift to obtain a maximum signal for ni=1 or ni2=1 or, alternatively, a minimum signal, followed by subtraction of 90 degrees. After the phase shift is determined the value is stored in the probefile. Any experiment using these values has part of the setup macro accessing the probefile value.

11.6.8 Experiment “Receptivity” Plots

The final plots are side-by-side comparisons of intrinsic pulse sequence “sensitivity” for the sample used. The N15 HSQC data is scaled to 150mm and the resulting vertical scale (vs) is used on subsequent NH-detected plots (Figure 39). Since all spectra were obtained using the same gain and nt values, this plot permits the user to assess the relative sensitivities. These vary because of losses from relaxation (T2) and are most severe for those sequences that have the desired magnetization spending more time in the backbone carbon framework. A similar plot of CH-detected data is made using the 13C HSQC as a base (Figure 40).

The final printout is a report showing the probefile and stored calibrations and settings (Figure 41).

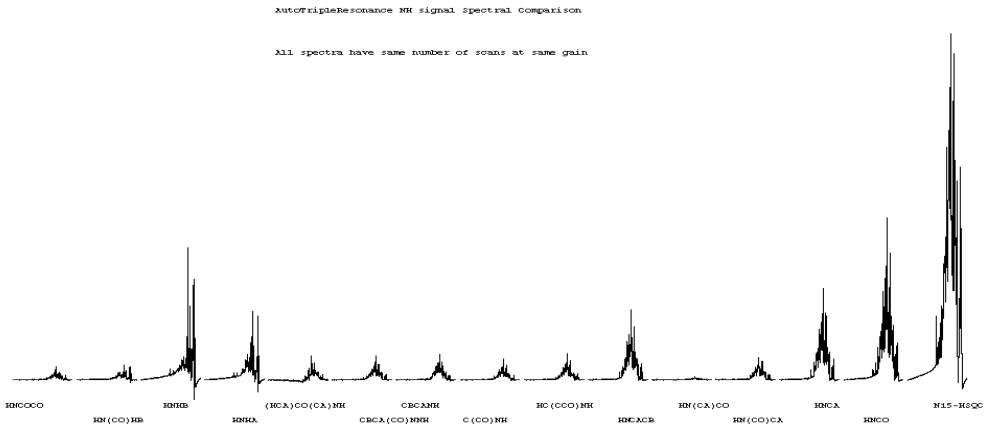


Figure 39 The NH-Receptivity Plot

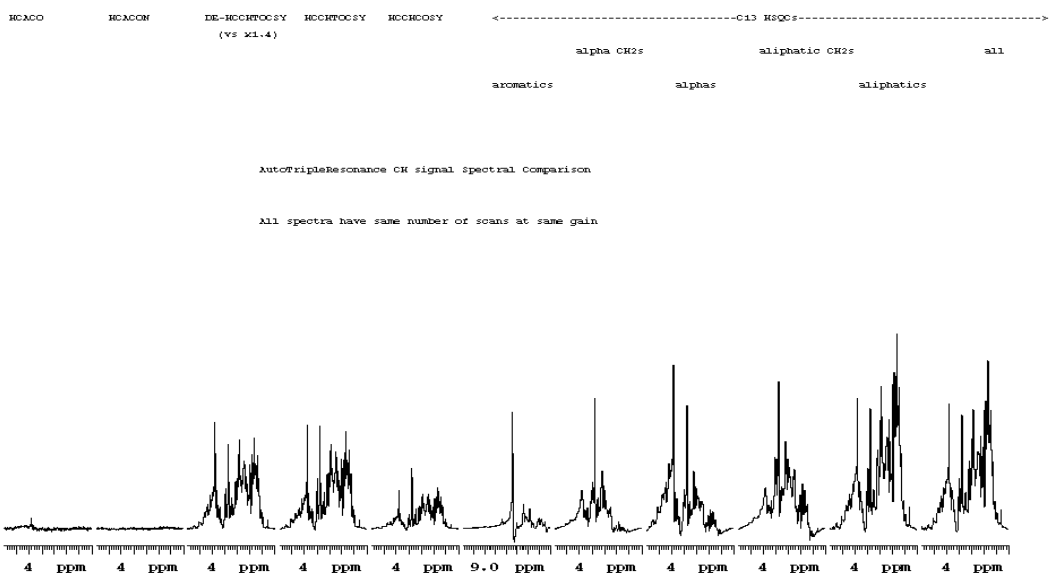
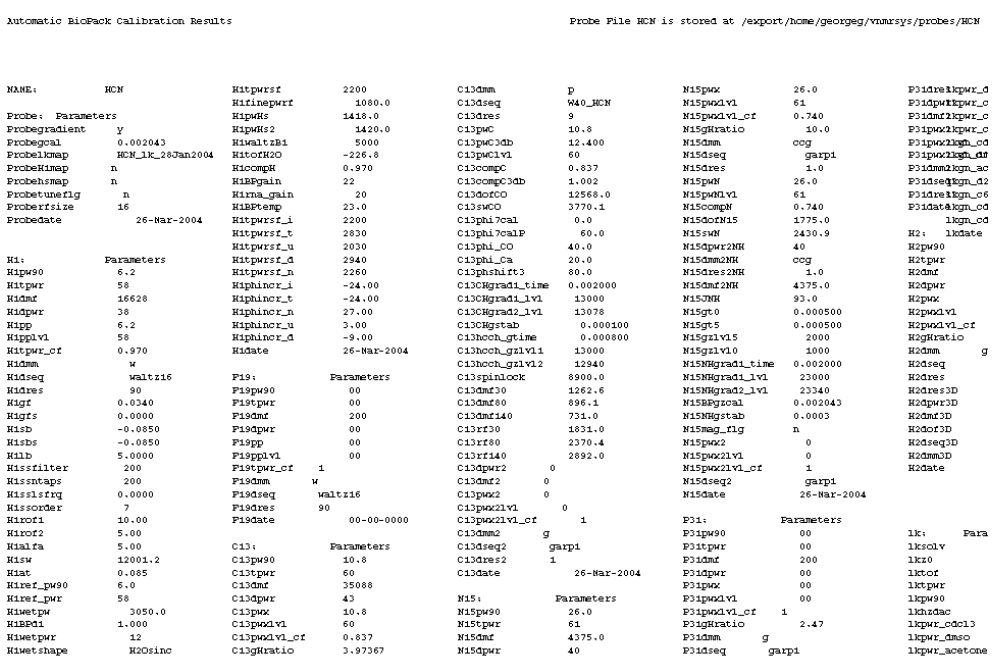


Figure 40 "Receptivity" plot for CH-detected Experiments (same gain/nt/vs).



Automatic BioPack Calibration Results — Probe File HCN is stored at /export/home/georgeg/vnmrsys/probes/HCN

NAME:	HCN	H1tpwrsf	2200	C13dmm	p	N15pwx	26.0	P31dres	lkpwr_d		
		H1finepwrf	1080.0	C13dseq	W40_HCN	N15pwx1v1	61	P31dpwr	lkpwr_c		
Probe: Parameters		H1pwHs	1418.0	C13dres	9	N15pwx1v1_cf	0.740	P31dmf2	lkpwr_c		
Probegradient	y	H1pwHs2	1420.0	C13pwC	10.8	N15gHratio	10.0	P31pwx2	lkpwr_c		
Probegcal	0.002043	H1waltzB1	5000	C13pwC3db	12.400	N15dmm	ccg	P31pwx2	lkgh_cd		
Probelkmap	HCN_lk_28Jan2004	H1tofH2O	-226.8	C13pwC1v1	60	N15dseq	garp1	P31pwx2	lkgh_df		
ProbeHimap	n	H1compH	0.970	C13compC	0.837	N15dres	1.0	P31dmm2	lkgm_ac		
Probehsmap	n	H1BPgain	22	C13compC3db	1.002	N15pwN	26.0	P31dseq	lkgm_d2		
Probetuneflg	n	H1rna_gain	20	C13dofCO	12568.0	N15pwN1v1	61	P31dres	lkgm_c6		
Proberfsize	16	H1BPtemp	23.0	C13swCO	3770.1	N15compN	0.740	P31date	lkgm_cd		
Probedate	26-Mar-2004	H1tpwrsf_i	2200	C13phi7cal	0.0	N15dofN15	1775.0		lkgm_cd		
		H1tpwrsf_t	2830	C13phi7calP	60.0	N15swN	2430.9	H2:	lkdate		
		H1tpwrsf_u	2030	C13phi_CO	40.0	N15dpwr2NH	40	H2pw90			
H1:	Parameters	H1tpwrsf_d	2940	C13phi_Ca	20.0	N15dmm2NH	ccg	H2tpwr			
H1pw90	6.2	H1tpwrsf_n	2260	C13phshift3	80.0	N15dres2NH	1.0	H2dmf			
H1tpwr	58	H1phincr_i	-24.00	C13CHgrad1_time	0.002000	N15dmf2NH	4375.0	H2dpwr			
H1dmf	16628	H1phincr_t	-24.00	C13CHgrad1_lv1	13000	N15JNH	93.0	H2pwx			
H1dpwr	38	H1phincr_n	27.00	C13CHgrad2_lv1	13078	N15gt0	0.000500	H2pwx1v1			
H1pp	6.2	H1phincr_u	3.00	C13CHgstab	0.000100	N15gt5	0.000500	H2pwx1v1_cf			
H1pplv1	58	H1phincr_d	-9.00	C13hcch_gtime	0.000800	N15gzlv15	2000	H2gHratio			
H1tpwr_cf	0.970	H1date	26-Mar-2004	C13hcch_gzlv11	13000	N15gzlv10	1000	H2dmm	g		
H1dmm	w			C13hcch_gzlv12	12940	N15NHgrad1_time	0.002000	H2dseq			
H1dseq	waltz16	P19:	Parameters	C13spinlock	8900.0	N15NHgrad1_lv1	23000	H2dres			
H1dres	90	P19pw90	00	C13dmf30	1262.6	N15NHgrad2_lv1	23340	H2dres3D			
H1gf	0.0340	P19tpwr	00	C13dmf80	896.1	N15BPgzcal	0.002043	H2dpwr3D			
H1gfs	0.0000	P19dmf	200	C13dmf140	731.0	N15NHgstab	0.0003	H2dmf3D			
H1sb	-0.0850	P19dpwr	00	C13rf30	1831.0	N15mag_flg	n	H2dof3D			
H1sbs	-0.0850	P19pp	00	C13rf80	2370.4	N15pwx2	0	H2dseq3D			
H1lb	5.0000	P19pplv1	00	C13rf140	2892.0	N15pwx2lv1	0	H2dmm3D			
H1ssfilter	200	P19tpwr_cf	1	C13dpwr2	0	N15pwx2lv1_cf	1	H2date			
H1ssntaps	200	P19dmm	w	C13dmf2	0	N15dseq2	garp1				
H1slsfrq	0.0000	P19dseq	waltz16	C13pwx2	0	N15date	26-Mar-2004				
H1ssorder	7	P19dres	90	C13pwx2lv1	0						
H1rof1	10.00	P19date	00-00-0000	C13pwx2lv1_cf	1	P31:	Parameters				
H1rof2	5.00			C13dmm2	g	P31pw90	00	lk:	Para		
H1alfa	5.00	C13:	Parameters	C13dseq2	garp1	P31tpwr	00	lksolv			
H1sw	12001.2	C13pw90	10.8	C13dres2	1	P31dmf	200	lkz0			
H1at	0.085	C13tpwr	60	C13date	26-Mar-2004	P31dpwr	00	lktof			
H1ref_pw90	6.0	C13dmf	35088			P31pwx	00	lktpwr			
H1ref_pwr	58	C13dpwr	43	N15:	Parameters	P31pwx1v1	00	lkpw90			
H1wetpw	3050.0	C13pwx	10.8	N15pw90	26.0	P31pwx1v1_cf	1	lkhzdac			
H1BPd1	1.000	C13pwx1v1	60	N15tpwr	61	P31gHratio	2.47	lkpwr_cdcl3			
H1wetpwr	12	C13pwx1v1_cf	0.837	N15dmf	4375.0	P31dmm	g	lkpwr_dmso			
H1wetshape	H2Osinc	C13gHratio	3.97367	N15dpwr	40	P31dseq	garp1	lkpwr_acetone			

Figure 41 Probefile Printout following AutoCalibration

11.7 Offsets for RNA and DNA

The spectral widths provided are adequate for 600 MHz so they can be scaled to your sfrq. The value of dof2 (N15) should be 200 ppm for all experiments. All dof (C13) values are 110ppm. These are changed appropriately in the psglib code for the specific pulse sequences

It is not necessary to use an RNA or DNA sample for AutoCalibration. A dilabeled protein sample can be used. RNA and DNA sequences use the same parameters as used by the protein experiments. Only spectral widths and offsets are different and these are handled by the experiment setup macros.

11.8 Auto1D and Auto2D

One of the useful features of the full AutoCalibration is the comparison of “receptivities” of a given sample to different pulse sequences (see above). If a full AutoCalibration has been done recently enough no 13C, 15N, or gradient calibration is likely to have changed. The probe is usually retuned for a new sample so that the 1H calibration may change (along with any related calibrations such as H2O selective pulses). If a “receptivity plot” is desired, but without the time necessary for calibrations, the Auto1D option may be selected. This has an option for first determining the 1H pw90 followed by a series of first increment spectra and the “receptivity” plots. Thus, a new sample can be rapidly screened for sensitivity. All data acquired are stored in the user’s vnmrsys/AutoTripRes1D directory. As in the case of the full autocalibration, any existing AutoTripRes1D file is backed up in the BioPack.dir directory present in the user’s vnmrsys prior to a new run.

Given that a new sample has been through an autocalibration, the Auto2D options permit a user to survey 2D “receptivities” as well. In this case the nature of the 2D spectra is of interest, perhaps indicating the relaxation or aggregation properties of the sample. Upon selecting the option, at the end of the series of 2D acquisitions the data can be automatically processed and plotted. By adjusting the d1 delay, options are available for 4, 8 and 16 hour runs. This is automatically done by selecting a button for a 4-, 8- or 16-hour run. In all cases, the ni and ni2 values are pre-selected to provide adequate resolution. The nt values are pre-selected based on the relative sensitivities of the experiments. The automatic processing is provided for as a convenience to the user. Individual data sets can be manually processed to reveal the full quality of the data.

11.8.1 Understanding What Happens “at the Click of a Button”

The most direct and easy way to perform all the above operations is to use the controls in the “Globals and Probefile” and “Calibrations” Setup pages through the use of entry boxes, displayed parameter values (textmessages), checkboxes and buttons. VnmrJ substantially expands the use of “obvious” control elements and has many pages to contain them. This leads to the ability to run all BioMolecular NMR experiments without recourse to a command line.

For those needing to go beyond this interface and find out what is actually done when a button is clicked or an entry box filled in, the panel (or page) editor is useful. This should be the first step in getting an understanding. The editor for VnmrJ is activated by the Edit function of the MainMenu (top line in the VnmrJ display). Use the Parameter Panel selection and a popup will appear. Double-clicking on the widget of interest will show the details in the popup.

Any underlying macro is displayed and it can then be read and understood using any editor program on the file in the relevant maclib. These interfaces can be customized by the user to add new buttons or other widgets in order to run user-created macros, adding to the usability of the interface.

Chapter 12 How to Add a New Experiment into BioPack

BioMolecular NMR experiments require, at a minimum, a compiled pulse sequence and an appropriate parameter set that includes all parameters used by the pulse sequence. In principle, it is not even necessary for the user to be able to view the values of parameters for the experiment to function. Of course, this is not acceptable, so visual displays are necessary to review the parameter values before beginning the experiment. While it is possible to just do an ordinary “rtp” to install the parameter set into the current Experiment, Section 5.0 above clearly demonstrates the power of using a setup macro. Many, if not most, Biomolecular experiments use one or more shaped pulses and waveform-based decoupling shapes. These need to be supplied or created if not done by the sequence itself.

1. Prepare an Experiment Setup Macro

   Chapter 9 illustrated the syntax and structure of a setup macro. These can be very simple or more elaborate, depending on the operations desired (and predictable). The general philosophy is to begin with BPrtppar, which calibrates the newly-setup parameter set from values in the probefile (used or not). Then parameters specific to the pulse sequence are set, followed by settings of parameters and initializations. Any new sequence added to BioPack should be written in a BioPack parameterization, if possible. However, the AutoCal approach (*A.c sequences) permit non-BioPack style sequences to be run as if they were. Normally, the setup macro has the same name as the C code, but without the “.c” at the end.

2. Prepare a Parameter Set

   The simplest way to start is to set up a related BioPack sequence or one that has the same number of dimensions and is of similar complexity in channels and gradients. This means that most of the parameters you need will already be present. Then set seqfil to the name of the new sequence. You should modify the text file in the parameter set appropriately. Next, any new parameters needed but not present need to be created using the “create” command. A fast way to check for completeness is to run “dps” with a text window open. Missing parameters will be listed. Next, the new parameters should be displayable. A convenient way to create parameters is via the “Attributes / Create / Destroy” menu in the “Text Out” page of the “Setup” folder.

   The oldest display of parameters is the “dg”-style of text output. The display is static, but is updated by VnmrJ as appropriate. Not all parameters need to be present in the “dg” panels, but important parameters should be visible. The command “paramvi(‘dg’)” permits the user to edit the “dg” parameter to add new parameters. There always is a “dg” parameter in the parameter set (in the “procpar” file) and this parameter can be modified by the editing process. Analogously, the “ap” parameter should be edited in the same way for the printed parameter table.The “Edit a File” button in the “Text Out” page of the Setup folder is useful for for editing “dg” and “ap”.

   New pulse sequence parameter sets should always have a proper “dg” family and “ap” parameters for useful and accurate display and reporting. After editing these files the parameter set should be saved to disk to avoid future need to repeat the editing. (The “dg”-style of display involves just parameter names and values, but not descriptive “NMR

language” phrases that appear on buttons or next to entry boxes. These capabilities are present only in the xml (VnmrJ) panels.)

3. Prepare a Manual File

   New experiments should have a “manual” file present in either /vnmr/manual, ~/vnmrsys/manual or in the active “appdir”. This text file is often the top part of a pulse sequence showing any comments provided by the author and gives suggestions as to parameter values, processing information and other points peculiar to the pulse sequence. This file usually has the same name as the sequence (without the “.c”) and is displayed in the text window by the “man(‘name’)” command or by any displayed “Sequence Manual” button. This command is often used within an experiment setup macro. It is available via button in the Text Output Page of the Acquire folder in VnmrJ.

4. Modify the Experiment Selection Menu

   The user should not have to use the command line to set up the experiment. This means that the drop-down menu in VnmrJ should be edited to include a function to run any new setup macro. The VnmrJ files are in /vnmr/templates/VnmrJ/interface/MainMenu.xml, ~/vnmrsys/templates/VnmrJ/interface/MainMenu.xml or in the relevant “appdir” file. These latter files call submenus. The relevant submenu should be changed to provide the additional experiment. All of these include statements that run the experiment setup macros. The syntax is obvious and one of the existing sections can just be copied and the name of the experiment changed.

5. Edit a Set of Experiment-Specific Parameter Panels

   A new experiment will often need new parameter entry “widgets” in the layout pages. As above, the easiest and fastest method is to use a shell and go to your user file ~/vnmrsys/templates/layout. If this is not present, make the directory. Then copy a set of panels from a related or similar experiment from /vnmr/templates/layout if BioPack is installed in /vnmr, from the local layout directory or from the relevant “appdir” file. Then rename this directory to the name of the new sequence. Now, the VnmrJ interface should reflect the related experiment. Use the parameter set already saved above. Either use your newly created setup macro, or recall the parameter set directly into the current experiment. The panels should look the same. Any new parameters, options, flags, etc should be added to the panel using the panel editor (Main Menu / Edit / Parameter Pages). Consult the User Programming Manual for details on panel editing.

   The panel editor also permits seeing the macros called by different “widgets”. When the panels are “saved” the relevant text files are stored in ~/vnmrsys/templates/layout (for VnmrJ). They are stored under the pulse sequence name. When the collection of files is complete the directory may be copied /vnmr/templates/layout (for VnmrJ) for use by all users or into a relevant “appdir” directory.

   These files should be created for maximum clarity and performance. The latter is accomplished by having the widgets run macros that use vnmr variables and Magical commands to make a task easy to do, but accurate and reliable. For example, one might have a button to set up an array, perform and acquisition and at the end perform processing that obtains a calibration value. Or, when a variable has a finite list of options, a radio button widget permits the variable to be set at one click, where the user sees the options with descriptive text alongside.

   Again, it is not necessary to have these panels for a new sequence to run. The default panels will be used by the software, but the power of the interface will be severely compromised.

   As a convenience to users running non-BioPack sequences, the default panels used are dependent on the existence of the parameter “tsatpwr”. If “tsatpwr” is present, a set of panels are displayed that cover a large number of non-BioPack sequences. If your

parameter set does not contain the parameter “tsatpwr” and does not have a BioPack set of pages, the 1D default page is displayed, but it does have a button that will create the “tsatpwr” parameter so that the more relevant panels can be displayed.

6. Create any Required Shapes

   Any new shapes required by a pulse sequence must reside in either /vnmr/shapelib or ~/vnmrsys/shapelib. These may be derived “on-the-fly” by Pbox statements within the pulse sequence code, or by the setup macro calling Pbox or a user-written program. As discussed above, the AutoUpdate (BPbiopack2) or AutoCalibrate macros run the macros BPsetupshapes and BPcal that produce any shapes needed by the pulse sequences provided.

   If your new sequence is constructed to use already-created BioPack shapes, no shapes are needed. However, the setup macro should set the parameter values such as shape name, shaped pulse width, shape power, etc. If special shapes are used they should be created and stored as .RF or .DEC files. Your setup macro or pulse sequence should set up the necessary parameter values for these shapes in your parameter set based on knowledge of the calibrations in the relevant channels. These are supplied by the BPrtppar macro and are therefore accessible to statements that follow within the setup macro.

   Once these files have been written, they may be put directly into the active BioPack directories and they will operate just as any other experiment. If you desire to share your work with others please submit the files as a compressed tar file to Varian (see BioPack.README in BioPack.dir for contact information) and the new experiment will appear in the next update or release, along with an announcement in Varian MR News.

Chapter 13 Pulse Sequence Design – Standard “C” code

BioMolecular NMR experiments can be designed (through pulse sequences) in a variety of ways. The traditional approach is to design elements that have parameter control and to build the sequence by constructing a series of delays, power statements, pulses etc. that leave the operator to set specific values for all parameters. Thus, the user must calibrate the system to provide the “hard pulse” pw90’s on all channels (this can be automated, as discussed above). However, sequence-specific elements such as region-selective pulses, water flipback pulses, decoupling and gradients would require the user to create text files for shapes and predict approximate power levels and pulse widths for the RF elements. This can be tedious and error-prone. A linear and predictable RF system, coupled with powerful pulse sequence and macro languages, permits the use of mathematics to calculate these values from the primary calibrations obtained by the user.

The underlying basis for the success of the BioNMR experiment is the predictability of spectrometer output. This means that calibrations of pulse widths at single power levels are sufficient for prediction of power levels for region-selective pulses. This is highly relevant for 13C in all of these experiments, but also for protons, where soft selective pulses are done on the water and spinlock periods are employed.

The other factor allowing parameter prediction is that natural resonance frequencies for carbon and proton nuclei in proteins are observed to fall in specific regions with few but predictable exceptions. Each of these experiments is designed so that the 13C frequency is set at a specific 13C chemical shift (e.g. 56 ppm for alpha carbons in HNCA). The proton frequency is always placed at the water position and the 15N frequency in the center of the amide region. Hence, the frequency change to the C=O region is known at the existing magnetic field(e.g. (174-56)*dfrq in Hz).

Since all of these sequences involve region-selective 90’s and 180’s on the carbonyl, alpha, or alpha/beta carbon regions, the lengths of the pulses can be easily and automatically calculated so as to provide the required excitation nulls (e.g. at 600 Mhz a 55 usec 90 degree pulse centered at 56 ppm has its first excitation null at 174 ppm). The BPcal macro, for example, uses the spectrometer frequency “sfrq” to automatically create all shapes needed by many pulse sequences. Similarly, the BPsetupshapes macro can calculate decoupling waveforms for many types of band-selective 13C decoupling. In principle, the *.RF files need only be created once since the magnetic field and offset positions (in ppm) are not changed. This forms the basis for the first implementation of BioMolecular NMR sequences currently used (as developed by Robin Bendall).

The power levels necessary for these pulses do need to be determined so that the proper flip angles are generated. This operation is done as a part of “go” within the pulse sequence, based on the tpwr, pwClvl, pwNlvl and amplifier compression values. The 13C and 15N coarse attenuators are not changed and all power levels can be adjusted via the linear modulators present on each channel, or Pbox can be called within the pulse program to generate the shapes (based on the same calibrations) and obtain the parameters for power and pulse timing (see below). Thus, as long as the “hardpulse” pulse widths are calibrated, nothing else is required. The flexibility necessary to do this is, of course, a property of the “c” language used in the pulse program. These principles are illustrated below in the section 13.1 for the HNCO experiment (ghn_co.c).

13.1 Excerpts from the ghn_co.c Pulse Sequence Code

The first part of most sequence codes is a comment section:

```
/*  ghn_co.c

Correlates CO(i) with N(i+1), NH(i+1).  Uses constant time evolution for
the 15N shifts.

Standard features include maintaining the 13C carrier in the CO region
throughout using off-res SLP pulses; square pulses on Ca with first
null at 13CO; one lobe sinc pulses on 13CO with first null at Ca; one
lobe
sinc pulse to put H2O back along z (the sinc one-lobe is significantly
more
selective than gaussian, square, or seduce 90 pulses); preservation of
H20
along z during t1 and t2; waltz H1 decoupling during N15 evolution to
decrease S/N loss via H1 exchange with H2O.

Magic-angle option for coherence transfer gradients.  TROSY option for
N15/H1 evolution/detection. */
,
, .    (statements omitted)
,
```

Later in the code, parameter values are obtained to be used in expressions. New variables are defined if needed (no getval). (All 13C power levels are set using the fine power (linear modulator) settings which can be set between 0 and 4095. The channel 2 attenuator used for 13C is left constant at pwClvl):

```
.
.
.    (statements omitted)
.
/* the following pulse lengths for shaped pulses are automatically
calculated */
/* by the experiment setup macro (ghn_co). Pulse shapes, "offC3" etc,
were    */
/* already created by the macro “BPcal”. They are stored in your
shapelib or  */
```

```
/* the system shapelib. The getvals obtain values from the your
parameter set  */

pwC3 = getval("pwC3"),   /*180 degree pulse at Ca(56ppm) null at
CO(174ppm) */

pwC6 = getval("pwC6"),        /* 90 degree selective sinc pulse on
CO(174ppm) */

pwC8 = getval("pwC8"),         /* 180 degree selective sinc pulse on
CO(174ppm) */

rf3,                                  /* fine power for the pwC3
("offC3") pulse */

rf6,                                 /* fine power for the pwC6 ("offC6")
pulse */

rf8,                                  /* fine power for the pwC8
("offC8") pulse */

pwHs = getval("pwHs"),               /* H1 90 degree pulse length at
tpwrs */

tpwrsf = getval("tpwrsf"),                /* fine power for pwHs pulse
*/

tpwrs,                               /* power for the pwHs ("H2Osinc")
pulse */

pwHd,                             /* H1 90 degree pulse length at tpwrd */

tpwrd,                               /* attenuator setting for 1H WALTZ
decoupling */

.

.

.
```

In this section, expressions are used (from theory) to calculate the fine power settings for the linear modulator on channel 2 for the band-selective 13C pulses:

```
.

.

.


/* Calculate fine power for 180 degree pulse on Ca, null at CO 118ppm
away */

rf3 = 4095.0*(compC*pwC*2.0/pwC3);

rf3 = (int) (rf3 + 0.5);

 /* Same for 90 degree one-lobe sinc pulse on CO, null at Ca 118ppm away
*/

rf6 = 4095.0*(compC*pwC*1.69/pwC6);       /* a sinc 90 needs 1.69 times
more */

rf6 = (int) (rf6 + 0.5);                /* power than a square pulse */
```

```
/* Same for 180 degree one-lobe sinc pulse on CO, null at Ca 118ppm away
*/
rf8 = 4095.0*(compC*pwC*2.0*1.65/pwC8);         /* a sinc 180 needs 1.65
times more*/
rf8 = (int) (rf8 + 0.5);                 /* power than a square pulse */
.
.
.
```

The rf3 value is calculated using the value of the selective pulse pwC3 and the hard-pulse 180-degree pulse calibration of (compC*pwC*2.0) (compC corrects pwC to the uncompressed value). The expression can be understood as a scaling down of 4095 by a factor of the ratio of the hard-pulse 180-degree pulse length to the region-selective 180-degree pulse length. In the trivial case of pwC3 being twice as long as compC*pwC*2.0, the ratio is 0.5 and the resulting fine power is just 2048, as expected.

The rf6 and rf8 cases are for “sinc” pulses. The fine powers have to be multiplied by a theoretical factor to compensate for the calibration being done with a rectangular pulse, whereas the region-selective pulses are to be done by a shaped pulse. Otherwise, the same considerations apply as for rf3.

Aside from non-selective 1H pulses done using tpwr and pw, 1H decoupling and selective H2O flipback pulses are needed. The power level for these pulses is controlled by the attenuator only. Again, the expressions use a ratio involving a high power pulse width and a pulse width at lower power:

```
.
.
.

/* selective H20 one-lobe sinc pulse done at power level tpwrs and pulse
width pwHs*/
    tpwrs = tpwr - 20.0*log10(pwHs/(compH*pw*1.69));        /* sinc
needs 1.69 times more */
    tpwrs = (int) (tpwrs+0.5);                               /* power
than a square pulse */


/* power level and pulse time for WALTZ 1H decoupling at field strength
waltzB1(Hz)*/
      pwHd = 1/(4.0 * waltzB1) ;
      tpwrd = tpwr - 20.0*log10(pwHd/(compH*pw));
      tpwrd = (int) (tpwrd + 0.5);
```

After all the calculations the actual pulse sequence is ready to begin. The next section shows how the calculated values are used for the region selective pulses offC6 and offC8, as well as the water flipback pulse(the fine power parameter tpwrsf allows adjustment of the power to account for radiation damping):

```
.
.
.

      decpwrf(rf8);                              /* set fine power */
      decshaped_pulse("offC8", pwC8, zero, 0.0, 0.0);
      decpwrf(4095.0);         /* reset fine power to maximum value  */
.
.
.

      decpwrf(rf6);
      decshaped_pulse("offC6", pwC6, t5, 0.0, 0.0);
      decpwrf(4095.0);
.
.
.

        obspower(tpwrs);    /* set attenuator for H2O selective pulse */
         if (tpwrsf<4095.0)
         obspwrf(tpwrsf);    /* permits fine adjustment of soft pulse */
       shaped_pulse("H2Osinc", pwHs, three, 0.0, 0.0);
        obspower(tpwr);          /* reset attenuator to normal power  */
.
.
.
}
```

13.2 “C” Pulse Sequence Design using Pbox

Two alternative methods of using phase-ramped pulses are included. These methods call Pbox for shape creation at the time of "dps" and "go". No shapes are required in shapelib for this method. These sequences have a "P" (BioNMR) or "A" (AutoCalibrated or Pbox-based) appended to the normal names.

13.2.1 BioNMR (“P”) Versions

The “*P.c” sequences must be compiled the "bionmr.h" include file, while the “*A.c” sequences requires the "Pbox_bio.h" include file (present in /vnmr/psg or ~/vnmrsys/psg).

The "*P.c", or BioNMR, versions use non-standard psg elements that are written with protein 13C bandwidths pre-defined so that terms such as "ca", "co" and "cacb" are used. The operation, parameters and performance of these are at least as good as the original sequences. The only differences are that the phi7cal value may be different, and that the "dof" value is always set to the carbonyl carbon frequency for all experiments. The virtue of this approach is that new sequences can still be written in “C”, but using high-level elements that do not require mathematical expressions (as above) to calculate power levels or pulse widths, and that no shapes need be prepared ahead of time. P sequence codes are typically half the length of the original versions. This method of sequence is of most use when a completely new sequence is written.

BioNMR parameter sets are accessible either by directly entering the macro, e.g.

ghn_co_caP or by selecting the BioNMR Version button in the Pulse Sequence page in the Acquire folder after selecting the generic experiment from the drop-down menu.

Remember the dof value for all P.c pulse sequences should be set for the carbonyl region. The BPbiopack2 macro does update the probefile for this offset, and experiment setup macros for P.c sequences set dof to this value. In addition, buttons on the PulseSequence.xml page that convert to the BioNMR (*P.c) version usually change dof to 174ppm.

The phi7cal value in both standard or BioNMR parameter sets is determined in the automatic calibration process, which updates the probefile. BioNMR experiments using phi7cal should use the phi7cal value for BioNMR sequences. The next Section shows how the same ghn_co.c sequence elements described above are modified for ghn_coP.c.

13.2.2 Excerpts from the ghn_coP.c Pulse Sequence Code

The first part of most sequence codes is a comment section:

```
/*  ghn_coP.c
.
.
.

    3D HNCO gradient sensitivity enhanced version using Pbox to generate
shapes.
    The N15 t2 evolution and the sensitivity enhancement train is common
    to all ghn_... sequences and the pulse sequence code for these final
    sections is in the include file, bionmr.h.
*/
.
.
.
```

The bionmr.h include file must be added…

```
#include <standard.h>
```

```
#include "bionmr.h"
.
.
.

 pwHs = 1.7e-3*500.0/sfrq;     /* length of flipback, 1.7ms at 500 MHz*/
 widthHd=2.681*(waltzB1/sfrq);/* 2.681 factor produces same bandwidth */

pwHd = h1dec90pw("WALTZ16", widthHd, 0.0);
  /* H1 90 length for WALTZ16 */

  /* get calculated pulse lengths of shaped C13 pulses */
pwS1 = c13pulsepw("co", "ca", "sinc", 90.0);
pwS2 = c13pulsepw("ca", "co", "square", 180.0);
.
.
.
```

The BioNMR “h1dec90pw” function returns the pulse width for a 90 degree pulse at the power level needed to produce the decoupling field widthHd (in Hz). The BioNMR function “c13pulsepw” returns the pulse width corresponding to the specified shape, region and flip-angle. The first two arguments specify the position at which the excitation occurs and the first null in the excitation. Note that these two arguments are in “protein” language and specify the alpha and carbonyl regions of the 13C spectrum. The pulse widths are not obtained from the parameter table, so a function is needed to obtain the values which are needed in subsequent expressions in the pulse sequence code.

Later, at the start of the pulse sequence, the BioNMR statement “set_c13offset” is used to position the 13C transmitter. Again, note that the argument is not an offset, but a codeword. The only power statements needed are those for the high-power hard pulses. No 13C pulse widths are used, other than the hard-pulse value pwC.

```
.
.
.

set_c13offset("co");
.
.
.
```

The H_2O selective pulse power and pulse width are handled by the BioNMR function “shiftedpulse” which permits specification of shape, pulse width, flip-angle, and offset from tof. No power level is needed:

```
.
.
.

shiftedpulse("sinc", pwHs, 90.0, 0.0, two, 2.0e-6, 0.0);
.
.
.
```

Proton decoupling is handled by the BioNMR function “h1waltzon” which specifies the decoupling pattern, field strength (in Hz) and offset from tof. No power values or pulse widths are needed:

```
.
.
.
h1waltzon("WALTZ16", widthHd, 0.0);
.
.
.
```

The 13C band-selective pulses power and pulse width are handled by the BioNMR function “c13pulse” which permits specification of shape, flip-angle, region to be excited and first null region.

The sequence ends with the BioNMR function “hn_evol_se_train” which performs the last ~30% of the sequence including a band-selective 13C 180-degree pulse (and first null) during N15 evolution, t2 evolution, TROSY option, gradient coherence transfer, reverse INEPT and 15N decoupling during detection:

```
.
.
.

c13pulse("co", "ca", "sinc", 180.0, zero, 2.0e-6, 0.0);
.
.
.

c13pulse("co", "ca", "sinc", 90.0, t5, 2.0e-6, 0.0);
.
.
.
hn_evol_se_train("co", "ca");
}
```

13.2.3 Summary of BioNMR psg elements

Offset:

set_c13offset(fband)

C13 Pulses:

c13pulsepw(excband, nullband, c13shape, c13flip)

c13pulse(excband, nullband, c13shape, c13flip, phase2, rofa, rofb)

c13adiab_inv_pulse(excband, bandwidth, c13shape, pulsewidth, phase2, rofa, rofb)

Simultaneous 1H/13C Pulses

sim_c13pulse(obsshape, excband, nullband, c13shape, obspw, c13flip ,phase1, phase2, rofa, rofb)

sim_c13adiab_inv_pulse(obsshape, excband, bandwidth, c13shape, obspw, pulsewidth, phase1, phase2, rofa,rofb)

Simultaneous 1H/13C/15N Pulses

sim3_c13pulse(obsshape, excband, nullband, c13shape, dec2shape,
obspw, c13flip, dec2pw, phase1, phase2, phase3, rofa, rofb)

sim3_c13adiab_inv_pulse(obsshape, excband, bandwidth, c13shape, dec2shape,
obspw, pulsewidth, dec2pw, phase1, phase2, phase3, rofa, rofb)

Frequency-Shifted 1H Shaped Pulse

shiftedpulse(anyshape, pwbw, flip, shift, phase1, rofa, rofb)

Frequency-Shifted 13C Shaped Pulse

decshiftedpulse(anyshape, pwbw, flip, shift, phase2, rofa, rofb)

Frequency-Shifted 15N Shaped Pulse

dec2shiftedpulse(anyshape, pwbw, flip, shift, phase3, rofa, rofb)

Frequency-Shifted 1H/13C/15N Simultaneous Centered Shaped Pulses

sim3shiftedpulse(obsshape, anyshape, dec2shape, obspw, pwbw, flip, shift,
dec2pw, phase1, phase2, phase3, rofa, rofb)

13C Decoupling

c13decon(decband, c13shape, decbandwidth)

c13decoff()

c13decouple(decband, c13shape, decbandwidth, duration)

1H Decoupling:

h1decon(h1shape, decbandwidth, shift)

h1decoff()

h1decouple(h1shape, decbandwidth, shift, duration)

h1waltzon(h1shape, decbandwidth, shift)

h1waltzoff(h1shape, decbandwidth, shift)

N15 t2 Evolution with 13C Refocussing and Sensitivity-Enhancement

nh_evol_se_train(excband, nullband)

Identify Desired Decoupling Waveform for Pbox

installdecshape(decshape)

Shape File Parameter Extraction Tool (to obtain parameters from Pbox-generatedshapes)

c13shapefiles(excband, nullband, c13shape, c13flip);

fshapefiles(anyshape, pwbw, flip, shift)

c13adiab_files(excband, bandwidth, c13shape, pulsewidth)

c13decfiles(decband, c13shape, decbandwidth)

h1decfiles(h1shape, decbandwidth, shift)

13.2.4 Automatic Pbox-Based Designs (the “AutoCal” approach)

The "A" versions include a header section in the pulse sequence where all shapes are created and the parameter values from these shapes are obtained at "go" time. This mode of operation allows the conversion of existing pulse sequences of any style to “A” versions without modification of the underlying codes (developed by Eriks Kupce, Varian). The difference between an “A”-sequence and the corresponding “BioNMR, or P-” sequence is that the later introduces a completely new syntax for executing shaped pulses, whereas the “A” sequence requires minimal interference with the existing pulse sequence code. Therefore, the syntax that is used in the “A” sequences provides a quick and easy way to convert long and extensive pulse programs into spectrometer-frequency independent, fully automatic experiments using the same parameter names and calibrations as standard experiments. The ultimate goal is to reduce the portability issues of complex multi-dimensional bio-NMR experiments. The steps necessary to convert a “standard” (ghn_co.c) pulse sequence are illustrated below.

13.2.5 Excerpts from the ghn_coA.c Pulse Sequence Code

```
/*  ghn_coA.c

3D HNCO gradient sensitivity enhanced version using Pbox to generate
shapes.

AutoCal version with optional offset checking.

*/

.

.

.
```

(existing ghn_co.c code statements)

.

.

.

This next part is added after the `#include <standard.h>` statement:

```
#include "Pbox_bio.h"          /* include new AutoCalfunctions */
static double   H1ofs=4.7, C13ofs=174.0, N15ofs=120.0, H2ofs=3.2;
static shape H2Osinc, wz16, offC3, offC6, offC8;
.
.
```

(existing ghn_co.c code statements)

.

.

```
double bw, ofs, ppm, ...      /* bandwidth, offset, ppm  are only new
parameters*/
.
.
```

(existing ghn_co.c code statements)

.

.

If autocal='n' the pwC3, pwC6 and pwC8 values will be obtained from the parameter set in the normal manner using the getval statements already used in the statements above and the rf3, rf6 and rf8 values will be calculated in the above section as before.

If autocal='y' the next section will calculate the needed shapes based on pwC, pwClvl and compC (using pbox_make) and then set the appropriate pulse widths pwC3, pwC6 and pwC8 to be used by the shaped pulse statements within the main body of the code. The fine powers rf3, rf6 and rf8 are also obtained from the header of the shape in the relevant shapelib. The shapes are created at "go" and are only done for the first increment.

Note that the arguments of "pbox_make" are the name of the new shape, the type of shape, the bandwidth of the pulse (in Hz), the offset from the 13C transmitter at which the excitation occurs and the calibration pulse width and power.

The proton decoupling waveform and the H2O selective shape are created based on the calibrations compH*pw and tpwr. The relevant pwHs, tpwrd and pwHd parameters are obtained from the shapefile header.

The optional offset check (ofs_check) produces a warning message if the specified actual offsets are in significant deviation from the actual defined shifts in ppm. This is done by an absolute chemical shift calculation.

```
setautocal();                    /* activate auto-calibration flags */
if (autocal[0] <> 'n')                 /* specifies automatic setup */
{
```

```
if(FIRST_FID)          /* calculate all shapes and do it only once */
{
ppm = getval("dfrq");      /* have Pbox use 13C frequencies, not 1H*/
bw = 118.0*ppm; ofs = -bw;                 /* calculate bw and ofs */
offC3 = pbox_make("offC3", "square180n", bw, ofs, compC*pwC, pwClvl);
offC6 = pbox_make("offC6", "sinc90n",    bw, 0.0, compC*pwC, pwClvl);
offC8 = pbox_make("offC8", "sinc180n",   bw, 0.0, compC*pwC, pwClvl);
H2Osinc = pbox_Rsh("H2Osinc","sinc90", pwHs, 0.0, compH*pw,  tpwr);
bw = 2.8*waltzB1;
wz16 = pbox_Dcal("WALTZ16", bw, 0.0, compH*pw, tpwr);
ofs_check(H1ofs, C13ofs, N15ofs, H2ofs);        /* check all offsets */
}
pwC3a = offC3.pw; rf3 = offC3.pw     /* get parameter values from shape
file*/
pwC6 = offC6.pw; rf6 = offC6.pwrf;
pwC8 = offC8.pw; rf8 = offC8.pwrf;
pwHs = H2Osinc.pw; tpwrs = H2Osinc.pwr-1.0;         /* 1dB correction
applied */
tpwrd = wz16.pwr; pwHd = 1.0/wz16.dmf;
}
```

The rest of pulse sequence follows as originally written. This is the virtue of the AutoCal approach. Any *.c code may be converted to an automatic version by inserting a similar section near the top of the existing code.

13.2.6 Automatic Pbox-based Pulse Sequence Definitions

setautocal() - Sets up the autocal and checkofs flags.

Syntax:(void) setautocal();

Description: Creates the autocal and checkofs flags and checks if they are set in the current experiment. If the flags are not set, they are defaulted to 'y' (yes). The available options for the checkofs flag are: 'y' and 'n' . For the autocal flag the available options are: 'y' , 'q' (quiet mode - suppress Pbox output), 'r' (read from file, no new shapes are created), 's' (semi- automatic mode - allows access to user defined parameters) and 'n' .

ofs_check() - Checks carrier frequencies on all 4 channels.

Syntax: (void) ofs_check(H1ofs, C13ofs, N15ofs, H2ofs);

double H1ofs, C13ofs, N15ofs, H2ofs;

Description: The ofs_check() function checks the settings of tof, dof, dof2 and dof3 of four RF channels against the values (in ppm) provided as arguments. The allowed deviations are set internally to 0.2 ppm for H-1 and H-2 and to 2.0 ppm for C-13 and N-15. It is assumed that channel 1 is H1, channel 2 is C-13, channel 3 is N-15 and channel 4 is H-2. The offset (tof, dof, dof2 and dof3) checks can be switched off individually by setting the corresponding argument to zero (0.0), or collectively by checkofs = 'n';

pbox_make() - Create and calibrate a shaped pulse.

Syntax:

```
shape   shp;
char    shn[MAXSTR], wvn[MAXSTR];
double  bw, ofs, ref_pw90, ref_pwr;
shp = pbox_make(shn, wvn, bw, ofs, ref_pw90, ref_pwr);
```

Description:

shp - pre-defined shape structure containing all the experimental parameters that are required in order to execute a shaped pulse or a decoupling waveform. shp must be declared as a shape.

shn - output shape name as used in the pulse sequence;

wvn - waveform name as in /vnmr/wavelib;

bw - excitation bandwidth (in Hz); alternatively pulse duration in seconds can be used;

ofs - excitation offset (in Hz) from the carrier frequency;

ref_pw90 - reference pulse length, e.g. compC*pwC;

ref_pwr - reference power level, e.g. pwClvl;

The pbox_make() function returns a shape structure (as defined in Pbox_psg.h) that contains all parameters required by the experiment.. For frequency-shifted pulses the maximum phase increment is internally restricted to 10 degrees, which guarantees a proper RF performance. The shapes are calibrated to use a fixed coarse attenuator setting that is set at ref_pwr level and a variable fine attenuator setting.

pbox_makeA() - Create and calibrate an adiabatic pulse.

Syntax:

```
shape    shp;
char     shn[MAXSTR], wvn[MAXSTR];
double   bw, pws, ofs, ref_pw90, ref_pwr, nst;
shp =    pbox_makeA(shn, wvn, bw, pws, ofs, ref_pw90, ref_pwr, nst);
```

Description:

shp - pre-defined shape structure containing all the parameters required in order to execute a shaped pulse or a decoupling waveform. shp must be declared as a shape.

shn - output shape name as used in the pulse sequence;

wvn - waveform name as in /vnmr/wavelib;

bw - excitation bandwidth (in Hz);

pws - pulse duration (in seconds);

ofs - excitation offset (in Hz) from the carrier frequency;

ref_pw90 - reference pulse length, e.g. compC*pwC;

ref_pwr - reference power level, e.g. pwClvl;

nst - number of steps

The same as pbox_make(), except simultaneous definition of bandwidth (bw) and pulse duration (pws) is allowed. This function is used to create and calibrate adiabatic pulses and waveforms. The nst parameter defines the number of steps in the shapefile. If nst is set to zero, the required number of steps is calculated by Pbox.

pbox_makeR() - Create and calibrate a shaped pulse

Syntax:

```
shape  shp;
char   shn[MAXSTR], wvn[MAXSTR];
double bw, ofs, ref_pw90, ref_pwr;
shp  =  pbox_makeR(shn, wvn, bw, ofs, ref_pw90, ref_pwr);
```

Description:

shn - output shape name as used in the pulse sequence;

wvn - waveform name as in /vnmr/wavelib;

bw - excitation bandwidth (in Hz); alternatively, pulse duration in seconds can also be used;

ofs - excitation offset (in Hz) from the carrier frequency;

ref_pw90 - reference pulse length, e.g. compC*pwC;

ref_pwr - reference power level, e.g. pwClvl;

The same as pbox_make(), except the status parameter is set to 1. The result is a time-reversed (de-excitation) pulse.

pbox_Rsh() - Create and calibrate a shaped pulse using coarse power

Syntax:

```
shape  shp;
char   shn[MAXSTR], wvn[MAXSTR];
double bw, ofs, ref_pw90, ref_pwr;
shp =  pbox_Rsh(shn, wvn, bw, ofs, ref_pw90, ref_pwr)
```

Description:

shn - output shape name as used in the pulse sequence;

wvn - waveform name as in /vnmr/wavelib;

bw - excitation bandwidth (in Hz); alternatively pulse duration in seconds can also be used;

ofs - excitation offset (in Hz) from the carrier frequency;

ref_pw90 - reference pulse length, e.g. compC*pwC;

ref_pwr - reference power level, e.g. pwClvl;

The same as pbox_make(), except the shapes are calibrated to use the coarse attenuator setting. The fine attenuator setting remains at 4095.0 and internal fine attenuation is used instead.

pbox_Dsh() - Create and calibrate a .DEC waveform using coarse power

Syntax:

```
shape  shp;
char   shn[MAXSTR], wvn[MAXSTR];
double bw, ofs, ref_pw90, ref_pwr;
shp =  pbox_Dsh(shn, wvn, bw, ofs, ref_pw90, ref_pwr);
```

Description:

shn - output shape name as used in the pulse sequence;

wvn - waveform name as in /vnmr/wavelib;

bw - excitation bandwidth (in Hz); alternatively pulse duration in seconds can also be used;

ofs - excitation offset (in Hz) from the carrier frequency;

ref_pw90 - reference pulse length, e.g. compC*pwC;

ref_pwr - reference power level, e.g. pwClvl;

The same as pbox_Rsh(), except .DEC type shape file is produced in the output.

pbox_Dec() - Create and calibrate a .DEC waveform for a given coarse power setting.

Syntax:

```
shape  shp;
char   shn[MAXSTR], wvn[MAXSTR];
double bw, decpwr, ref_pw90, ref_pwr;
shp =  pbox_Dec(shn, wvn, bw, decpwr, ref_pw90, ref_pwr);
```

Description:

shn - output shape name as used in the pulse sequence;

wvn - waveform name as in /vnmr/wavelib;

bw - excitation bandwidth (in Hz); alternatively pulse duration in seconds can also be used;

ofs - excitation offset (in Hz) from the carrier frequency;

ref_pw90 - reference pulse length, e.g. compC*pwC;

ref_pwr - reference power level, e.g. pwClvl;

The pbox_Dec() function is used to create decoupling and mixing waveforms. If a positive decpwr value is supplied, the dmf and dres parameters are calibrated as for the given decpwr; alternatively the required decoupling bandwidth can be used to calculate dpwr, dmf and dres. Note that in mixing experiments the effective mixing bandwidth is usually 2*B1(max) or less. This should be used to set up the bw parameter. The fine attenuator should be set to 4095.0;

pbox_Dcal() - decoupling calibration routine

Syntax:

```
shape    shp;
char     wvn[MAXSTR];                    /* parameter names */
double   bw, decpwr, ref_pw90, ref_pwr;
shp  =   pbox_Dcal(wvn, bw, decpwr, ref_pw90, ref_pwr);
```

Description:

wvn - waveform name as in /vnmr/wavelib;

bw - required decoupling bandwidth (in Hz), if 0.0 then not used;

decpwr - required decoupler power, if 0.0 then not used;

ref_pw90 - reference pulse length, e.g. compC*pwC;

ref_pwr - reference power level, e.g. pwClvl;

The same as pbox_Dsh(), except no shapefile is created in the output. The experimental parameter set is returned in the form of the shape structure, that contains shape.pwr (coarse power), shape.pwrf (fine power, set to 4095), shape.dres (dres parameter) and shape.dmf (dmf parameter);

pbox_Rcal() - selective pulse calibration routine

Syntax:

```
shape    shp;
char     wvn[MAXSTR];                    /* parameter names */
double   bw, ref_pw90, ref_pwr;
shp  =   pbox_Rcal(wvn, bw, ref_pw90, ref_pwr);
```

Description:

wvn - waveform name as in /vnmr/wavelib;

bw - required decoupling bandwidth; alternatively pulse duration in seconds can also be used;

ref_pw90 - reference pulse length, e.g. compC*pwC;

ref_pwr - reference power level, e.g. pwClvl;

The same as pbox_Dcal() except the shapes are calibrated to use a fixed coarse attenuator setting set at ref_pwr level and a variable fine attenuator setting, as used in bio-pack sequences. The returned shape structure contains parameters required for .RF shape format: shape.pwr (coarse power), shape.pwrf (fine power), shape.pw (pulseduration). No shapefile is produced in the output.

Chapter 14 3D Projection Reconstruction Experiments

A number of BioPack pulse sequences have been coded to permit “linked” acquisition of indirect dimensions to permit rapid 3D experiments. These have a “pra” parameter. If the value of “pra” is zero or 90 degrees, a normal F1F3 or F2F3 2D experiment will be performed.

If “pra” is between 0 and 90 degrees, the t1 and t2 evolution times are co-incremented. Under favorable circumstances it is possible that the chemical shifts characterized by F1, F2 and F3 for each residue may be determined from as few as one projection (or “tilted”) 2D plane along with the orthogonal F1F3 and F2F3 2D planes. More typically it may require several tilted planes.

Any of the above mentioned experiments may be run just by setting “pra”. However, the best projection angles needed are not known in advance. Several different 2D experiments using different “pra” values could be set up and run manually, but this may result in more time used than optimal or necessary. For this reason, BioPack has an interface to manage the optimal collection of PR experiments.

This management is done via the “Proj.Reconstr.” page in the “Acquire” folder in the VnmrJ interface. This page will be visible for any sequence using the “pra” variable (see Figure 42).

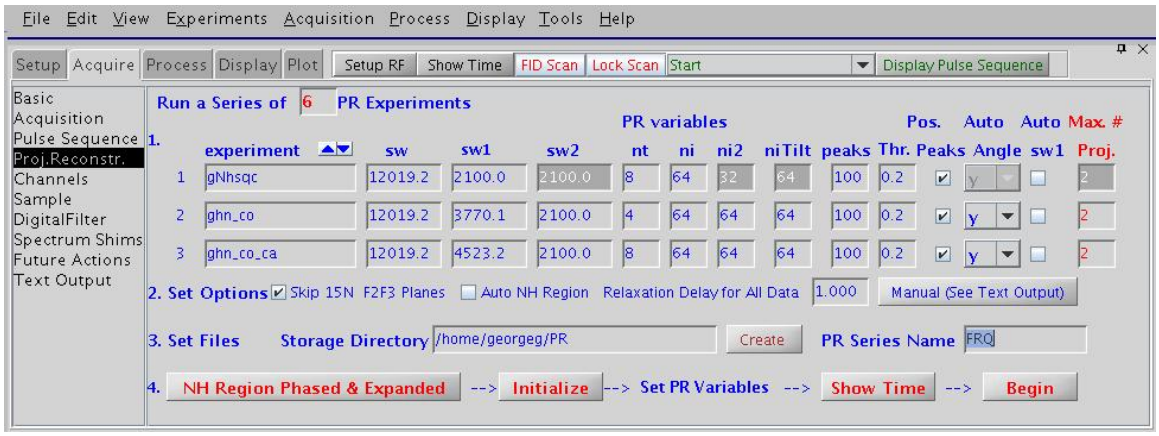


Figure 42 Projection Reconstruction Acquisition Control Panel

The variables visible in this interface are all “global” parameters, not those in the current parameter set. The current experiment’s parameters will be overwritten when the PR queue begins. However, if the current parameter set has a first increment ghn_co data set the spectrum can be referenced, phased and expanded to show only the NH’s. The values of sp/wp and rp/lp will be set into global variables by the button at the lower left of the panel. This is an important step because only signals within this window will be analyzed, and no autophasing is done for any PR data. Rather, the phase settings for the ghn_co data set will be used, via the stored global variables.

The purpose of the panel is to specify the total number of pulse sequences to be used , the names of these sequences and the values controlling the 2D acquisitions. After specifying the names of the experiments, the “Initialize” button runs a macro that sequentially runs the setup macros for each of the specified pulse sequences (although not a PR experiment, gNhsqc may be used as the first experiment to acquire a high S/N and complete NH 2D experiment. If the “skip F2F3” option is used, this data set serves for obtaining the NH 2D chemical shifts). After each setup macro is run, the values of the various spectral widths are set into the PR global variables visible in the panel. Because these are “global” they are not overwritten by recalling a parameter set.

The user is now free to modify the parameter values as desired. The number of transients is set relative to the intrinsic sensitivity of the sequence specified. A larger number than normal for ni and ni2 may be used for better resolution since these are 2D acquisitions for the orthogonal planes and each orthogonal plane is only acquired once, as opposed to the possibly several tilted planes.

Analysis is most reliable if there is a reasonable estimate of the number of peaks to be expected for each experiment. This is a function of the sequence and number of amino acids in the protein. To be safe, a somewhat larger number is specified than necessary, but not smaller than the number of expected peaks.

The Threshold parameter is used to prevent analysis of “noise” or “artifact” peaks. Since most backbone assignment data have peaks with a limited range of intensities, there is

little loss of information by setting the threshold to 0.2-0.4. If S/N is high, the threshold can be lower. If all peaks for a given experiment should be positive, the “Positive Only” checkbox can be selected. This will make the analysis more robust.

The “Show Time” button is used to show the minimum and maximum time needed for the requested experiments. Recall that at least one (F1F3) orthogonal plane is acquired, along with one or more experiments where “pra” is between 0 and 90 degrees. If the Automatic Angle option is used, the data is analyzed after the last orthogonal plane acquisition and after each tilted plane acquisition to obtain the most informative value of “pra” for the next tilted plane acquisition. This results in the most efficient use of time.

Once the values of ni/ni2/nt are re-adjusted to fit the total available time, the “Begin” button is used to start the queued experiments.

All acquisitions occur in the “current experiment”. Data are stored in the specified storage directory under subfiles named for the different pulse sequences used. Acquisition is controlled by the normal “wexp” variable. Experiments may be interrupted by aborting acquisition in the normal manner, but the current 2D data is then lost. When the process is desired to resume, the normal “au” command will restart the acquisition with those parameters in the “current experiment”. For this reason, no local variables should be changed before restarting.

Note that there is a “Manual (See Text Output)” button. This produces a set of instructions for setting up a PR queue. The manual lists and defines all the global PR variables (See Figure 43). After using this button, select the “Text Output” page.

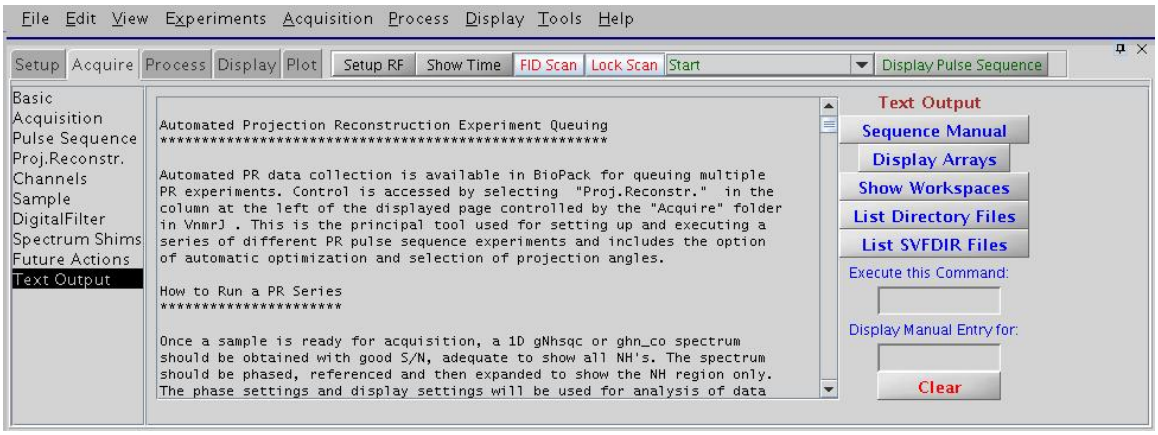


Figure 43 Projection Reconstruction Manual Display

14.1 How to Run a PR Series

Once a sample is ready for acquisition, a 1D ghn_co spectrum is obtained with good S/N, adequate to show all NH's. The spectrum is phased, referenced and then expanded to show the NH region only. The phase settings and display settings will be used for analysis of data from the PR experiments.

The number of PR experiments desired is entered in the top entry box and then their names in the "experiment" column, scrolling if more than 3 sequences are to be used.

The "NH Region Phased and Expanded" button is clicked to memorize the phasing and display parameters in global variables. The "Initialize" button is used to populate the parameter settings. Any of these values can be modified as desired at this time, including the relaxation delay used for all experiments, whether to collect F2F3 planes for each as opposed to using data from gNhsqc or ghn_co (if the first experiment in the list) and whether to have the software automatically select the NH region of the first experiment for further analysis.

The spectral widths in F1 and F2 for 13C and 15N are matched to the actual spread of frequencies for these nuclei in order to have the highest possible resolution for the chosen ni, ni2 and niTilt. This permits the most accurate analysis of peak. The options are Auto Angle Select, Peak Threshold, Number of Peaks and Positive Peaks. The Maximum Number of Angles is set relative to the total time available.

The full pathname of the storage directory and the name of the directory used for this series is then set. The "Show Time" button is used to obtain an estimate of the total time. This step also checks for too many increments in constant time experiments and will display output in the text window if the maximum number of increments is exceeded for any of the experiments, assuming that the pulse sequence code has been written to show these messages.

If this time is acceptable, "Begin" is selected, otherwise parameters are adjusted to meet the requirements. The data is stored in the specified storage directory. This directory has subdirectories named for the sequences used. These subdirectories have the stored fids and analysis output in the form of text files. While the experiments are running, histograms are displayed showing the optimum angle used for the next experiment for the best chance of resolving overlapped peaks.

14.1.1 Pulse sequences

The following BioPack sequences have the option for Projection Reconstruction and have a 15N Semi-Constant-Time option – `SCT = 'y':`

ghn_co, ghn_coA, ghn_ca, ghn_caA, ghn_co_ca, ghn_co_caA, ghn_ca_co, ghn_ca_coA, gcbca_co_nh, gcbca_co_nhA, gcbca_nh, ghc_co_nh, ghc_co_nhA, ghn_cacb, ghn_cacbA.

This permits collection of much higher resolution data with more accurate peak picking and reduction of overlap.

Chapter 15 4D Projection Reconstruction Experiments

A set of of BioPack pulse sequences have been submitted by Ron Venters, Brian Coggins and Pei Zhou of Duke University to permit “linked” acquisition of indirect dimensions allowing for rapid acquisition of 4D PR experiments (JACS 127(24), 8785-8795 (2005)). These have angle parameters similar to the “pra” parameter described above for 3D experiments. Since there is one extra dimension there are two “angles” for converting a 4D sequence into a 2D version. Additionally, there are several 4D -> 3D sequences in which case only one angle is needed. If the value of either of these angles is zero or 90 degrees, an orthogonal 2D projection will be collected. If the projection angles are between 0 and 90 degrees, the t1, t2 and t3 evolution times are co-incremented. Because of the higher dimensionality, the number of tilted planes required to remove all ambiquity is typically larger than in the 3D versions.

Any of the above mentioned experiments may be run simply by manually entering the projection angles, except that certain combinations are physically impossible. The permissible projection angles needed are not known in advance, but entering two angles will result in the automatic determination of the third angle. These experiments have parameter sets in which the best angle combinations have already been determined. The user just selects the number of tilted planes and this sets up the list of permissible angles. BioPack has a dedicated interface to manage the optimal collection of these 4D PR experiments.

The Acquire/Acquisition page is optimized for these experiments (see Figure 44). The box at the left shows the detection dimension (usually NH protons). The “PR Acquisition” boxes show the active number of increments and spectral window for the 2D experiment. The spectral windows for orthogonal planes are entered (swCa and swCb in this example) along with the angles. The “Spectral width” (sw1) value is actually calculated from the choices made for the individual spectral windows and for the input angles to make sure that all signals will appear at the correct (non-folded) positions. Thus the sw1 value will change for different combinations of tilt angles.

The NH orthogonal plane is controlled by the swN parameter and the “increments in t1”

value. All of the angles and the sw1 value are set by operator choice in the “Acquisition Mode” section at the lower right. When one of these is selected, the values of sw1, phase, phase2 and the angles are set. The sequences are named PR42… or PR43…,indicating 4D -> 2D or 4D -> 3D PR experiments.

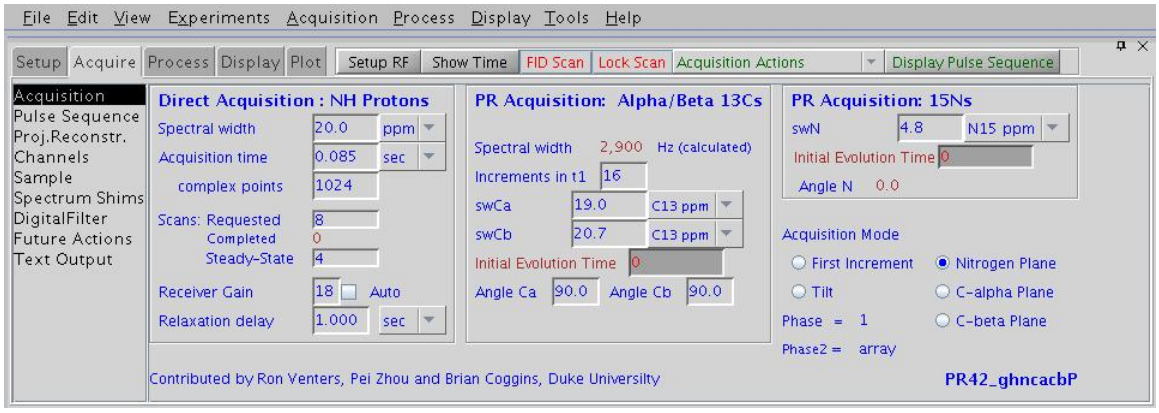


Figure 44 Acquisition Panel for the PR42_ghncacBP Experiment

Management of acquisition of multiple 2D experiments with different tilt angles, suitable for spectral reconstruction, is done via the “Proj.Reconstr.” page in the “Acquire” folder in the VnmrJ interface. This page will be visible for any of these experiments (see Figure 45).

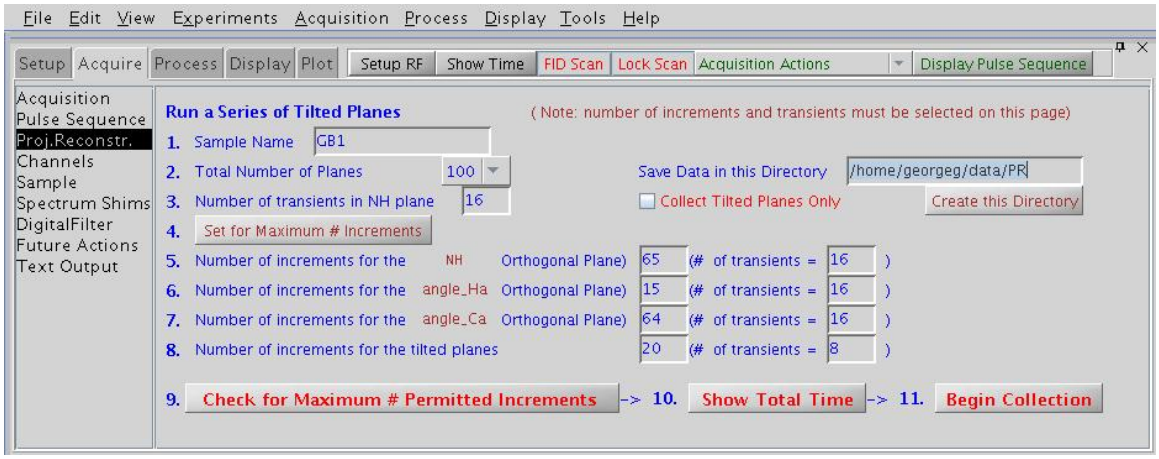


Figure 45 Projection Reconstruction Acquisition Control Panel for PR42_ghacanhP

The variables visible in this interface are all “local” parameters, not “globals” as in the 3D PR panel. These variables are created, if needed, by a macro that runs as part of the experiment setup macro. The first experiment’s nt, sw1, ni and angle parameters will be set when the PR queue begins. Whereas, the above 3D PR queueing was for multiple pulse sequences, this queueing is for different angle combinations for the same pulse sequence.

After specifying the sample name and storage directory, the user uses a menu to select a total number of tilted planes to acquire (6,8,10,12,20,30,40 and 100 are the options). Then the number of transients used for the NH orthogonal plane is set. The number of transients should be set relative to the intrinsic sensitivity of the sequence specified and the concentration of the sample. The number of transients for the other orthogonal planes and all tilted plane acquisitions are set relative to the expected sensitivities.

A button is provided (“Set for Maximum # Increments”) that runs a macro to set the maximum permissible number of increments for the orthogonal and tilted planes. These values are placed in the entry boxes on lines 5 through 8 (Figure 45). These maxima are determined by the presence or absence of a constant time evolution period (If the output number is 1000, there is no constant time period and the number may be set as desired).

The natures of the orthogonal planes are indicated by a text message (in red) given by the angle name. The number of transients for each orthogonal and tilted plane is also enterable.

After setting these parameters, the button “Check for Maximum # Permitted Increments” will run a macro that calculates the maximum number permissible. It is important, of course, to check the total time with the “Show Total Time” button. In this case, as opposed to the 3D case with automatic angle selection, the time is deterministic.

If the time is within the desired allotted time,the “Begin” button starts acquisition. The data directory is used to store the data as acquired. 2D data processing is provided within VnmrJ for all projection planes using the processing panels, but Projection Reconstruction data processing is be done with the Duke program PRCALC.

15.1 Projection Reconstruction using PRCALC

Projection Reconstruction software may be obtained directly from the Duke NMR group by using their web site (http://zhoulab.biochem.duke.edu/software/pr-calc). Their software (PRCALC) has been described in the literature (B.Coggins and P. Zhou, J.Biomolecular NMR, 34, 179-195 (2005)) and is usable via NMRpipe software (http://www.onemoonscientific.com).

Chapter 16 Non-Linear Sampling (NLS)

Normal multi-dimensional data sets are collected using regular (linear) incrementation of (an) evolution time(s) because a “Fast Fourier Transform” algorithm is used for processing. The multidimensional transform is realized by sequential application of a 1D FFT. Other processing methods have become available (Multidimensional Fourier Transform, Maximum Entropy, Multi-Dimensional Decomposition, etc.) that are able to process non-linearly sampled data. This can represent a significant timesaving in acquisition if only 5-50% of the data needs to be sampled and the data can be accurately processed.

The flexibility of the pulse programming language of VnmrJ has allowed a straightforward and generic implementation of acquisition of NLS data. In particular, the evolution time(s) for a specific fid are looked up from a text file, rather than being calculated from ni(2) and sw(2). All that is necessary is for the sequence to be modified to replace the internally calculated evolution time(s) with ones from a text file.

BioPack has all the necessary files to make this process easy to use. All sequences have a “Options” (formerly “DigitalFilter”) page in the Acquire folder. This has a menu to facilitate setting up an NLS experiment (see Figure 46)

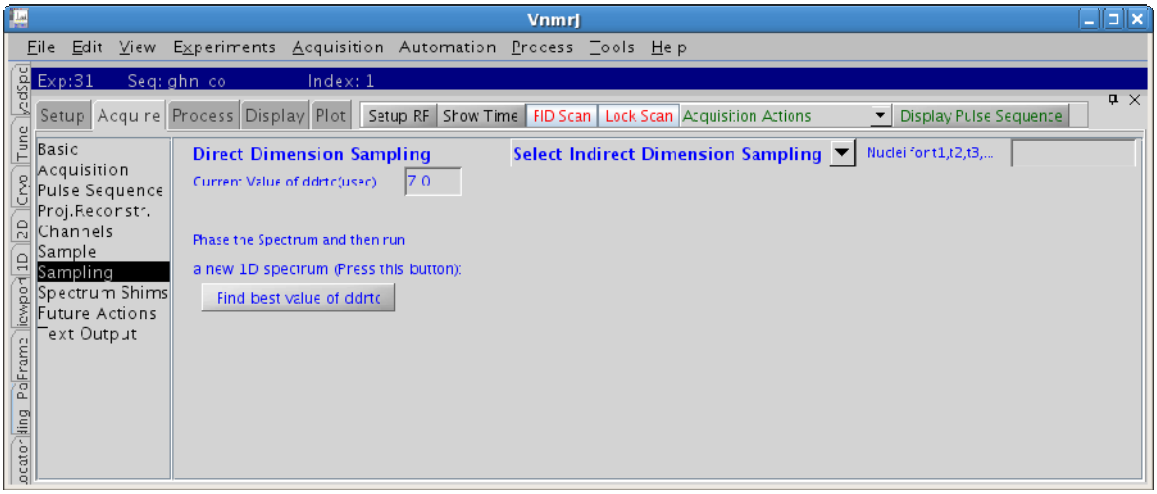


Figure 46 The Sampling Page

The menu has four options: (1) Non-Linear (Sparse) Sampling via a method contributed by Vladislav Orekhov and Victor Jaravine (Swedish NMR Center, Gothenburg, Sweden), (2) Non-Linear (Explicit) Sampling via a method contributed by Krzysztof Kazimierczuk, Anna Zawadzka and Wiktor Kozminski (Warsaw University), (3) Automatic Spectral Compression contributed by Lescop and Brutscher, and (4) Linear (Normal) Sampling. The Linear (Normal) Sampling option just renames “seqfil” to the non-NLS version.

Necessary files are present in BioPack to convert an existing pulse sequence into a NLS version. The menu selection runs a macro that checks to see if the NLS version of the sequence is present as a compiled version. If it is not present, the current pulse sequence code is copied into the user's ~/vnmrsys/psglib under the same name, but with "_S" or "_TAB" appended ("S" for Sparse and "TAB" for Table). A script is run to modify the code and the new sequence is compiled. The macro also creates a ~/vnmrsys/templates/layout file under the new sequence name that has all the same files as the original sequence. Therefore, the panels visible will lookIdentical. The Acquire folder now has a "Sampling/NLS" page visible for the new sequence (see Figure 47 and Figure 48).

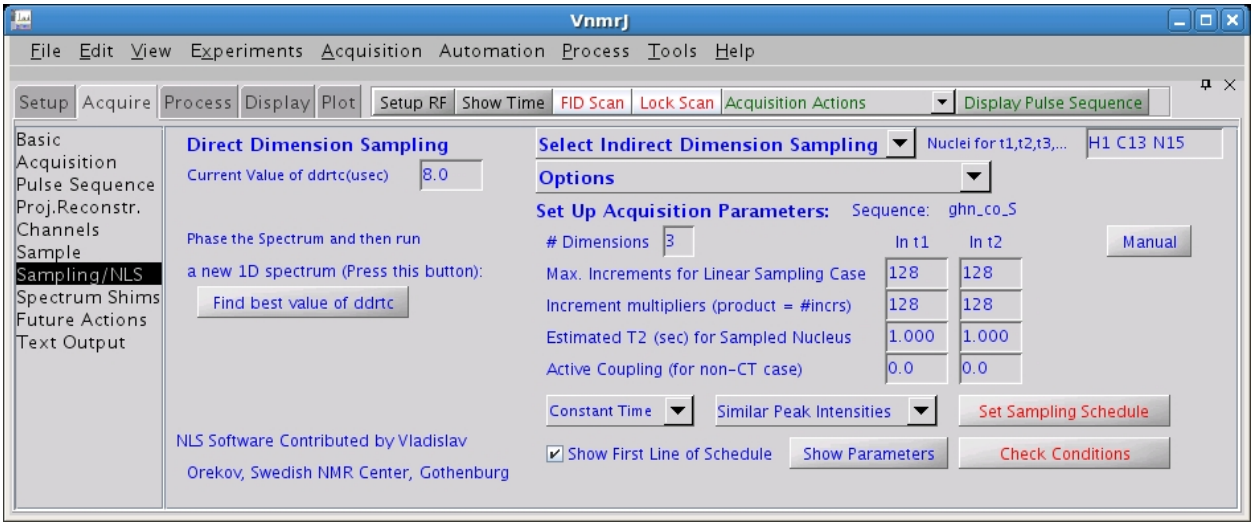


Figure 47 The Sampling/NLS Page for the "Orekhov" Method

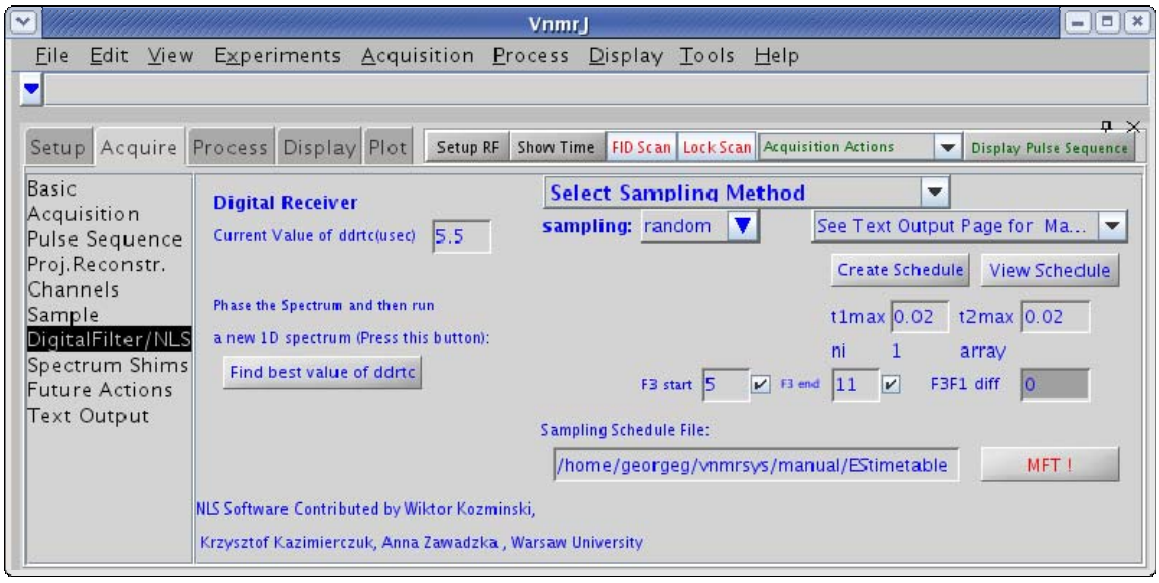


Figure 48 The Sampling/NLS Page for the "Kozminski" Method

16.1 The Orekhov Method

The changes in the pulse sequence do not affect the original operation of the sequence and it can be run as before by setting SPARSE='n' (the parameter SPARSE is created, if necessary, by the BPrtppar macro that runs in the experiment setup macro). The execution of the code is different for SPARSE='y', however, in that a text file containing the evolution times is necessary.

The above page is used to set the acquisition parameters for the multi-dimensional experiment. The number of dimensions should be set for the total dimensionality of the experiment (e.g. "3" for ghn_co_S). The maximum number of increments (nimax, ni2max or ni3max) would be that for a normal linearly sampled data set (these determine the real resolution in the indirect dimensions). The "increment multipliers" (stored as ni, ni2 and ni3) would be the reduced number. Please note that for 3D NLS acquisition the only purpose of these values is to define the total number of actually acquired FID's and only the product of these numbers is important. For example, combinations 32, 32 and 16, 64 give exactly the same sampling schedule in Fig. 47.

The "Estimated T2…" should be set relative to the estimated T2's of the signals in the specified evolutions. This permits the software to make a more intelligent choice of which evolution times to select. In non constant-time evolution times, signals may go through a null because of an active coupling. In these cases the coupling information can also permit the software to make a more intelligent choice of which evolution times to select.

A menu is provided to specify which evolution times have a constant-time character. This information is not used for acquisition, but to notify the MDD processing software that mirror imaging of the time domain signal is possible.

Similarly, a menu is provided to specify if similar intensity cross-peaks are to be expected (cexp='y' - R-MDD processing would be used) or if the cross-peaks are of widely differing amplitude (cexp='n' - MDD processing would be used). The former case would prevail for experiments such as ghn_co, while the latter case would be appropriate for the NOE plane of gnoesyNhsqc. This information is later used by MDD processing and may have important effect on the sampling scheme. It affects the checking and corrections of "holes" in the sampling scheme, which are situations when a certain row or column in the sampling scheme matrix has no or too few data measurements. The uncorrected holes lead to serious problems during MDD processing. In case of cexp='y', the check is more relaxed, which means that data recorded with cexp='n' can be later processed using both cexp='n' and 'y', but data recorded with cexp='y' can only be processed with cexp='y'. On the other hand, cexp='n' typically requires significantly more data points in NLS schedule, i.e. sparse levels >=10-20%. Note that a cexp value is needed for each dimension, i.e. for ghn_co_S cexp='nyn' (for t1, t2 and t3). The final value of the cexp text string is always 'n'.

Once the parameters have been entered the "Set Sampling Schedule" button runs a script to generate the necessary subfiles including the text file specifying the evolution times. At this time information is displayable in the Text Output window. A full NLS parameter list is also generated in the Text Output window when the "All parameters" button is used.

The "Check Conditions" button runs "go('check')" to check for error conditions such as too many increments in an indirect dimension (normally limited in constant-time evolutions). The TextOutput page displays any error messages. The error condition checking would give the same error for the non-NLS version and its nature depends on how the pulse program was coded, not the NLS part of the sequence. If the number of increments (nimax, ni2max or ni3max) in a specific evolution period is too large, reduce the number until no error results when using the "Check Conditions" button.

At this time the experiment can be started normally. Data will be acquired and stored as in any other experiment. Data may be processed normally using wft and wft1da, but note that the first increment is usually not for d2=0, and therefore data may be modulated. Use "dps" to verify the actual evolution time. While wft1da may be used to examine the entire data set, do not use wft2da. 2D processing will require NLS-compatible software.

Use the macro BPsvf (normal BioPack "Save Fid" button", not the command "svf") to save the data. Apart from the FID, the macro also saves the sampling schedule and other important files.

Reference material for learning about the NLS schedule and data processing can be found at:

Orekhov, V. Y.; Ibraghimov, I.; Billeter, M., Optimizing resolution in multidimensional NMR by three-way decomposition. *J. Biomol. NMR* **2003**, 27, 165-173.

Tugarinov, V.; Kay, L. E.; Ibraghimov, I.; Orekhov, V. Y., High-resolution four-dimensional H-1-C-13 NOE spectroscopy using methyl-TROSY, sparse data acquisition, and multidimensional decomposition. J. Am. Chem. Soc. **2005**, 127, 2767-2775.

Jaravine V, Ibraghimov I, Orekhov VY. Removal of a time barrier for high-resolution multi-dimensional NMR spectroscopy. *Nature Methods*, **2006**, 3: 605-607.

16.1.1 NLS Parameters, Macros, Files, and Commands

SPARSE is a flag used by the pulse sequence in order to run a normal linearly sampled ('n') experiment or an NLS experiment ('y'). SPARSE is set to 'y' when using the button in the "Options" page.

nimax, ni2max and ni3max define final dimensions of the spectrum (this would be the numbers for the desired resolution in the linear sampling case).

ni, ni2 and ni3 affect only the number of actually recorded FIDs and consequently the total measurement time (use the "time" command as usual). The values are not used by the pulse sequence if SPARSE='y'. While ni is used for a 2D NLS experiment, only the product nixni2(xni3) is used for 3D (or 4D) experiments to determine the number of FIDs to be acquired. Therefore, specify the desired resolution by setting the nimax parameters.

There are two parameters that affect the sampling schedule:

T2sp - estimated T2 time. Use large value in case of CT evolution (e.g. 1 second).

Jsp - estimated J-coupling constant, e.g. in case of 13C evolution, e.g. in non-constant time HNCA it is 35 Hz. Use Jsp=0 if the coupling is not relevant.

The proper value of the "cexp" parameter should be set depending on the nature of the sequence (see above).

The macro BP_NLSset generates the sampling schedule (via the button).

The "All Parameters" button runs the macro "dgnls" which shows parameters and their values in the Text Output window.

Text files nls.hdr_3 and nls.in are automatically created in the current experiment. These store the sampling schedule and an input file for NLS processing.

The processing of the sparse data must be performed by using non-FT software.

Files:

bin-

init_SP.gawk, runspheader.sh, spheader, spheaderLNX,

spheaderMAC, spheaderSUN, split_dims.sh

maclib-

BP_NLSinit	a macro to convert a pulse sequence to NLS capability
BP_NLSset	a macro to set up a sampling schedule
BP_NLScheck	a macro to run go('check') for ni*max values
BP_NLSstop	a macro to switch back to parent sequence
BPsvf	saves NLS text files from current exp along with fid, etc.
BPrtppar	creates SPARSE flag (BPrtppar is called in BioPack experiment setup macros
dgnls	a macro to display NLS parameters in text window

16.1.2 Command Line Operation

1. Use experiment setup macro (e.g. ghn_co_ca)
2. Enter BP_NLSinit
3. Enter parameters (use dgnls to view)
4. Enter BP_NLSset to create sampling schedule
5. Enter BP_NLScheck to check limits on increments
6. Acquire data with "go"
7. Save fid with BPsvf(name)

16.1.3 Description of the Algorithm for the Generation of the NLS Schedule

For the case of a three-dimensional spectrum the schedule matches a two-dimensional probability density function for the indirect evolution dimensions. The function is defined on a two-dimensional grid (t1, t2) determined by spectral widths and maximal acquisition times (t1max, t2max) in the two dimensions. Sampling on a grid, which we define as "sparse sampling", is a special case of a more general non-uniform sampling that allows sampling at arbitrary selected time points. The distribution is obtained as a product of the two envelopes, P(t1,t2) = P1(t1) x P2(t2). The envelope functions P1(t1) and P2(t2) are devised to match the signal coherences in the indirect dimensions for a particular system and experiment. Currently three possibilities are implemented:

mono-exponential relaxation: P (t1) = exp(-t/T2)

modulation by the one-bond J-coupling: P(t2) = cos(t pi/J)

The J-modulation can be combined with the relaxation decay

The transverse relaxation time T2 and value of the J coupling are parameters of the procedure. For a given probability distribution, we use the following procedure to generate the NUS schedules. First, a pair of integer indices is randomly selected that corresponds to the acquisition times (t1, t2). Then the pair is added to the sampling schedule table if the corresponding value of the probability distribution P(t1,t2) is larger than another randomly generated number ranging between 0 and 1; otherwise, the index pair is discarded. This process is repeated until the sampling table contains the requested number of data points. Thus, a NLS schedule is a table of evolution delays (t1, t2) spanning maximal acquisition times and spectral widths in the indirect dimensions.

runspheader.sh, a csh script to generate NLS

The non-uniform sampling scheme is produced by the runspheader.sh script, which takes all needed parameters from an input file (sequence name appended by ".in",e.g. ghn_co_S.in, present in the current experiment)

For example, runspheader.sh ghn_co_S.in

The function of the runspheader.sh script is to call the program spheader with appropriate parameters. Normally, the input file is produced by a spectrometer program (e.g. BioPack), which also starts runspheader.sh . However the script can be run directly from

a Unix shell window, provided the input file is prepared (see below). The script produces several header files with names (defined by the file parameter present in the input file) with extensions .hdr_X X=1-5. The file nls.hdr_3 is the most relevant. It contains "t1 t2 [t3...]" indexes of time evolution points in the indirect spectral dimensions. If needed, the evolution delays can be produced by multiplying these indexes by the corresponding dwell times (i.e. 1/sw1).

Description of the input file format - input for runspheader.sh

The file contains lines started by keywords followed by a list of parameters values. All the keywords given below are mandatory.

Below, several examples of input files are presented. The examples can be found in the mddnmr software version 1.3 (Sep 2006).

3D NOESY-HSQC

160 x 44 complex points in 1H(NOE) x 15N dimensions. Ratio NI/NIMAX=45/160 gives ~ 28% sparse level

(see example 3, azurin3dNOESY/az15n2_5%2_nohsqc_3d/az15n2.in)

file az15n2	this is root name for all header files
NDIM 3	dimensionality of the experiment
seed 4321	seed for random number generator.
SPARSE n	reserved parameter for spectrometer software
sptype shuffle	reserved parameter
f180 nnn	180 degree flag for corresponding dimension if evolution time is started at half a dwell time.
CT_SP nnn	constant time flag. MUST be nnn for NOESY-HSQC
CEXP nnn	must be nnn for NOESY-HSQC
NIMAX 160 44 1	final time domain dimensions (1=direct dimension)
NIMIN 0 0 0	reserved parameter
NI 45 44 1	the product of the numbers relative to the. NIMAX product. It determines the sparse level as s=(NI1*NI2)/(NI1max*NI2max) IMPORTANT NOTE: only first numbers in NIMAX and NI can be different for NOESY-HSQC
SW 8000.0 2000 8000.0	Spectral widths for all dimensions
T2 0.05 0.1 1	estimated T2s for indirect dimensions

```
Jsp   0 0 0                    one-bond coupling.

3D HNCO
(see example 2, ubidemo/ubi_ghnco_3d_600Y.fid/ghnco.in)

file  ubi_ghnco_3d_600Y
NDIM   3                      3 dimensions:  C' N  Hn
seed   54321
SPARSE n                       y/n toggles NUS/uniform mode
sptype shuffle                 modes of NUS - reg/shuffle/over (default is
                               shuffle); shuffle=matched no repeats;
                               over=matched with repeats; reg=uniform
f180   nnn                     180 degree flag for corresponding dimension
                               if evolution time is started at half dwell time
CT_SP  nyn                     CT evolution in 15N
CEXP   yyn                     R-MDD mode for 13C and 15N dimensions
NIMAX  64 32 1
NIMIN  0  0  0
NI     7  32 1                 sparse level 11 percent  (7*32/(64*32))
SW     1500.0 1499.981 12001.2
T2     0.05 1 1                T2=50ms for 13C and 1s for 15N
                              (CT implies no decay in evolution)
Jsp    0 0 0
```

16.1.4 Structure of the NLS (nls.hdr_3) Table

The NLS table is an ASCII file, where every line contains integer increments corresponding to evolution times for all indirect dimensions. The number of entries in the table corresponds to a complete uniformly sampled experiment with dimensions NI x NI2 [x NI3 ...] for t1, t2 [,t3 ...]. However, values of the indexes in the table vary in the range from zero to NImax, NI2max,[NI3max ...], respectively. Thus, the scheme is generically

applicable to any existing pulse sequence. Uniformly incremented indexes in the pulse sequence must be substituted by the values from the NLS table for every new FID. This means that the original values of the evolution incremental delays are not used. New evolution delays are produced by multiplying these indexes by corresponding dwell time (i.e. 1/sw1).

16.1.5 Implementation in BioPack

The BioPack implementation features automatic creation of a new NLS version of any multi-dimensional pulse sequence by use of a button in the DigitalFilter page in the Acquire Folder of VnmrJ. After specifying the number of increments, etc., a single button

is used to generate all the NLS files including the table of increments (nls.hdr_3) via the "Set Sampling Schedule" button in the same page. Acquisition is performed in the normal manner. Saving of the data using the BioPack macro "BPsvf" also saves the NLS files and a script to permit easy processing by the MDD software.

16.1.6 Pre-processing of NLS Spectra using VNMRJ software

The “Options” menu on the Sampling/NLS page in the Acquire folder has an option for converting VnmrJ data into a format suitable for MDD processing. In this case, the Sampling/NLS page is changed to a form illustrated in Figure 49. The data may also be MDD-processed using user-installed MDD software so that the resulting data may be processed with VnmrJ using standard software (see section 16.1.7 for instructions on how to obtain MDD software).

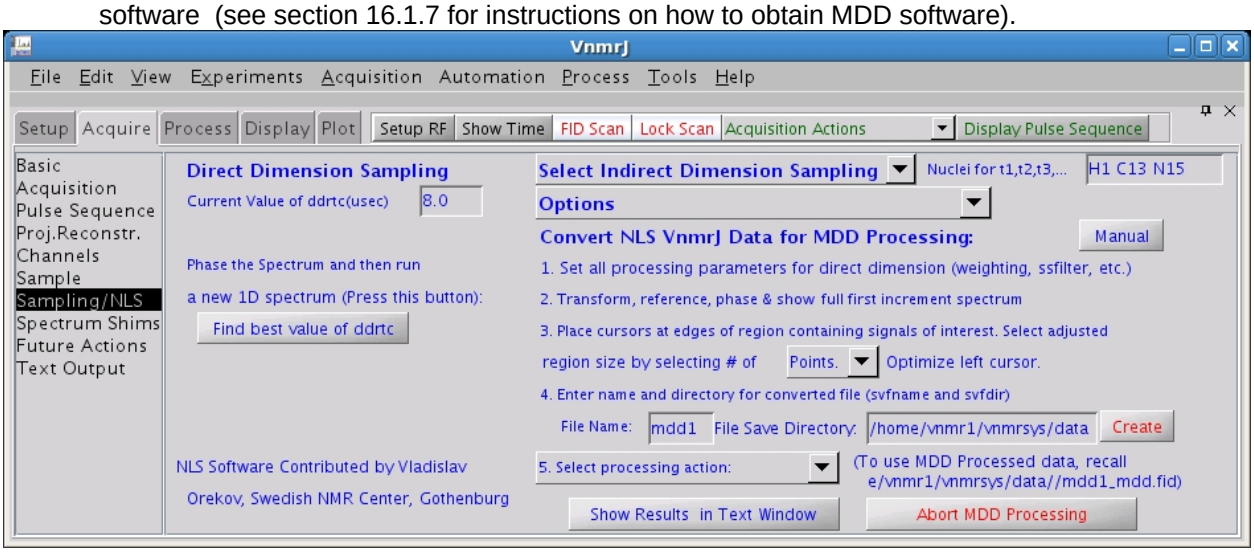


Figure 49 The Sampling/NLS Page for Converting VnmrJ Data for MDD Processing

The following list of instructions details the process:

From VNMRJ:

1. FT the first trace by wft(1). Phase and reference the spectrum and adjust the windowing parameters, solvent suppression filter, etc.
2. Choose the region of interest, i.e. if the experiment is NH-detected then set the cursors around the NH region containing all the NH proton signals (if there is little signal in the first increment use cursor positions determined from a more sensitive experiment for the same resonances. The "Points" menu permits selection of a power of 2. After using the menu the cursor positions will change to reflect the chosen number of points. Otherwise,the cursors determine the selected region. The "Points" menu uses the BP_select_mdd_region macro. Once the region size is selected, the left cursor may be moved to center the resonances.
3. Enter the path name of the directory into which the new MDD-compatible data are saved along with the desired new file name for that file. The standard VnmrJ global variables "svfname" and "svfdir" are used for these. If the directory does not exist, a button will create it.

4. Select the type of data and if MDD processing is to be done.For no sensitivity-enhancement in indirect dimensions use "States. If the indirect dimension uses the gradient echo-antiecho (sensitivity-enhanced) method (often used for the N15 dimension) use the sensitivity-enhanced option.

   The macros BP_v2mdd('filename') and BP_v2mdd('filename','se'), respectively, are used for the first two options in the menu. If MDD processing is desired automatically, use the final two options which run the macro BP_mdd2v. The data is first converted to an MDD-readable format and then the (user-installed MDD program) is used to produce subfiles within the target fid file. This may take from minutes to hours depending on the size of the data set. The target fid file has a directory "MDD" that is populated during the calculation. When this file stops being populated theprocess is finished.

   To use VnmrJ to process the results of the MDD calculation, a new fid file in VJ-compatible format will be created (if the MDD processing is specified). This file will have the same name as entered above, but it will have an "_mdd" appended. Normal VnmrJ processing parameters and commands can be used to produce the nD spectrum. Note that the ni/ni2 parameters have been set to the nimax/ni2max values specfied when the initial data was collected. Use weighting functions appropriate for these sizes.

5. To view the log file generated during the conversion process, use the "Show Results of Data Conversion in Text Window" button. This log file is present in the *fid file generated by the conversion process. The original fid file is also saved in the same directory.

From a Shell

The following steps need to be taken to convert the Varian data into the format that is directly accessible by the mdd software.

In VnmrJ:

1. FT the first trace by wft(1). Phase and reference the spectrum and adjust the windowing parameters, solvent suppression filter, etc. ;
2. FT the first trace by wft(1) and choose the region of interest, i.e. if the experiment is NH-detected then set the cursors around the NH region containing all the NH proton signals (if there is little signal in the first increment use cursor positions determined from a more sensitiveexperiment for the same resonances.
3. FT all traces (wft) and "flush" the data (run the command "flush") ;

In a shell:

4. Use the rsw command to cut out the region of interest, e.g.

   ```
   > rsw exp20 filename.fid 810 1024
   ```

   exp20 is the experiment containing the FT'd and flushed data, filename.fid is the destination filename, 810 is the chosen starting point of the region of interest 1024 is the size of the region of interest; The filename.fid can be inspected in VnmrJ using the ft('noft') and ddff(1) commands.

```
> rsw exp20 fname.fid 810 1024 -t
```

The -t option allows saving the data in the time domain (FID) format. Inverse FT is applied and the processing parameters in the directly detected dimension are disabled.

5. For sensitivity-enhanced data in N-15, use the proc_se command to convert the data into the conventional format that can be used for Fourier transform:

```
> proc_se filename.fid
```

The filename.fid can be inspected in Vnmr(J) using the ft('noft') and ddff(1) commands.

6. Convert the Varian data file into a format that can be read by mdd directly:

```
> BP_v2mdd filename.fid
```

7. Copy the mdd parameter files from exp20 to the fname.fid directory:

```
> cp ~/vnmrsys/exp20/*.hdr_3 filename.fid/.
> cp ~/vnmrsys/exp20/*.in filename.fid/.
```

The filename.fid file now should contain the fid.v2m binary file that can be read directly by the mdd software.

Reducing Data Size:

The BP_rsw macro (in VnmrJ) runs BP_select_mdd_region and the program "rsw". It is used to preserve the region of interest and thus reduce the size of an nD data set. The data are first processed in the directly detected dimension in VnmrJ and then saved by rsw as partially transformed data set for further processing. Use this macro as an independent option. It is not necessary for normal use.

Sensitivity-Enhanced Data:

`proc_se` - converts a sensitivity enhanced data set into the conventional format.

Usage:

```
> proc_se filename.fid
```

In special cases the loop size can be passed as an argument. For instance, if array = 'phase2,phase,phase3' and the sensitivity enhanced dimension is associated with phase2parameter, the following command should be executed:

```
> proc_se fname.fid 8
```

where 8 is the loop size in this particular case. If the array = 'phase2,phase' the loop size is 4.

The default loop size is 2 (array = 'phase,phase2').

16.1.7 Conversion of MDD data for VNMRJ Processing

The “Options” menu on the Sampling/NLS page in the Acquire folder has an option for converting MDD data into a format suitable for VnmrJ processing. When this option is selected, the Sampling/NLS page is as illustrated in Figure 50.

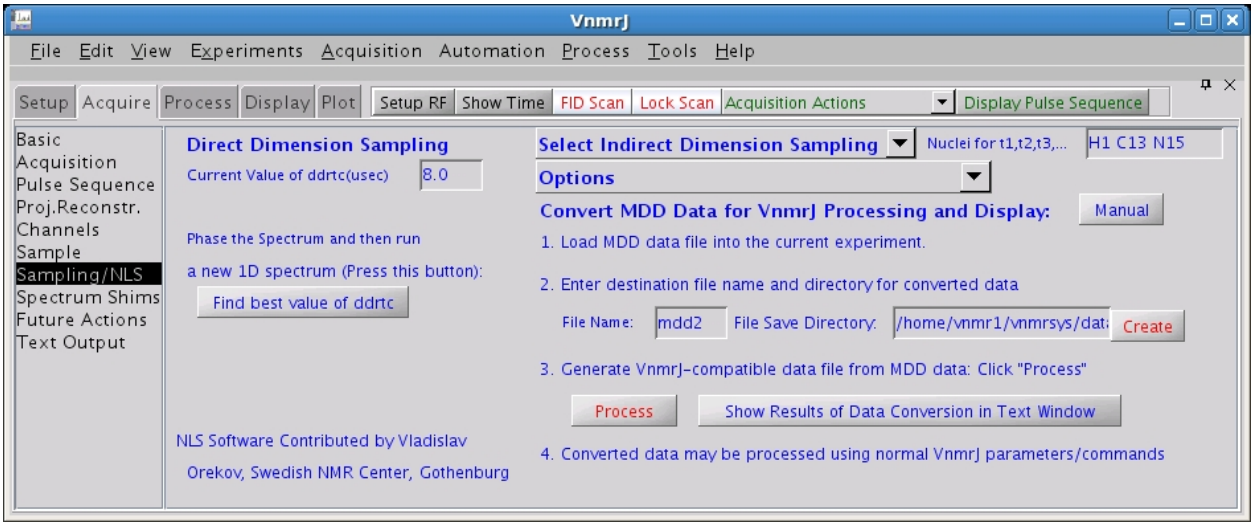


Figure 50 The Sampling/NLS Page for Converting MDD Data for VnmrJ Processing

This page permits use of VnmrJ processing, display and plotting of MDD-processed NLS data acquired using VnmrJ. The converted data is presented as time-domain data with linear t1 and t2 incrementation to t1max and t2max specified by the selection of nimax and ni2max prior to data acquisition. No change in t3 processing parameters is possible, but t1 and t2 processing parameters may be selected as if the data were acquired in a normal linear sampling manner. Normal 3D processing using standard VnmrJ commands and panel "widgets" is used.

From VnmrJ

Select "Convert MDD Data for VnmrJ Processing" from the "Options" menu on the Acquire/Sampling page. To view this page setup the same experiment via the Experiments drop-down menu. Then select the Orekhov method in the "Select Indirect Dimension Sampling" page in the Acquire folder. This willthen show the "Options" menu.

Enter the full path name of the MDD data file and the destination file name and directory for the VnmrJ-compatible data file. Click the "Process" button to produce the file.

Use the ordinary VnmrJ browser,Fuuture Actions Page or command line to retrieve this data file into the current experiment. The ni/ni2 values will reflect the increment numbers used for the "Max. Increments for Linear Sampling Case" when the data were acquired (shown when the "Set//View Acquisition Parameters" option is selected), not the "Increment multipliers"

The processing generates a log file in the VnmrJ fid file and "Show Results of Data Conversion in Text Window" button.

Macros

```
BP_mdd2v('filename.fid','myfid.fid')
```

The mdd-processed data file filename.fid/fid.m2v is converted into a VnmrJ-accessible format and stored in a user specified filename, myfid.fid The mdd2v.log file shows the progress of the processing.

NOTE: In the new file, myfid.fid the processing parameters in the directly detected (np) dimension have been disabled and no zero filling, phasing, referencing or window functions should be used in the subsequent ft3d processing.

All these functions should be applied prior to the VnmrJ to MDD dataconversion. The indirectly detected dimensions are processed as desired.

From a Shell

Convert the mdd data file into a Varian format using the following command:

```
> mdd2v filename.fid myfid.fid
```

The myfid.fid file is now ready for the conventional 3D processing as described above.

16.1.8 Processing of NLS Spectra using MDDNMR software

Processing of spectra recorded using NUS can be performed indepdent of VnmrJ by using mddnmr v1.6 (or higher). Software can be downloaded from http://pc8.nmr.gu.se/~mdd/Downloads/.

It is free to academics and requires "nmrPipe" software if processing is done outside VnmrJ. There is a Google group at http://groups.google.com/group/mddnmr that provides documentation, recent news and advice.

Contact:

Vladislav Y. Orekhov

Associate Professor

Swedish NMR Centre at Gothenburg University

Box 465, Gothenburg

SE 40530, Sweden

E-mail: orov@nmr.gu.se

16.2 The “Kozminski” Method

Selecting “Explicit Sampling” from the menu in the Sampling page executes the macro BP_ESset. This converts existing pulse sequence code to a new version (with “_TAB.c” appended), creates layout files for the VnmrJ interface and creates any new required parameters. Multi-dimensional FT (MFT) processing is possible via a button (see a paper by Kazimierczuk et. al., J. Biomol. NMR, 36, 157-168(2006)).

This method differs from the Orekhov method in that several different types of sampling schedules are selectable from the VnmrJ interface (other types of sampling schedules are possible, but Gaussian or exponential sampling should be used as a default). The method does not require “on grid” sampling (all points in time are multiples of 1/sw1 and 1/sw2). In general, there are all kinds of sampling schemes that use evolution times irrelevant to conventional grid sampling.

Conventional programming of multidimensional pulse sequences uses implicitly incremented evolution delays (i.e. d2, d3, d4). For the i^{th}-dimension the increments of $1/(sw_i)$ are calculated automatically and added in consecutive steps of time domain sampling. Therefore, the user controls the experiment by specifying number of increments in each dimension (ni, ni2, ni3).

In conventional experiments, the maximum evolution time is evaluated as (number of increments -1)/spectral width, while the total number of data points in the indirectly sampled time domains equals the product of the numbers of increments (i.e. ni*ni2*ni3).

Processing of conventional multidimensional NMR spectra is based on series of one dimensional Fourier Transforms. This approach requires time domain points lying on Cartesian grid. This limitation could be omitted by application of a non-FFT method of processing. There are several approaches to this such as Maximum Entropy and Multi-Dimensional Decomposition (MDD). Multidimensional Fourier Transform (MFT) (see above) is another method that is implemented in BioPack and accessible via the Sampling/NLS page in the Acquisition Folder in the VnmrJ interface.

Since MFT can be performed using any sampling scheme it enables one to choose time coordinates for each time domain point and acquire data in any order. The time domain points coordinates are listed explicitly in a text file, and the numbers are normalized to a range between zero and one (this allows using the same file in different experiments).

Note that this approach allows for planning of any sampling pattern, including conventional, radial (used in Projection Reconstruction techniques), spiral and random distributions. Any sampling pattern stored in a table is possible without the need of changing pulse sequence code. It is also possible to prepare such a file independently, storing in each line two numbers from the range 0 – 1 in text format.

The macro "BP_ESset" is executed first. It is run from the command line or by selecting Explicit Sampling in the "Select Sampling Method" menu in "Options" page. It creates all required parameters and produces the pulse sequence code (if necessary). It may rarely happen that the sequence does not convert successfully. In such a case, see the file “manual/ EXPL_samp_seq” (also given below). The necessary modifications of pulse sequence code are straightforward.

The total number of acquired fids is controlled only by ni (note that ni2, and ni3 should be set to 1). The array parameter should be set the same as for conventional experiments, to “phase,phase2”, the array order is important for processing. The path name containing time domain points coordinates must be specified in the string parameter timetab_name, eg.

timetab_name= userdir+'/tablib/EStablename.

The same file is necessary for proper Multidimensional FT processing, so a new sampling schedule should not be created until the data are saved (the current timetab_name file is saved by BPsvf). The general rule is to set the smallest possible value of “nt” to accomplish the basic phase cycle and the highest possible “ni” for the given experimental time.

Real time, rather that constant time evolution should be used. (e.g. for 13C) and semi-constant time (available in many experiments for 15N) instead of constant time.

16.2.1 How to create a time table file

The name of the time table file should be set in the entry box in the "Options" page. Using the same name as an existing file will result in the existing file being overwritten if a new sampling schedule is generated. It is not necessary to generate a new sampling schedule for each experiment, as long as the number of lines in the file is at least equal to the number of increments specified (ni). For example, a very large random distribution time table could be generated and used for all cases of random distribution.

The number of points (ni) should be entered and the sampling type should be chosen. The values of phase and phase 2 should each be set to 1,2 (check the Hypercomplex 2D checkboxes in the F1 and F2 Acquisition panels in the Acquisition page or the Acquire folder). For most types of sampling there are some extra parameters. Available types of sampling are:

radial (points on a few lines)
parameters: phi_0 (angle of first line), d_phi (angular step),
n_phi (number of lines)

spiral (points on two spirals, clockwise and anticlockwise)
parameter: n_rounds (number of rounds of each spiral)

random (uniform distribution in both dimensions)
parameters: none

gauss (gaussian distribution in both dimensions, with maximum at t1/t2=0)
parameter: sigma (the probability is proportional to exp(-t^2/(2sigma^2))

exp (exponential distribution in both dimensions)
parameter: sigma (rate of decay, the probability is proportional to
exp(- t/sigma))

g-r (uniform in first dimension and gaussian in second one)
parameter: sigma (for the first dimension)

e-r (uniform in first dimension and exponential in second one)
parameter: sigma (for the first dimension)

All tables are created using maximum times equal to 1. The sigma parameter should be adjusted to this value. Of course, a table of any set of points can be also created outside VnmrJ, but then maximum times should also be 1.

The button "Create New Schedule" creates the a new sampling schedule using the specified "ni" and sampling method (and any associated parameters) under the name and location specified by the Sampling Schedule File entry box.

The button "View Schedule" displays a plot of time domain data points. It requires installed gnuplot. This popup window must be closed to resume VnmrJ operation.

16.2.2 Starting the experiment

The maximum evolution times should be set by parameters t1max and t2max. t1max and t2max should be set to obtain optimal resolution, however the signal decay due to transverse relaxation should be considered. In order to achieve quadrature detection in F1 and F2, the array parameter should be left as in the conventional experiment, i.e array=phase,phase2 for three dimensional experiments. Thus, in a 3D experiment four fids are acquired for each time domain point. These phase settings control echo-antiecho selection or switching between sine- and cosine-modulated spectra in the States method.

The f2coef 3D parameter should be present in the parameter set. It describes the way of getting frequency signs (quadrature) in the last indirectly detected dimension, and depends on the actual pulse sequence. Standard BioPack parameter sets have f1coef and

f2coef set properly by default. The values are not used in controlling acquisition, but are used in data processing.

For example:

f2coef = '1 0 –1 0 0 -1 0 –1' for echo-antiecho, or

f2coef = '1 0 0 0 0 0 –1 0' for the States method.

The current version of the MFT processing module so far supports only one phase modulation (echo-antiecho) in the last indirectly detected dimension. Quadrature in other dimensions, i.e. F1 in 3D, is assumed to be accomplished by the States technique. Thus, for the most part, MFT is currently usable for NH-detected experiments with NH F2F3

2D spectra for 0/90 deg. projection (i.e. F1F3 and F2F3 planes in the 3D case) using the conventional sequence should be acquired before using Explicit Sampling. The data should be transformed, referenced and phased so that the values of rp, lp, rp1, lp1, rp2 and lp2 are correct and present in the current parameter set.

Then, if desired, copy the parameter set to a different "exp" and select "Explicit Sampling" from the menu in the "Sampling/NLS" page in the "Acquire" folder. The

values of rp and lp for directly detected dimension and rp1,lp1,rp2 and lp2 are then present in the parameter set and can be used in MFT processing, if desired. While active, there is no particular benefit to using "half-dwell time" delays in t1 and/or t2, so set f1180='n' and f2180='n' or deactivate the relevant checkboxes.

The experiment is begun with the "Begin Experiment" button. This does an ordinary "go" as well as copying the time domain points file (the full path is visible in the Sample Schedule File entry box) into the current "exp" acqfil. This latter function is not necessary for collection of the data, but it ensures that a copy of the relevant evolution time values is kept unchanged until the next "go" in the current experiment.

If the resulting FID is stored by the BioPack BPsvf macro, the time domain points file from curexp/acqfil is stored together with the fid.

16.2.3 Data Processing

Any non-FFT processing software, in principle, can be used for processing the data. For the case of MFT the following parameters are important.

- Transform sizes: (fn, fn1, fn2) The number of real points is equal to fn*/2 as in standard experiments. There is no relationship between fn1/fn2 and ni. The detected dimension fn should be set as usual fn>=np.
- Weighting parameters: (**sin/cos**: sb, sbs, sb1, sbs1, sb2, sbs2; **exp**: lb, lb1, lb2)
- If weighting is not desired, set the parameters to "not used" by un-checking the checkboxes in the VnmrJ processing pages). At the current time, only sine-bell and exponential weighting are supported. Weighting should not be used when exponential or gaussian distributions are used, but can be used for random sampling.
- Reference frequencies: (reffrq, reffrq1, reffrq2).
- It is especially important is to set a proper reffrq when upper and lower F3 limits are activated (F3start, F3end- in ppm). Reffrq is typically equal to sfrq, but in this case it must be corrected for the actual chemical shift in the center of F3.
- F1F3 diff (sfshift1)

This is the difference between centers of F3 (1H) and F1 (1H) dimensions.
This parameter might be necessary in 3D 13C/15N edited NOESY measurements.

All the parameters entered in the panel will be automatically saved in out.txt file in curexp+'/acqfil' at the start of MFT processing. The output file, with the spectrum in UCSF format (compatibile with the Sparky program), is stored in the same folder as "fid" and "procpar" (curexp/acqfil/).

The button MFT! starts the transformation. The transformation time is proportional to product of fn*fn1*fn2*ni. The MFT Program could also be started directly from a shell. We recommend processing on a data station rather than spectrometer host if speed is an issue.

The file out.txt contains a log of program activity. The file "ucsf" is produced in curexp+'/acqfil' and is the file to use for Sparky.

If you are using the MFT package please cite the following works:

Kazimierczuk K, Kozminski W, Zhukov I, J Magn Reson 179: 323-328 (2006)

Kazimierczuk K, Zawadzka A, Kozminski W, Zhukov I, J Biomol NMR 36: 157-168 (2006)

Kozminski W, Zhukov I, J. Biomol. NMR 26: 157-166, (2003). (2D-quadrature)

16.2.4 File Descriptions

Executable files for Linux

bin/timetab_gen generates table with coordinates of evolution time points

bin/view_tab generates the gnuplot script file for plotting time points

bin/acq_mod creates new panel in "Acquisition" (not needed in BioPack)

bin/seq_mod modifies pulse sequence

bin/ToASTD_3D transforms explicit sampling data using MFT

Interface files

templates/vnmrj/panelitems/TAB_samp.xml

Explicit Sampling panel layout (not implemented in BioPack)

templates/layout/default3dBP/basic.xml (not implemented in BioPack)

templates/layout/default3dBP/Digital_Filter_BioPack.xml

Macros

maclib/BP_ESset creates all required parameters, modifies pulse sequence and produces VnmrJ layout panels

maclib/BP_TABset runs timetab_gen program with appropriate parameters

maclib/BP_PLOTset plots the time points (requires GNUPLOT)

maclib/BP_MFTset executes ToASTD_3D with appropriate parameters

maclib/BPsvf modified BPsvf macro (to store sampling parameters)

Manuals

manual/EXPL_samp MFT experiment manual

manual/EXPL_samp2 MFT processing manual

manual/EXPL_samp_seq How to modify pulse sequences

manual/LICENSE_ESMFT License

manual/README_ESMF This portion of the manual

Contributed by *Wiktor Kozminski, Krzysztof Kazimierczuk, and Anna Zawadzka, Warsaw University.* email: kozmin (at) chem.uw.edu.pl, nmr700.chem.uw.edu.pl

16.2.5 Conversion of a BioPack Sequence to an Explict Sampling Version

The appropriate changes in the pulse sequence code are made by macro BP_ESset which will work for most BioPack sequences. When the macro fails, the following section will assist in doing a proper conversion.

Definitions

The following definition of max_ni should be added to avoid attempting sampling more points than are stored in the timetab file

```
#define MAX_NI 32000
```

Variables declaration

The following variables should be declared

```
FILE *pliktime;
char timetab_name[MAXSTR];
int j,
double t1max = getval("t1max"),
t2max = getval("t2max"),
time_tab[MAX_NI][2],
for 4D experiments :
t3max = getval("t3max"), and
time_tab[MAX_NI][3],
```

Initialize variables

```
getstr("timetab_name",timetab_name);
if( ni>max_ni )
{
printf("ni should be below max_ni ");
psg_abort(1);
 }
pliktime=fopen(timetab_name,"rb");
for (j=0;j<NI;J++)
{
scanf(pliktime,"%lf
%lf\n",&time_tab[j][0],&time_tab[j][1]); //in 3D
}
fclose(pliktime);
if ( ni2 > 1)
{
printf(" ni2 should be set to 1 ");
psg_abort(1);
}
```

For constant time evolution the maximum evolution time should be verified as, for example:

```
if ( 0.5*t2max > timeTN - WFG3_START_DELAY)

{
printf(" t2max is too big. " );
psg_abort(1);

}
```

The variables d2, d3 and d4 should be substituted by:

```
time_tab[d2_index][0],time_tab[d2_index][1] and
time_tab[d2_index][2].
```

In a BioPack sequences it might look like:

```
tau1=t1max*time_tab[d2_index][0];
tau1 = tau1/2.0;
tau2=t2max*time_tab[d2_index][1];
tau2 = tau2/2.0;
```

The States-TPPI calculations could be left as they are.

However, they are not recommend the case of random sampling. (comment them out).

In the case of semi constant time evolution, a few additional modifications should be performed. For example, in the case of BioPack ghn_co.c:

The original code:

```
  ******************************************************

  /* Set up CONSTANT/SEMI-CONSTANT time evolution in N15 */

              halfT2 = 0.0;
              CTdelay = timeTN + pwC8 + WFG_START_DELAY - SAPS_DELAY;


              if(ni>1)
              {
              if(f1180[A] == 'y') /* Set up f1180 */
              tau1 += 0.5*csa/sw1; /* if not PRexp then csa = 1.0 */
              if(PRexp)
              {
                halfT2 = 0.5*(ni-1)/sw1; /* ni2 is not defined */
              if(f1180[A] == 'y')

{
              tau2 += 0.5*sna/sw1; halfT2 += 0.25*sna/sw1;
              }
              t2b = t1_counter*((halfT2 - CTdelay)/(ni-1));
              }
              }
              if (ni2>1)
              {
              halfT2 = 0.5*(ni2-1)/sw2;
              if(f2180[A] == 'y') /* Set up f2180 */
              {
              tau2 += 0.5/sw2; halfT2 += 0.25/sw2;
              }
              t2b = t2_counter*((halfT2 - CTdelay)/(ni2-1));
                 }
                 tau1 = tau1/2.0;
                 tau2 = tau2/2.0;
                 if(tau1 < 0.2e-6) tau1 = 0.0;
                 if(tau2 < 0.2e-6) tau2 = 0.0;
                 if(t2b < 0.0) t2b = 0.0;
                 t2a = CTdelay - tau2 + t2b;
                 if(t2a < 0.2e-6) t2a = 0.0;
******************************************************
```

should be changed to:

```
******************************************************
```

```
/* Set up CONSTANT/SEMI-CONSTANT time evolution in N15 */
                halfT2 = 0.0;
                CTdelay = timeTN + pwC8 + WFG_START_DELAY -
             SAPS_DELAY;

                if(ni>1)
                {
             if(f1180[A] == 'y') /* Set up f1180 */
             tau1 += 0.5*csa/sw1; /* if not PRexp then csa = 1.0 */
                }
                halfT2 = 0.5*t2max;
                if(f2180[A] == 'y') /* Set up f2180 */
                {
             tau2 += 0.5/sw2; halfT2 += 0.25/sw2;
                }
                t2b = tau2*((halfT2 - CTdelay)/t2max);
                tau1 = tau1/2.0;
                tau2 = tau2/2.0;
                if(tau1 < 0.2e-6) tau1 = 0.0;
                if(tau2 < 0.2e-6) tau2 = 0.0;
                if(t2b < 0.0) t2b = 0.0;
                t2a = CTdelay - tau2 + t2b;
                if(t2a < 0.2e-6) t2a = 0.0;
*****************************************************
```

16.2.6 Usage of the MFT Software

MFT and all included libraries (both in source and binary form) and all documentation in this package (the "Software") are copyrighted by Anna Zawadzka, Krzysztof Kazimierczuk and Wiktor Kozminski (the "Authors"). They may be separately used only under the terms of a license agreement from the Authors. If not superceded by another license from the Authors, this license shall govern all use of the Software. The Authors grant to you, an individual or an organization (the "Recipient") free permission to use and display the Software on your computer systems for non-commercial purpose, subject to the conditions below on attribution. The Authors grant to the Recipient permission to modify the Software for internal use only. The Recipient is expressly restricted from redistributing any modified and non-modified versions of the Software without Authors' permission to any other individual or entity.

As a condition of use, the Authors require of the Recipient that any publication or public presentation of scientific results determined using the Software cite the following papers:

Kazimierczuk K, Kozminski W, Zhukov I,Two-dimensional Fourier transform of arbitrarily sampled NMR data sets,J MAGN RESON 179 (2): 323-328 APR 2006

Kazimierczuk K, Zawadzka A, Kozminski W, Zhukov I, Random sampling of evolution time space and Fourier transform processing, J BIOMOL NMR 36 (3): 157-168 NOV 2006

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, DISCLAIMING EXPRESS OR IMPLIED WARRIANTIES, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS

FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Chapter 17 Automatic Spectral Compression

This program uses a peak pick list from a standard 2D experiment to calculate the maximum amount of spectral folding possible without overlap and outputs this value. The input peak list should not contain any aliased peaks. This option is now accessible from the Sampling page in the Acquire folder using the menu "Set Indirect Dimension Sampling" (Figure 13). The 2D spectrum should be phased properly, expanded, and with th/vs2d set such that the peaks displayed are the relevant peaks.

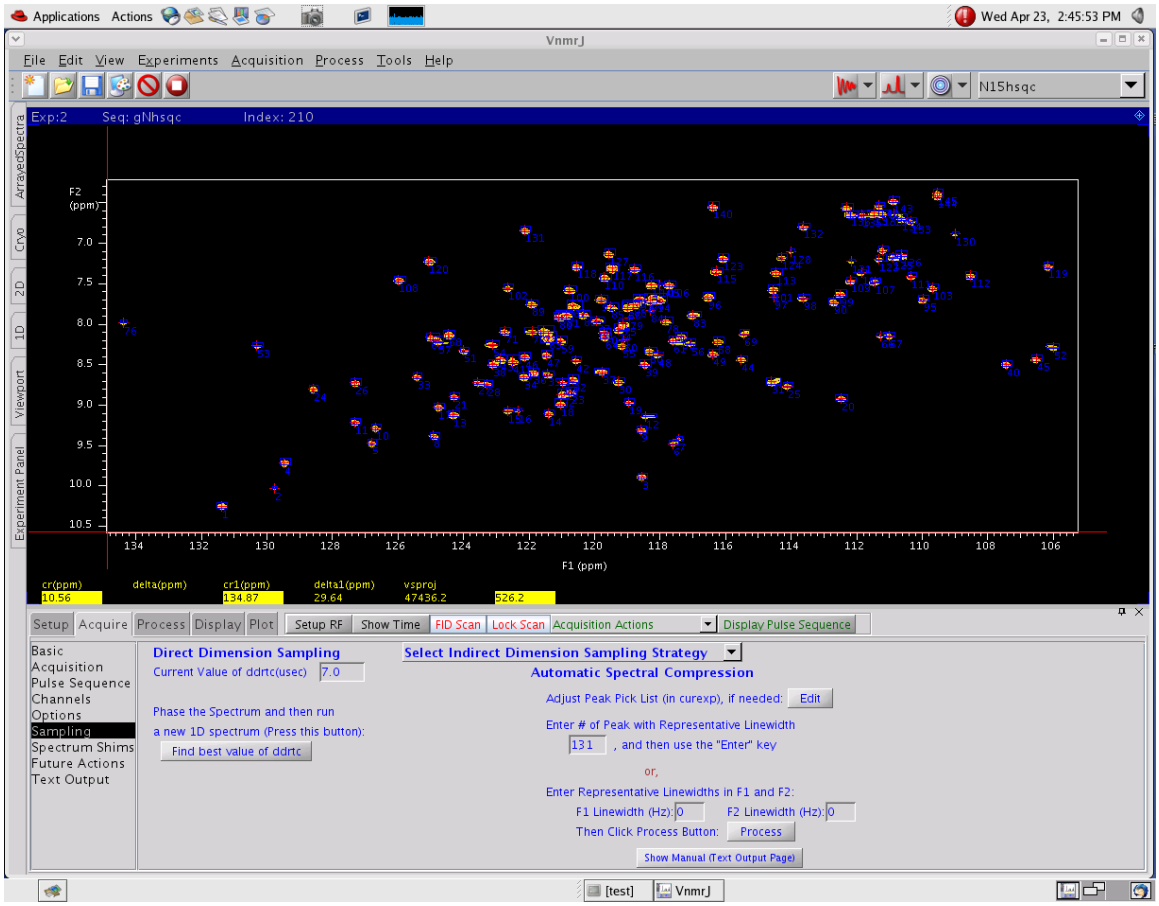


Figure 51 Sampling Page after ASCOM selection

Selecting this option will then perform a peak picking of the displayed 2D spectrum.

The “Edit” button permits editing the peak list to remove any peaks not desired to be considered in the analysis. Enter the peak number for a peak having a representative linewidth followed by the return key, or enter representative linewidths for peaks in F1 and F2 and click the “process” button. In either case, the BestSW4varian.pl program is run which gives output of the new sw1 and ni to give the same indirect dimension resolution (with smaller sw1 and ni).

This permits a much faster acquisition with the same resolution. Separate 2D experiments can be run for F1F3 and F2F3 2D planes in a 3D experiment to calculate new values of sw1 and sw2 for each dimension. This can result in a reduction of time (for the same resolution) of a factor of 4-16.

F1/F2 linewidths

The cross-peaks linewidths in F1 and F2 are directly related to the allowed distance between the centers of two cross-peaks and have a great effect on the degree of folding.

When a clear crosspeak separation is required for precise intensity or peak position extraction (15N relaxation, J-couplings experiments), the F1/F2 linewidths should be set large enough, and should include at least the base of the broader cross-peak.

When 3D experiments are run (e.g. H-N-X), partial overlap in the H-N plane can be allowed as long as the center of the 3D cross-peaks are well separated. Linewidths can be much smaller, for example 15 Hz (@600MHz) in both (1H/15N) dimensions.

Chemical Shifts in F1

The correct frequencies are contained in the "reference" peak list used by ASCOM. The only requirement is to use analysis/visualization software that can automatically fold the "reference" peak list during the mapping on the spectra. NMRView does it very well. For instance, the "strips" command extracts strips from 3D 1H-15N-X experiments at the correct folded 15N position. The algorithm is very simple: subtract or add the spectral width value to the reference 15N frequency until the peak position falls into the spectral window (carrier+-SW/2).

This is also true for ASCOMized 3D HNCO used for 4D experiments. For 4D HNCO-X spectra, a strips4D.tcl script for extracting H-X strips at the aliased N/CO frequencies for NMRView is present in a standalone ASCOM package that will be very soon online on the IBS website. Unfortunately, it seems that sparky does not allow automatic folding.

See *Lescop, Schanda, Rasia and Brutscher, JACS, 129, 2756-2757(2007)*