# TopSpin

## Processing Commands and Parameters

User Manual

Version 006

Innovation with Integrity

NMR

P/N: H9776SA4

# Contents

# Contents

# Contents

# Contents

# 1 Introduction

## 1.1 About this Manual

This manual is a reference to TopSpin processing commands and parameters. Every command is described on a separate page with its syntax and function as well and its main input/output files and input/output parameters. Most of them are processing commands in the sense that they manipulate the data. The manual, however, also includes several commands that analyse data or send information to the screen or printer.

## 1.2 Conventions

### Font and Format Conventions

| Type of Information | Font | Examples |
|---|---|---|
| **Shell Command, Commands,** "All that you can enter" | Arial bold | Type or enter **fromjdx zg** |
| **Button, Tab, Pane and Menu Names**<br><br>"All that you can click" | Arial bold, initial letters capitalized | Use the **Export To File** button.<br>Click **OK**.<br>Click **Process**… |
| **Windows, Dialog Windows, Pop-up Windows Names** | Arial, initial letters capitalized | The Stacked Plot Edit dialog will be displayed. |
| **Path, File, Dataset and Experiment Names**<br>**Data Path Variables**<br>**Table Column Names**<br>**Field Names (within Dialog Windows)** | Arial Italics | *$tshome/exp/stan/nmr/lists*<br>*expno, procno,* |
| **Parameters** | Arial in Capital Letters | VCLIST |
| **Program Code**<br>**Pulse and AU Program Names**<br>**Macros**<br>**Functions**<br>**Arguments**<br>**Variables** | Courier | `go=2`<br>`au_zgte`<br>`edmac`<br>`CalcExpTime()`<br>`XAU(prog, arg)`<br>`disk2, user2` |
| **AU Macro** | Courier in Capital Letters | `REXPNO` |

*Table 1.1: Font and Format Conventions*

### File/Directory Conventions

*<tshome>* - The TopSpin home directory (default *C\:Bruker\Topspin* under Windows (if *C:* is the default drive) or */opt/topspin* under Linux).

*<userhome>* - The user home directory.

**Header Conventions**

SYNTAX - Only included if the command described requires arguments.

USAGE IN AU PROGRAMS - Only included if an AU macro exists for commands described here.

**Commands Conventions**

Please note that after the description of every command the related commands can be found in the paragraph *See Also*. There the mentioned commands are linked and can be clicked. If the mentioned commands are in parenthesis they have no own chapter in this manual, so look for them in the index.

## 1.3 About Directions

TopSpin can process data up to 8-dimension. The directions of a dataset are indicated with the terms F6, F5, F4, F3, F2 and F1 which are used as follows:

**1D data**

F1 - first and only direction

**2D data**

F2 - first direction (acquisition or direct direction)

F1 - second direction (indirect direction)

Commands like **xf2** and **abs2** work in the F2 direction. **xf1**, **abs1** etc. work in F1. **xfb**, **xtrf** etc. work in both F2 and F1.

**3D data**

F3 - first direction (acquisition or direct direction)

F2 - second direction (indirect direction)

F1 - third direction (indirect direction)

**4D data**

F4 - first direction (acquisition or direct direction)

F3 - second direction (indirect direction)

F2 - third direction (indirect direction)

F1 - fourth direction (indirect direction)

Commands like **tf3** and **tabs3** work in F3. **tf2**, **tabs2** etc. work in F2. **tf1**, **tabs1** etc. work in F1.

Data with dimension > 3, can be processed with the command **ftnd**.

## 1.4 About Time and Frequency Domain Data

The result of an acquisition is a representation of intensity values versus acquisition time (seconds); the data are in the time domain. The result of a Fourier transform is a representation of intensity values versus frequency (Hz or ppm); the data are in the frequency domain.

Examples of time domain data are:

- raw data (1D, 2D, and 3D)
- 1D data processed with **bc**, **em** or **gm**

- 2D data processed with **xf2** (time domain in F1)
- 3D data processed with **tf3** (time domain in F2 and F1)

Examples of frequency domain data are:

- 1D data processed with **ft**, **ef**, **gf**, **efp**, **gfp**, **trf***
- 2D data processed with **xfb**, **xf2**, **xf1**, **xtrf***
- 3D data processed **tf3**, **tf2**, **tf1**

Be aware: the commands **trf*** and **xtrf*** only perform a Fourier transform if the processing parameter FT_mod (type **edp**) is set (see **trf**).

Time and frequency domain data can usually be distinguished by the data type (FID versus spectrum) and axis labelling (Hz or ppm versus sec). The only unequivocal way to distinguish them, however, is the processing parameter FT_mod (type **dpp**):

- FT_mod = no : no FT was done and the data are still in the time domain
- FT_mod = f* : FT was done and the data are in the frequency domain
- FT_mod = i* : FT and IFT was done and the data are again in the time domain

## 1.5 About Raw and Processed Data

The result of an acquisition are raw data. Raw data are data which have not been processed in any way. They are stored in:

- *<dir>/data/<user>/nmr/<name>/<expno>/*
  - *fid* - 1D raw data
  - *ser* - 2D or 3D raw data

The result of processing are processed data. They are stored in:

- *<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*
  - *1r*, *1i* - 1D processed data
  - *2rr*, *2ir*, *2ri*, *2ii* - 2D processed data
  - *3rrr*, *3irr*, *3rir*, *3rri* - 3D processed data

Concerning their input data, processing commands can be divided into:

- commands which only work on raw data
- commands which only work on processed data
- commands which work on raw or processed data

### 1.5.1 Commands That Only Work On Raw Data

The following commands only work on raw data. If no raw data exist, they stop with an error message.

- 1D commands **bc**, **trf**, **addfid**, **convdta**
- 2D commands **xtrf**, **xtrf2**, **addser**, **convdta**
- 3D commands **tf3**, **convdta**

### 1.5.2 Commands That Work on Raw Data or Processed Data

The following processing commands work on raw or processed 1D data:

- **em**, **gm**, **sinm**, **qsin**, **sinc**, **qsinc**, **tm**, **traf**, **trafs**,

- **ft**, **ef**, **gf**, **efp**, **gfp**
  - They work on raw data if one of the following is true:
    - no processed data exist (file *1r* and/or *1i* do not exist)
    - processed data exist but they are already Fourier transformed
  - They work on processed data if the following is true:
    - processed data exist but they are not Fourier transformed
- **add**, **addc**, **and**, **div**, **filt**, **ls**, **mul**, **mulc**, **or**, **rs**, **rv**, **xor**, **zf**, **zp**
  - They work on raw data if the parameter DATMOD = raw
  - They work on processed data if the parameter DATMOD = processed

The following processing commands work on raw or processed 2D data:

- **xfb**, **xf2**, **xf1**
  - They work on raw data if one of the following is true:
    - the option *raw* is added, e.g. **xfb raw**
    - no processed data (i.e. the file *2rr*) exist
    - the processing status parameter files *procs* or *proc2s* do not exist or are not readable
    - for **xf2**: data are already Fourier transformed in F2
    - for **xf1**: data are already Fourier transformed in F1
    - for **xfb**: data are already Fourier transformed in both F2 and F1
    - the processing status parameter PH_mod is set to *ps* (power spectrum) or *mc* ( magnitude spectrum) in F2 and/or F1
  - They work on processed data if one of the following is true:
    - the option *proc* is used, e.g. **xfb proc**
    - none of the conditions for using raw data is fulfilled

### 1.5.3 Commands That Always Work on Processed Data

Several processing commands can, by definition, only work on processed data. If no processed data exist, they stop with an error message.

On 1D data:

- **abs**, **absf**, **absd**, **apk**, **apk0**, **apk1**, **apks**, **bcm**, **sab**, **trfp**, **ift**, **ht**, **genfid**, **filt**

On 2D data:

- **abs2**, **abs1**, **abst2**, **abst1**, **sub2**, **sub1**, **sub1d2**, **sub1d1**, **bcm2**, **bcm1**, **xf2p**, **xf1p**, **xfbp**, **xf2m**, **xf1m**, **xfbm**, **xf2ps**, **xf1ps**, **xfbps**, **sym**, **syma**, **symj**, **tilt**, **ptilt**, **ptilt1**, **rev2**, **rev1**, **xif2**, **xif1**, **xht2**, **xht1**, **xtrfp**, **xtrfp2**, **xtrfp1**, **add2d**, **genser**

On 3D data:

- **tf2**, **tf1**, **tht3**, **tht2**, **tht1**,**tf3p**, **tf2p**, **tf1p**,**tabs3**, **tabs2**, **tabs1**

## 1.6 About Digitally Filtered Avance Data

The first points of the raw data measured on an Avance spectrometer are called group delay. These points represent the delay caused by the digital filter and do not contain spectral information. The first points of the group delay are always zero. The group delay only exists if digital filtering is actually used, i.e. if the acquisition parameter DIGMOD is set to digital.

## 1.7 Usage of Processing Commands In Au Programs

Many processing commands described in this manual can also be used in AU programs. The description of these commands contains an entry USAGE IN AU PROGRAMS. This means an AU macro is available which is usually the name of the command in capitalized letters. If the entry USAGE IN AU PROGRAMS is missing, no AU macro is available. Usually, such a command requires user interaction and it would not make sense to put it in an AU program. However, to use such a command in AU, use the XCMD macro which takes a TopSpin command as argument. Examples are:

XCMD("edp")

XCMD("setdef ackn no")

AU programs can be set up with the command **edau**.

Most TopSpin commands can also be used in a TopSpin macro (see **edmac**) or Jython program (see **edpy**).

## 1.8 Clicking Commands from the TopSpin Menu

This manual describes all processing commands as they can be entered on the command line. However, they can also be clicked in the TopSpin menu. Most commands can be found under the *Processing* or *Analysis* menu. The corresponding command line commands are specified in square brackets or appear on right-clicking the menu item.

## 1.9 User Specific Handling of Source Directories

The following paragraph describes the fundamental handling how TopSpin is searching for information like pulse programs, parameter sets, AU programs, lists like VD-list and files like intrng-files (see listing below, section *Source Directories* [▷ 14]). The information where to find these files is stored in the definition of **Source Directories** in TopSpin. There each TopSpin user can add/remove directories and change the order of directories. The order of the directories defines the priority for TopSpin when searching for a file.

This function is complemented now with the function called **Manage Source Directories**. There all user preferences regarding Directory Handling can be defined and are kept.

### 1.9.1 Examples of Use

The following examples describe the new user specific handling of Source Directories in TopSpin in detail:

1. Protection of user defined files.
2. With the new user specific handling of Source Directories all user specific files can be protected. If e.g. all user-files are stored in the own Home-Directory nobody else than the actual user can read or modify any file, because this directory is read- and write protected. This protection for example can be important for pulse program development.
3. Simple and secure working in laboratories with various spectrometers.
4. All TopSpin installations that provide the basis for spectrometer control can use the same directories. **Manage Source Directories** allows to use pulse programs from one common directory so that all modifications and improvements can be used from all spectrometers located in the laboratory. Hence, source directory handling becomes much more comfortable.

### 1.9.2 Source Directories

In TopSpin users can specify individual directories for:

- Pulse Programs
- CPD Programs
- Shape Files
- Gradient Files
- Parameter Sets
- Macros
- Jython Programs
- Python 3 Programs
- AU Programs
- VD Delay lists
- VP Loup Cont lists
- VC lists
- VA Amplitude lists
- VT Temperature lists
- F1 Frequency lists
- SP Shape lists
- DS Data Set lists
- Solvent Region Files
- Phase Program lists
- *intrng* files
- *peakrng* files
- *baslpnts* files
- *base_info* files
- *peaklist* files
- *clevels* files
- *reg* files
- *int2drng* files
- Structure files

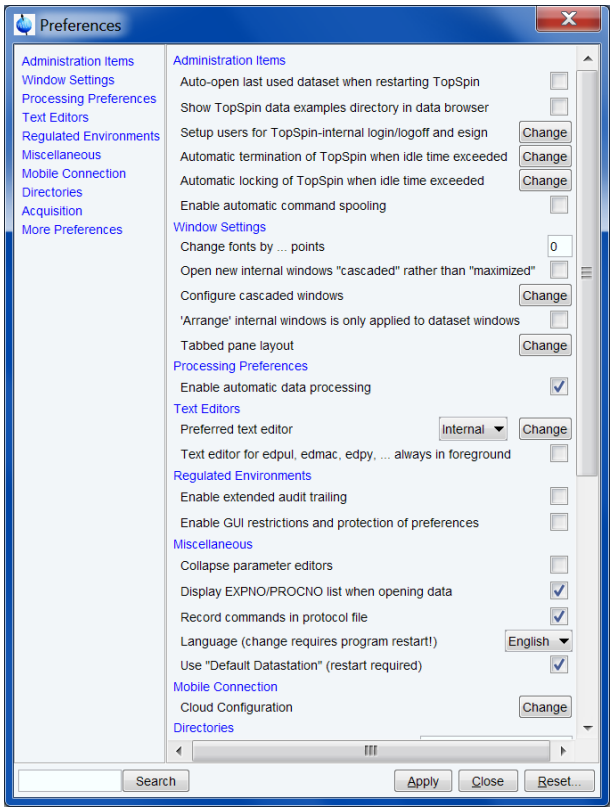
### 1.9.3 Default directories

The default paths for directories, e.g. Pulse Programs, are:
Bruker files in: *.../exp/stan/nmr/lists/*pp
User files in: *.../exp/stan/nmr/lists/pp/user*
The default path for lists, e.g. VD lists, is
Bruker/User files in: *.../exp/stan/nmr/lists/vd*

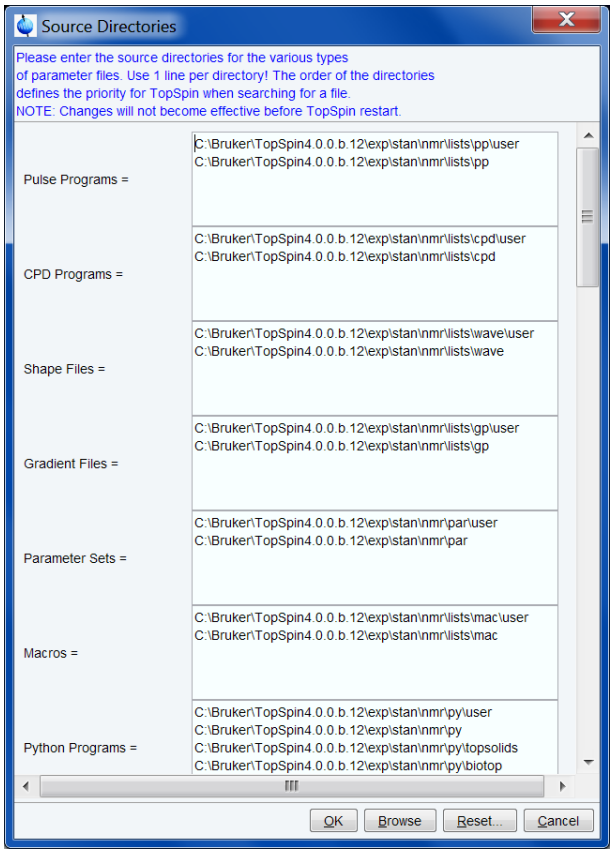
### 1.9.4 How to Define User Specific Directories

- In the menu bar, click **Setup Preferences**  .

- In the Preferences window, in the group *Directories*, in the line *Manage source directories for edpul, edau, etc.* click **Change**.




- In the Source Directories window, click **Browse** to select a user defined directory and click **OK**.




With this structure each user can define his own directories in an unlimited number.

This window enables the user to define the individual directories for all files as Pulse Programs, AU Programs etc. For the complete list of Source Directories see paragraph *Source Directories [▷ 14]*.

The order of the directories defines the priority for TopSpin when searching for a file.

Note that changes will not become effective before TopSpin restarts.

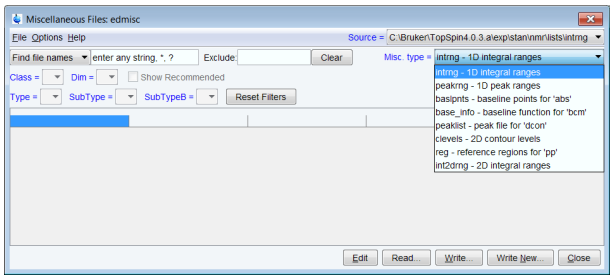
### 1.9.5 How to Define User Specific Directories with Commands

User specific directories can also be configured from the corresponding reading/writing and editing commands for the respective information like pulse programs, parameter sets, AU programs, lists and files.

For defining special lists please enter the corresponding command in the command line:

- Pulse Programs (**edpul**)
- CPD Programs (**edcpd**)
- Shape Files (**edshape**)
- Parameter Sets (**edpar**)
- Macros (**edmac**)
- Jython programs (**edpy**)
- Python 3 programs (**edpy3**)
- AU Programs (**edau**)
- VD, VP, VC, VA, VT, F1, DS, Solvent Region Files, Phases (**edlist**)
- *intrng* Files, *peakrng* Files etc. (**edmisc**)

After entering the respective command in the command line, TopSpin will open the corresponding window in appearance like the following window. Here the example for the command **edmisc**:




On the top right of this window the sources are listed in the pull-down menu and below the file types are shown also in a pull-down menu.

All items can be edited, read, written or written new depending on user wishes.

- Click **Options** | **Manage Source Directories** to define user-specific directories for Source Directories as described above.

Please note that in the following chapters where the respective commands for pulse programs, parameter sets, AU programs, lists and files are described, we will always refer to this chapter and the function **Options** | **Manage Source Directories**.

# 2 TopSpin Parameters

## 2.1 About TopSpin Parameters

TopSpin parameters are divided in acquisition and processing parameters. In this manual, we will mainly concern ourselves with processing parameters.

The following terms are used:

**Processing Parameters**

Parameters which must be set, for example by entering **edp** or clicking the Procpars tab, and are interpreted by processing commands.

**Acquisition Status Parameters**

Parameters which are set by acquisition commands like **zg**. They represent the acquisition status of a dataset and can be viewed, for example, by entering **dpa** or clicking the Acqupars tab. Some acquisition status parameters are used as input by processing commands.

**Processing Status Parameters**

Parameters which are set by processing commands. They represent the processing status of a dataset and can be viewed, for example, by **dpp** or by clicking the Procpars tab. Most processing status parameters get the value of the corresponding processing parameter as it was set by the user (**edp**). Some parameters, however, are explicitly set or modified by the processing command.

**Input Parameters**

Parameters which are interpreted by processing commands. These can be:

- Processing parameters (set by the user). Most input parameters are processing parameters.
- Acquisition status parameters (set by an acquisition command). An example is parameter AQ_mod.
- Processing status parameters (set by the previous processing command). An example is the parameter SI set by **ft** and then interpreted by **abs**. This means you cannot change the size between **ft** and **abs**.

**Output Parameters**

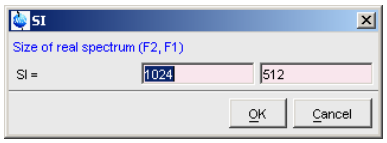
Parameters which are set or modified by processing commands. These can be:

- Processing status parameters. Examples are FT_mod and YMAX_p, set by **ft**. Most output parameters are processing status parameters.
- Processing parameters. Examples are PHC0 and PHC1, set by **apk** and SR and OFFSET, set by **sref**.

Processing parameters can be set with the parameter editor **edp** and processing status parameters can be viewed with **dpp**. Alternatively, each parameter can be set or viewed by entering its name in lowercase letters on the command line. For example, the parameter SI:

- **si** - set the parameter SI
- **s si** - view the status parameter SI

The dimensionality of the dataset is automatically recognized. For example, for a 2D dataset the following dialog box is offered:




Since F1 is the acquisition direction and F2 the indirect direction, the 2D spectrum data will acquire FID's with 1024 points using 512 experiments. Although status parameters are normally not changed by the user, a command like **s si** allows to do that. This, however, could make the dataset inconsistent which can be checked with the command **auditcheck**.

Before any processing has been done, the processing status parameters of a dataset do not contain significant values. After the first processing command, they represent the current processing status of the data. Any further processing command will update the processing status parameters.

After processing, the relevant processing status parameters are usually set to the same values as the corresponding processing parameters. In other words, the command has done what you told it to do. There are, however, some exceptions:

- When a processing command was interrupted, the processing status parameters might not have been updated yet.
- Some processing parameters are modified by the processing command, e.g. STSI is rounded to the next higher multiple of 16 by **xfb**. The rounded value is stored as the processing status parameter.
- The values of some parameters are a result of processing. They cannot be set by the user (they do not appear as processing parameters) but they are stored as processing status parameters. Examples are NC_proc, S_DEV and TILT.

## 2.2 Parameter Values

With respect to the type of values they take, parameters can be divided into three groups:

- Parameters taking integer values, e.g. SI, TDeff, ABSG, NSP.
- Parameters taking float or double values, e.g. LB, PHC0, ABSF1.
- Parameters using a predefined list of values, e.g. BC_mod, WDW, PSCAL.

You can easily see to which group a parameter belongs from the parameter editor opened by entering **edp** or clicking the Procpars tab.

Note that the values of parameters which use a predefined list are actually stored as integers.

The first value of the list is always stored as 0, the second value as 1 etc. The following table shows the values of the parameter PH_mod as an example:

| Parameter Value | Integer Stored in the Proc(s) File |
|---|---|
| no | 0 |
| pk | 1 |
| mc | 2 |
| ps | 3 |

## 2.3 Parameter Files

TopSpin parameters are stored in various files in the data set directory tree.

**In a 1D Dataset:**

- *<dir>/data/<user>/nmr/<name>/<expno>/*
  - *acqu* - acquisition parameters
  - *acqus* - acquisition status parameters
- *<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*
  - *proc* - processing parameters
  - *procs* - processing status parameters

**In a 2D Dataset:**

- *<dir>/data/<user>/nmr/<name>/<expno>/*
  - *acqu* - F2 acquisition parameters
  - *acqu2* - F1 acquisition parameters
  - *acqus* - F2 acquisition status parameters
  - *acqu2s* - F1 acquisition status parameters
- *<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*
  - *proc* - F2 processing parameters
  - *proc2* - F1 processing parameters
  - *procs* - F2 processing status parameters
  - *proc2s* - F1 processing status parameters

**In a 3D Dataset:**

- *<dir>/data/<user>/nmr/<name>/<expno>/*
  - *acqu* - F3 acquisition parameters
  - *acqu2* - F2 acquisition parameters
  - *acqu3* - F1 acquisition parameters
  - *acqus* - F3 acquisition status parameters
  - *acqu2s* - F2 acquisition status parameters
  - *acqu3s* - F1 acquisition status parameters
- *<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*
  - *proc* - F3 processing parameters
  - *proc2* - F2 processing parameters
  - *proc3* - F1 processing parameters
  - *procs* - F3 processing status parameters
  - *proc2s* - F2 processing status parameters
  - *proc3s* - F1 processing status parameters

## 2.4 List of Processing Parameters

This paragraph contains a list of all processing parameters with a description of their function and the commands they are interpreted by. Please note that composite processing commands like **efp** (which combines **em**, **ft** and **pk**) are not mentioned here. Nevertheless,

they interpret all parameters which are interpreted by the single commands they combine. Processing parameters can be set from the parameter editor, which can be opened by entering **edp** or clicking the Procpars tab. Alternatively, set the parameters by entering their names in lowercase letters on the command line.

**ABSF1 - low field limit of the region which is baseline corrected**

- used in 1D, 2D and 3D data sets in all directions
- takes a float value (ppm) and must be greater than ABSF2
- interpreted by **absf**, **apkf**, **abs1**, **abs2**, **abst***, **absot***, **zert***, **tabs***
- The 1D commands **abs** and **absd** do not interpret ABSF1 because they work on the entire spectrum. The command **apkf**, for automatic phase correction, uses ABSF1 as the left limit of the region on which it calculates the phase values.

**ABSF2 - high field limit of the region which is baseline corrected**

- used in 1D, 2D and 3D data sets in all directions
- takes a float value (ppm), must be smaller than ABSF1
- interpreted by **absf**, **apkf**, **abs2**, **abs1**, **abst***, **absot***, **zert***, **tabs***
- The 1D commands **abs** and **absd** do not interpret ABSF2 because they work on the entire spectrum. The command **apkf**, for automatic phase correction, uses ABSF2 as the right limit of the region on which it calculates the phase values.

**ABSG - degree of the polynomial which is subtracted in baseline correction**

- used in 1D, 2D and 3D data sets in all directions
- takes an integer value between 0 and 5 (default is 5)
- interpreted by **abs**, **absd**, **absf**, **abs2**, **abs1**, **abst***, **absot***, **tabs***
- A polynomial of degree ABSG is calculated by the baseline correction commands and then subtracted from the spectrum.

**ABSL - integral sensitivity factor with reference to the noise**

- used in 1D data sets
- takes a float value between 0 and 100 (default is 3)
- interpreted by **abs**, **absd**, **absf**
- Data points greater than ABSL*(standard deviation) are considered spectral information, all other points are considered noise.

**ALPHA - correction factor**

- used in 2D data sets in F2 and F1
- takes a float value
- interpreted by **ptilt**, **ptilt1** and **add2d**
- For **ptilt**, F2 ALPHA is the tilt factor. For **ptilt1**, F1 ALPHA is the tilt factor. They must have a value between -2.0 and 2.0. For **add2d**, F2 ALPHA is the multiplication factor for the current dataset (see also parameter GAMMA).

**AQORDER - Acquisition order**

- used in data sets with dimensionality ≥ 3
- takes one of the values *321*, *312* for 3D data
- takes one of the values *4321*, *4312, 4231*, etc. for 4D data
- takes ..... etc.

- only interpreted if AQSEQ is not set, by the processing commands **ftnd** and **tf3**
- AQORDER describes the order in which the indirect directions have been acquired. For example, a 3D pulse program usually contains a double nested loop with loop counters `td1` and `td2`. If `td1` is used in the inner loop and `td2` in the outer loop, the acquisition order is 312. Otherwise it is 321.

The acquisition order is normally evaluated from the acquisition status parameter AQSEQ. Only if this parameter is not set, AQORDER is used.

**ASSFAC - assign the highest or second highest peak as** reference for scaling

- used in 1D data sets
- takes a float value (default is 0.0)
- interpreted by **pp***, **lipp***
- This parameter is interpreted as follows:
- If ASSFAC > 1, the second highest peak is used as reference for scaling, if the following is true: *h2* < *hmax/ASSFAC*, where *h2* is the intensity of the second highest peak and *hmax* the intensity of the highest peak. If this condition is false, the highest peak is used as reference.
- Other values of ASSFAC have no effect on the plot scaling.

**ASSWID - region excluded from second highest peak search**

- used in 1D data sets
- takes a float value (Hz, default is 0)
- interpreted by **pp***, **lipp***
- ASSWID is interpreted as follows:
- If abs(ASSFAC) > 1, a region of width ASSWID around the highest peak is excluded from the search for the second highest peak

**AUNMP - processing AU program name**

- used in 1D, 2D and 3D data sets in the first direction
- takes a character string value
- interpreted by **xaup**
- In all Bruker standard parameter sets, the parameter AUNMP is set to a suitable processing AU program.

**AZFE - integral extension factor**

- used in 1D data sets
- takes a float value (ppm, default 0.1)
- interpreted by **abs**
- Integral regions are extended at both sides by AZFE ppm. If this extension causes adjacent regions to overlap, the centre of the overlap is used as the limit of the two regions.

**AZFW - minimum distance between peaks for independent integration**

- used in 1D data sets
- takes a float value (ppm)

- interpreted by **abs**, **ldcon**, **gdcon**, **mdcon**
- If peaks are more than AZFW apart, they are treated independently. If peaks are less than AZFW ppm apart, they are considered to be overlapping.

**BCFW - filter width for FID baseline correction.**

- used in 1D data sets
- takes a float value (ppm)
- interpreted by **bc** when BC_mod = sfil or qfil
- sfil/qfil is used to suppress signals in the center of the spectrum. BCFW determines the width of the region, around the center of the spectrum, which is affected by **bc**.

**BC_mod - FID baseline correction mode**

- used for 1D, 2D, and 3D dataset in all directions
- (only useful in the acquisition direction)
- takes one of the values *no*, *single*, *quad*, *spol*, *qpol*, *sfil*, *qfil*
- interpreted by **bc**, **em**, **gm**, **ft**, **trf**, **xfb**, **xf2**, **xf1**, **xtrf***, **tf***
- The values of BC_mod and the corresponding functions are shown in the next table. Most commands evaluate BC_mod for the function to be subtracted but not for the detection mode. The latter is then evaluated from the acquisition status parameter AQ_mod. This means, for example, it does not matter if you set BC_mod to *single* or *quad*. Only **trf** and **xtrf*** evaluate the detection mode from BC_mod and distinguish between BC_mod = single and BC_mod = quad. The same counts for the values *spol/qpol* and *sfil/qfil*.

| BC_mod | Function Subtracted from the FID | Detection Mode |
|---|---|---|
| no | no function | |
| single | average intensity of the last quarter of the FID | single channel |
| quad | average intensity of the last quarter of the FID | quadrature |
| spol | polynomial of degree 5 ( least square fit) | single channel |
| qpol | polynomial of degree 5 (least square fit) | quadrature |
| sfil | Gaussian function of width BCFW **a** | single channel |
| qfil | Gaussian function of width BCFW **a** | quadrature |

Marion, Ikura, Bax, J. Magn. Res. 84, 425-420 (1989)

**COROFFS - correction offset for FID baseline correction**

- used in 1D, 2D and 3D data sets in all directions
- takes a double value (Hz, default is 0.0)
- interpreted by **bc**, **em**, **gm**, **trf**, **xfb**, **xf2**, **xf1**, **xtrf***, **tf3**, **tf2**, **tf1**
- COROFFS is only interpreted for BC_mod = qpol or qfil. The center of the baseline correction is shifted by COROFFS Hz.

**CURPLOT - Default plotter for Plot Editor**

- used in 1D and 2D data sets
- interpreted by **plot** and **autoplot**
- The plotter set by CURPLOT overrides the plotter specified in the Plot Editor Layout. It allows to use the same plotter for all layouts.

**DATMOD - data mode: work on 'raw' or 'processed data**

- used in 1D data sets
- takes the value *raw* or *proc*
- interpreted by **add**, **addc**, **and**, **div**, **filt**, **mul**, **mulc**, **ls**, **or**, **rs**, **rv**, **xor**, **zf**, **zp**

**DC - multiplication factor or addition constant**

- used in 1D data sets
- takes a float value
- interpreted by **add**, **addc**, **addfid** and **mulc**
- For **addc**, DC is an addition constant. For **add**, **addfid** and **mulc**, DC is a multiplication factor.

**DFILT - Digital filter filename**

- used in 1D data sets
- takes a character string value
- interpreted by **filt**
- The file specified by DFILT must reside in the directory: *<tshome>/exp/stan/nmr/filt/1d*
- and must be set up from a command shell. One standard file called threepoint is delivered with TopSpin.

**FCOR - first (FID) data point multiplication factor**

- used in 1D, 2D and 3D data sets in all directions
- takes a float value between 0.0 and 2.0
- interpreted by **ft**, **trf**, **xfb**, **xf2**, **xf1**, **xtrf**, **xtrfp**, **tf3**, **tf2**, **tf1**
- For 1D digitally filtered Avance data (DIGMOD = digital), FCOR does not play a role because the first raw data point is always zero. FCOR, however, allows to control the DC offset of the spectrum in the following cases:
  - on A*X data
  - on Avance data measured in analog mode (DIGMOD = analog)
  - on 2D/3D Avance data in the second/second+third direction

**FT_mod - Fourier transform mode**

- used in 1D, 2D and 3D in all directions
- takes one of the values *no*, *fsr*, *fqr*, *fsc*, *fqc*, *isr*, *iqr*, iqc, *isc*
- interpreted by **trf**, **xtrf***, **xtrfp***
- the Fourier transform commands **ft** (1D), **xfb**, **xf2**, **xf1** (2D) and **tf*** (3D) do not interpret FT_mod because they evaluate the Fourier transform mode from the acquisition status parameter AQ_mod. They do, however, set the processing status parameter FT_mod.
- The values of FT_mod have the following meaning:

| FT_mod | Fourier Transform Mode |
|---|---|
| no | no Fourier transform |
| fsr | forward, single channel, real |
| fqr | forward, quadrature, real |
| fsc | forward, single channel, complex |
| fqc | forward, quadrature, complex |
| isr | inverse, single channel, real |
| iqr | inverse, quadrature, real |
| isc | inverse, single channel, complex |
| iqc | inverse, quadrature, complex |

**GAMMA - multiplication factor**

- used in 2D data sets in F2
- takes a float value
- interpreted by **add2d**
- GAMMA is the multiplication factor for the second dataset (see also parameter ALPHA).

**GB - Gaussian broadening factor for Gaussian window multiplication**

- used in 1D, 2D and 3D data sets in all directions
- takes a float value between 0.0 and 1.0
- interpreted by **gm**
- interpreted by **trf**, **xfb**, **xf2**, **xf1**, **xtrf***, **tf*** if WDW = EM or GM

**INTBC - automatic baseline correction of integrals created by abs**

- used in 1D data sets
- takes the value *yes* or *no*
- interpreted by **li**, **lipp**, **lippf**
- INTBC has no effect on integrals which were created interactively in the *Integration* mode.

**INTSCL - scale 1D integrals relative to a reference dataset**

- used in 1D data sets
- takes an integer value
- interpreted by **li**, **lipp**, **lippf**
- INTSCL is used as follows:
  - For INTSCL > 0, the integral values are scaled individually for each spectrum.
  - For INTSCL = 0, the integrals on the plot will obtain the same numeric values as defined interactively in the integration mode.
  - For INTSCL = -1, scaling is performed relatively to the last spectrum plotted.

**ISEN - integral sensitivity factor with reference to the largest integral**

- used in 1D data sets
- takes a positive float value (default 128)
- interpreted by **abs**, **absd**, **absf**

- Only the regions of integrals which are larger (area) than the largest integral divided by ISEN are stored.

**LB - Lorentzian broadening factor for exponential window multiplication**

- used in 1D, 2D and 3D data sets in all directions
- takes a float value
- interpreted by **em**, **gm**
- interpreted by **trf**, **xfb**, **xf2**, **xf1**, **xtrf***, **tf*** if WDW = EM or GM
- LB must be positive for an exponential and negative for Gaussian window multiplication.

**LEV0 - lowest 2D contour level multiplication factor**

- used in 2D data sets in F2
- takes a positive float value (default is 35)
- interpreted by **levcalc**
- **levcalc** sets the lowest contour level to LEV0*S_DEV, where S_DEV (standard deviation) is a processing status parameter.

**LPBIN - number of points for linear prediction**

- used in 1D, 2D and 3D data sets in all directions
- takes a positive integer value
- interpreted by **ft**, **trf**, **xfb**, **xf2**, **xf1**, **xtrf***, **tf***
- also interpreted by **em**, **gm**, ***sin***, **tm**, **traf***

For backward prediction, LPBIN represents the number of input points with a maximum of TD - abs(TDoff). The default value of LPBIN is zero, which means all data points are used as input. The status parameter LPBIN (**dpp**) shows how many input points were actually used. For forward prediction, LPBIN can be used to reduce the number of prediction output points as specified in the next table . **Note**: LPBIN only has an effect in the last two cases. If LPBIN is smaller than TD or greater than 2*SI this has the same effect as LPBIN = 0.

| Parameter Values | Normal Points | Predicted Points | Zeroes |
|---|---|---|---|
| LPBIN = 0, 2*SI < TD | 2*SI | - | - |
| LPBIN = 0, TD < 2*SI < 2*TD | TD | 2*SI - TD | - |
| LPBIN = 0, 2*TD < 2*SI | TD | TD | 2*SI - 2*TD |
| TD < LPBIN < 2*SI< 2*TD | TD | LPBIN - TD | 2*SI - LPBIN |
| TD < LPBIN < 2*TD < 2*SI | TD | LPBIN - TD | 2*SI - LPBIN |

**MAXI - maximum relative intensity for peak picking**

- used in 1D data sets
- takes a float value (cm)
- interpreted by **pp***, **li**, **lipp***
- only peaks with an intensity smaller than MAXI will appear in the peak list. MAXI can also be set from the **pp** dialog box and, interactively, in peak picking mode.

**MC2 - Fourier transform mode of the second (and third) direction**

The processing parameter MC2 is only interpreted if the acquisition status parameter FnMODE (**dpa**) does not exist or has the value *undefined*. FnMODE must be set (with **eda**) according to the experiment type before the acquisition is started. As MC2, FnMODE only exists in the second (and third) direction. On data sets acquired with XWIN-NMR 2.6 or earlier, MC2 is interpreted and must be set before the data are processed. The parameter MC2:

- is used in 2D data sets in the second direction (F1)
- is used in 3D data sets in the second and third direction (F2 and F1)
- takes one of the values *QF*, *QSEQ*, *TPPI*, *States*, *States-TPPI*, *echo-antiecho*
- is interpreted by **xfb**, **xf2**, **xf1**, **xtrf***, **tf***

**ME_mod - FID linear prediction mode**

- used in 1D, 2D and 3D data sets in all directions
- takes one of the values *no*, *LPfr*, *LPfc*, *LPbr*, *LPbc, LPmifr*, *LPmifc*
- interpreted by **ft**, **trf**, **xfb**, **xf2**, **xf1**, **xtrf***, **tf***
- also interpreted by **em**, **gm**, ***sin***, **tm**, **traf***
- The values of ME_mod have the following meaning:

| | |
|---|---|
| LPfr | forward LP on real data |
| LPfc | forward LP on complex data |
| LPbr | backward LP on real data |
| LPbc | backward LP on complex data |
| LPmifr | mirror image forward LP on real data |
| LPmifc | mirror image forward LP on complex data |

Linear prediction is only performed for NCOEF > 0. Furthermore, LPBIN and, for backward prediction, TDoff play a role. The commands **ft**, **xfb**, **xf2** and **xf1** evaluate ME_mod but do not distinguish between LPfr and LPfc nor do they distinguish between LPbr and LPbc. The reason is that the detection mode (real or complex) is evaluated from the acquisition status parameter AQ_mod. However, **trf**, **xtrf** and **xtrf2** evaluate the detection mode from ME_mod. In 1D, a combination of forward and backward prediction can be done by running **trf** with ME_mod = LPfc and **trfp** (or **ft**) with ME_mod = LPbc. In 2D, this would be the sequence **xtrf** - **xtrfp** (or **xfb**). Note that not only Fourier transform but also window multiplication commands perform linear prediction when ME_mod is set. This allows to easily see the effect of linear prediction on the FID, for example by executing **em** with LB = 0.

**MI - minimum relative intensity for peak picking**

- used in 1D data sets
- takes a float value (cm)
- interpreted by **pp***, **li**, **lipp***
- only peaks with an intensity greater than MI will appear in the peak list. MI can also be set from the **pp** dialog box and, interactively, in peak picking mode.

**NCOEF - number of linear prediction coefficients**

- used in on 1D, 2D and 3D data sets in all directions
- takes a positive integer value (default is 0)

- interpreted by **ft**, **trf**, **xfb**, **xf2**, **xf1**, **xtrf***, **tf***
- also interpreted by **em**, **gm**, ***sin***, **tm**, **traf***
- NCOEF is typically set to 2-3 times the number of expected peaks. For NCOEF = 0, no prediction is done. Linear prediction also depends on the parameters ME_mod, LPBIN and TDoff.

**NLEV - number of positive contour levels in a 2D spectrum**

- used in 2D data sets in the F2 direction
- takes positive integer value (default 6)
- interpreted by **levcalc**
- The total number of levels (positive and negative) calculated by **levcalc** is 2*NLEV

**NOISF1 - low field (left) limit of the noise region**

- used in 1D data sets
- takes a float value (ppm)
- interpreted by **sino**
- The noise in the region between NOISF1 and NOISF2 is calculated according to the algorithm described for the command **sino**.

**NOISF2 - high field (right) limit of the noise region**

- used in 1D data sets
- takes a float value (ppm)
- interpreted by **sino**
- The noise in the region between NOISF1 and NOISF2 is calculated according to the algorithm described for the command **sino**.

**NSP - number of data points shifted during right shift or left shift**

- used in 1D data sets
- takes a positive integer value (default is 1)
- interpreted by **ls** and **rs**
- NSP points are discarded from one end and NSP zeroes are added to the other end of the spectrum.

**NZP - number of data points set to zero intensity**

- used in 1D data sets
- takes a positive integer value (default is 0)
- interpreted by **zp**
- **zp** sets the intensity of the first NZP points of the dataset to zero.

**OFFSET - the ppm value of the first data point of the spectrum**

- used in 1D, 2D and 3D data sets in all directions
- takes a float value (ppm)
- set by **sref** or interactive calibration
- also set by **accumulate**
- The value is calculated according to the relation:

  OFFSET = (SFO1/SF-1) * 1.0e6 + 0.5 * SW * SFO1/SF

Where SW and SFO1 are acquisition status parameters. In fact, the relation for OFFSET depends on the acquisition mode. When the acquisition status parameter AQ_mod is *qsim*, *qseq* or *DQD*, which is usually the case, the above relation counts. When AQ_mod is *qf*, the following equation is used:

OFFSET = (SFO1/SF-1) * 1.0e6

**PC - peak picking sensitivity**

- used in 1D data sets
- takes a float value
- interpreted by **pp***, **li**, **lipp***
- a spectral point is only a considered peak if it is a maximum which is greater than the previous minimum plus 4*PC*noise. In addition to MI, PC provides an extra way of controlling the peak picking sensitivity. It allows, for instance, to detect a shoulder on a large peak.

**PHC0 - zero order phase correction value (frequency independent)**

- used in 1D, 2D and 3D data sets in all directions
- takes a float value (degrees)
- set by **apk**, **apks**, **apkf**, **apk0** on 1D data sets
- set interactively in Phase correction mode on 1D and 2D data sets
- interpreted by **pk**, **xfbp**, **xf2p**, **xf1p**, **tf*p**
- interpreted by **trf**, **xfb**, **xf2**, **xf1**, **xtrf***, **tf3**, **tf2**, **tf1** when PH_mod = pk
- PHC0 is one of the few examples where a processing parameter is set by a processing command. For example, **apk** sets both the processing and processing status parameter PHC0. **pk** reads the processing parameter and updates the processing status parameter. For multiple phase corrections, the total zero order phase value is stored as the processing status parameter PHC0.

**PHC1 - first order phase correction value (frequency dependent)**

- used in 1D, 2D and 3D data sets in all directions
- takes a float value (degrees)
- set by **apk**, **apks**, **apkf**, **apk1** on 1D data sets
- set interactively in Phase correction mode on 1D and 2D data sets
- interpreted by **pk**, **xfbp**, **xf2p**, **xf1p**, **tf*p**
- interpreted by **trf**, **xfb**, **xf2**, **xf1**, **xtrf***, **tf3**, **tf2**, **tf1** when PH_mod = pk
- PHC1 is one of the few examples where a processing parameter is set by a processing command. For example, **apk** sets both the processing and processing status parameter PHC1. **pk** reads the processing parameter and updates the processing status parameter. For multiple phase corrections the total first order phase value is stored as the processing status parameter PHC1.

**PH_mod - phase correction mode**

- used in 1D, 2D and 3D data sets in all directions
- takes one of the value *no*, *pk*, *mc*, *ps*
- interpreted by **trf**, **xfb**, **xf2**, **xf1**, **xtrf***, **tf***
- The values of PH_mod are described in following table:

| PH_mod | Mode |
|---|---|

| | |
|---|---|
| no | No phase correction |
| pk | Phase correction according to PHC0 and PHC1 |
| mc | Magnitude calculation |
| ps | Power spectrum |

- The value PH_mod = pk is only useful if the phase values are known and the parameters PHC0 and PHC1 have been set accordingly. In 1D, they can be determined with **apk** or **apks**, or, interactively, from the Phase correction mode. In 2D and 3D, they can only be determined interactively.

**PKNL - group delay compensation (Avance) or filter correction (A*X)**

- used in 1D, 2D and 3D data sets in the first direction
- takes the value *true* or *false*
- interpreted by **ft**, **trf**, **xfb**, **xf2**, **xf1**, **xtrf***, **tf***
- On A*X spectrometers, PKNL = true causes a non linear 5th order phase correction of the raw data. This corrects possible errors caused by non linear behaviour of the analog filters. On Avance spectrometers, PKNL must always be set to TRUE. For digitally filtered data, it causes **ft** to handle the group delay of the FID. For analog data it has no effect.

**PSCAL - determines the region with the reference peak for vertical scaling**

- used in 1D data sets
- takes one of the values *global*, *preg*, *ireg*, *pireg*, *sreg*, *psreg*, *noise*
- interpreted by **pp***, **li**, **lipp***
- the values of PSCAL have the following meaning:

| PSCAL | Peak used as reference for vertical scaling |
|---|---|
| global | The highest peak of the entire spectrum. |
| preg | The highest peak within the plot region. |
| ireg | The highest peak within the regions specified in the reg file. If the reg file does not exist, global is used. |
| pireg | as ireg, but the peak must also lie within the plot region. |
| sreg | The highest peak in the regions specified in scaling region file. This file is specified by the parameter SREGLST. If SREGLST is not set or specifies a file which does not exist, global is used. |
| psreg | as sreg but the peak must also lie within the plot region. |
| noise | The intensity of the noise. |

- For PSCAL = ireg or pireg, the reg file is interpreted. The reg file can be created in interactive *integration* mode and can be viewed or edited with the command **edmisc reg**.
- For PSCAL = sreg or psreg, the scaling region file is interpreted. This feature is used to exclude the region in which the solvent peak is expected. The name of a signal region file is typically of the form NUCLEUS.SOLVENT, e.g. 1H.CDCl3. For all common nucleus/solvent combinations, a scaling region file is delivered with TopSpin. In several 1D standard parameter sets which are used during automation, PSCAL is set to *sreg* and SREGLST to NUCLEUS.SOLVENT as defined by the parameters NUCLEUS and SOLVENT.

**PSIGN - peak sign for peak picking**

- used in 1D data sets
- takes the value *pos*, *neg* or *both* (default is *pos*)
- interpreted by **pp***, **lipp***
- in most 1D standard parameter sets PSIGN is set to *pos* which means only positive peaks are picked

**REVERSE - flag indicating to reverse the spectrum during Fourier transform**

- used in 1D, 2D and 3D data sets in all directions
- takes the value *true* or *false* (default is *false*)
- interpreted by **ft**, **trf**, **xfb**, **xf2**, **xf1**, **xtrf***, **tf***
- Reversing the spectrum can also be done after Fourier transform with the commands **rv** (1D) or **rev2**, **rev1** (2D).

**SF - spectral reference frequency**

- used in 1D, 2D and 3D data sets in the first direction
- takes a positive float value
- set by **sref** or interactive calibration
- **sref** calculates SF according to the relation:
- SF=BF1/(1.0+RShift * 1e-6)
- 
- Where *RShift* is taken from the **edlock** table and BF1 is an acquisition status parameter. SF is interpreted by display and plot routines for generating the axis (scale) calibration.

**SI - size of the processed data**

- used in 1D, 2D and 3D data sets in all directions
- takes an integer value
- interpreted by processing commands which work on the raw data (commands working on processed interpret the processing status parameter SI)
- The total size of the processed data (real+imaginary) is 2*SI. In Bruker standard parameter sets (see **rpar**), SI is set to TD/2, where TD is an acquisition status parameter specifying the number of raw data points.

**SIGF1 - low field (left) limit of the signal region**

- used in 1D and 2D data sets
- takes a float value (ppm), must be greater than SIGF2
- interpreted by **sino**
- If SIGF1 = SIGF2, the signal region is defined by the entire spectrum minus the first 16th part or, if the scaling region file exists, by the regions in this file. The name of the scaling region file is NUC1.SOLVENT where NUC1 and SOLVENT are acquisition status parameters.
- SIGF1 is also used in 2D data sets as the low field limit for 2D baseline correction by **abst2**, **abst1**, **absot2**, **absot1**, **zert1**, and **zert2**.

**SIGF2 - high field (right) limit of the signal region**

- used in 1D and 2D data sets
- takes a float value (ppm), must be smaller than SIGF1
- interpreted by **sino**

- If SIGF1 = SIGF2, the signal region is defined by the entire spectrum minus the first 16th part or, if the scaling region file exists, by the regions in this file. The scaling region file is defined as NUC1.SOLVENT where NUC1 and SOLVENT are acquisition status parameters.
- SIGF2 is also used in 2D data sets as the high field limit for 2D baseline correction by **abst2**, **abst1**, **absot2**, **absot1**, **zert1**, and **zert2**.

**SINO - signal to noise ratio**

- used in 1D data sets
- takes a float value
- used in AU as an acquisition criterion (not used by processing commands)
- the processing parameter SINO (set with **edp**) can be used in an AU program to specify a signal/noise ratio which must be reached in an acquisition. The acquisition runs until the value of SINO is reached and then it stops. An example of such an AU program is **au_zgsino**. SINO can be set with **edp** but not from the command line. The reason is that entering **sino** on the command line would execute the command **sino**. Note that the processing parameter SINO (**edp**) has a different purpose than the processing status parameter SINO (**dpp**). The latter represents the signal to noise ratio calculated by the processing command **sino**.

**SREGLST - name of the scaling region file**

- used in 1D data sets
- takes a character string value
- interpreted by **li**, **lipp*** if PSCAL = sreg or psreg
- interpreted by **sino**
- scaling region files contain the regions in which the reference peak is searched. They are used to exclude the region in which the solvent peak is expected. Because this region is nucleus and solvent specific the name of a scaling region file is of the form NUCLEUS.SOLVENT, e.g. 1H.CDCl3. For all common nucleus/solvent combinations, a signal region is delivered with TopSpin. They can be viewed or edited with **edsolv**.

**SSB - sine bell shift**

- used in 1D, 2D and 3D data sets in all directions
- takes a positive float value
- interpreted by **sinm**, **qsin**, **sinc**, **qsinc**
- interpreted by **trf**, **xfb**, **xf2**, **xf1**, **xtrf***, **tf*** if WDW = sine, qsine, sinc or qsinc

**SR - spectral reference**

- used in 1D, 2D and 3D data sets in all directions
- takes a float value (Hz)
- set by **sref** or interactive calibration
- The spectral reference is calculated according to the relation:
- SR = SF - BF1

**STSI - strip size: number of output points of strip transform**

- used in 1D, 2D and 3D data sets in all directions
- takes an integer value between 0 and SI (default 0)
- interpreted **ft**, **trf**, **xfb**, **xf2**, **xf1**, **xtrf**, **xtrf2**, **tf3**, **tf2**, **tf1**

- During strip transform, only the region determined by STSI and STSR is stored. For STSI = 0, a normal (full) transform is done. STSI is always rounded; in 1D to the next lower multiple of 4, in 2D and 3D to the next higher multiple of 16. Furthermore, when the 2D (3D) data are stored in submatrix (subcube) format, STSI is rounded to the next multiple of the submatrix (subcube) size.

**STSR - strip start: first output point of a strip transform**

- used in 1D, 2D and 3D data sets in all directions
- takes an integer value between 0 and SI (default 0)
- interpreted **ft**, **xfb**, **xf2**, **xf1**, **xtrf**, **xtrf2**, **tf3**, **tf2**, **tf1**
- During strip transform, only the region determined by STSI and STSR is stored.

**TDeff - number of raw data points to be used for processing**

- used in 1D, 2D and 3D data sets in all directions
- takes an integer value between 0 and TD (default is 0 which means all)
- interpreted by processing commands which work on the raw data
- The first TDeff raw data points are used for processing. For TDeff = 0, all points are used, with a maximum of 2*SI.

**TDoff - number of raw data points ignored or predicted**

- used in 1D, 2D and 3D data sets in all directions
- integer value between 0 and TD (default is 0)
- interpreted by 2D and 3D processing commands which work on raw data
- The first raw data point that contributes to processing is shifted by TDoff points. For 0 < TDoff < TD the first TDoff raw data points are cut off at the beginning and TDoff zeroes are appended at the end (corresponds to left shift). For TDoff < 0, -TDoff zeroes are prepended at the beginning and:
  - for SI < (TD-TDoff)/2 raw data are cut off at the end
  - for DIGMOD=digital, the zeroes would be prepended to the group delay which does not make sense. Avoid that by converting the raw data with **convdta** before processing them.
- also interpreted by 1D, 2D and 3D processing commands which do linear backward prediction, i.e. **ft**, **xfb** of **tf3** when ME_mod is lpbr or lpbc.
- For TDoff > 0, the first TDoff points are replaced by predicted points. For TDoff < 0, abs(TDoff) predicted points are added to the beginning and cut off at the end of the raw data. If zero filling occurs (2*SI > TD), then only zeroes are cut off at the end as long as abs(TDoff) < 2*SI - TD. Note that digitally filtered Avance data start with a group delay. This means that a backward prediction does not make sense unless the data are first converted AMX format with **convdta**.

**TM1 - the end of the rising edge of a trapezoidal window**

- used in 1D, 2D and 3D data sets in all directions
- takes a float value between 0.0 and 1.0
- interpreted by **tm**
- TM1 represents a fraction of the acquisition time and must be smaller than TM2

**TM2 - the start of the falling edge of a trapezoidal window**

- used in 1D, 2D and 3D data sets in all directions
- takes a float value between 0.0 and 1.0

- interpreted by **tm**
- TM2 represents a fraction of the acquisition time and must be greater than TM1.

**TOPLEV - highest 2D contour level**

- used in 2D data sets in the F2 direction
- takes a float value between 0 and 100 (default is 100%)
- interpreted by **levcalc**
- TOPLEV is a percentage of the maximum intensity in the spectrum as expressed by the processing status parameter YMAX_p. For TOPLEV = 0, the highest level is set to 85% of the maximum intensity.

**WDW - FID window multiplication mode**

- used in 1D, 2D and 3D data sets in all directions
- takes one of the values no, em, gm, sine, qsine, trap, user, sinc, qsinc, traf, trafs
- interpreted by **trf**, **xfb**, **xf2**, **xf1**, **xtrf***, **tf***
- On 1D data, window multiplication is usually done with commands like **em**, **gm**, **sinm** etc. which do not interpret WDW. These commands are already specific for one type of window multiplication. The values of WDW have the following meaning:

| WDW value | Function | Dependent parameters | Specific 1D command |
|---|---|---|---|
| em | Exponential | LB | **em** |
| gm | Gaussian | GB, LB | **gm** |
| sine | Sine | SSB | **sinm** |
| qsine | Sine squared | SSB | **qsin** |
| trap | Trapezoidal | TM2, TM1 | **tm** |
| sinc | Sine | SSB, GB | **sinc** |
| qsinc | Sine squared | SSB, GB | **qsinc** |
| traf | Traficante (JMR, **71**, 1987, 237) | | **traf** |
| *trafs* | Traficante (JMR, **71**, 1987, 237) | | **trafs** |

## 2.5 Processing Status Parameters

After processing, most processing status parameters have been set to the same value as the corresponding processing parameter. For some processing status parameters, however, this is different. The reason can be that:

- the corresponding processing parameter does not exist, e.g. NC_proc
- the corresponding processing parameter is not interpreted, e.g. FT_mod
- the value of the corresponding processing parameter is adjusted, e.g. STSI

These type of processing status parameters are listed below and described as output parameters for each processing command. They can be viewed with **dpp** (see also section *About TopSpin Parameters* [▶ 18]).

**BYTORDP - byte order of the processed data**

- used in 1D, 2D and 3D datasets in the first direction
- takes the value *little* or *big*
- set by the first processing command
- interpreted by various processing commands
- Big endian and little endian are terms that describe the order in which a sequence of bytes are stored in a 4-byte integer. Big endian means the most significant byte is stored first, i.e. at the lowest storage address. Little-endian means the least significant byte is stored first. TopSpin only runs on computers with byte order little endian. However, TopSpin's predecessor XWIN-NMR also runs on SGI workstations which are big endian. The byte order of the raw data is determined by the computer which controls the spectrometer and is stored in the acquisition status parameter BYTORDA (type **s bytorda**). This allows raw data to be processed on computers of the same or different storage types. The first processing command interprets BYTORDA, stores the processed data in the byte order of the computer on which it runs and sets the processing status parameter BYTORDP accordingly (type **s bytordp**). All further processing commands interpret this status parameter and store the data accordingly. As such, the byte order of the computer is handled automatically and is user transparent. 2D and 3D processing commands, however, allow to store the processed data with a byte order different from the computer on which they run. For example, the commands **xfb big** and **tf3 big** on a Windows or Linux PC store the data in big endian although the computer is little endian. The processing status parameter BYTORDP is set accordingly.

**FT_mod - Fourier transform mode**

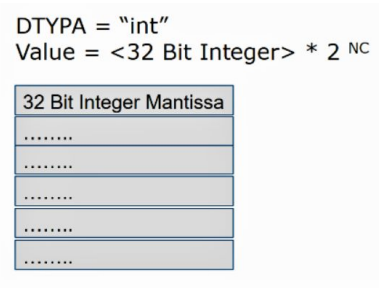
- used in 1D, 2D and 3D datasets in all directions
- takes one of the values *no*, *fsr, fqr*, *fsc*, *fqc*, *isr*, *iqr*, *iqc*, *isc*
- set by all Fourier transform commands, e.g. **ft**, **trf**, **xfb**, **xf2**, **xf1**, **trf***, **xtrf***, **tf3**, **tf2**, **tf1**
- interpreted by **trf** and **xtrf***.
- also exists as processing (**edp**) parameter (interpreted by **trf** and **xtrf***)
- The values of FT_mod are described in chapter *List of processing parameters [▶ 20]*.

**MC2 - Fourier transform mode of the second (and third) direction**

- is used in 2D datasets in the second direction (F1)
- is used in 3D datasets in the second and third direction (F2 and F1)
- takes one of the values *QF*, *QSEQ*, *TPPI*, *States*, *States-TPPI*, *echo-antiecho*
- is set by **xfb**, **xf2**, **xf1**, **xtrf***, **tf***
- is interpreted by **xf1**, **xtrf1**, **tf2**, **tf1**
- The processing status parameter MC2 is set according to the acquisition status parameter FnMODE. If, however, FnMODE = undefined, the processing status parameter MC2 is set according to the processing parameter MC2. Furthermore, status MC2 is interpreted during 2D processing in F1, on processed data, for example by **xf1** on data which have already been processed with **xf2**.

**NC_proc - intensity scaling factor**

Processing in TopSpin performs calculations in double precision floating point. The processing status parameter DTYPP defines, how the data values are stored. If the DTYPP is 0 ("int"), the stored value represents a mantissa of the data point value, the acquisition status parameter NC_proc is the exponent. All data points share in this case the same exponent.




- takes an integer value
- set by all processing commands
- only exists as processing status parameter
- During double to integer conversion, the data are scaled up or down such that the highest intensity of the spectrum lies between $2^{28}$ and $2^{29}$. This means the 32 bit resolution is not entirely used. This allows for the highest intensity to be increased, for example during phase correction, without causing data overflow. NC_proc shows the amount of scaling that was done, for example:
  - NC_proc = -3 : data were scaled up (multiplied by 2) three times
  - NC_proc = 4 : the data were scaled down (divided by 2) four times
- Although NC_proc is normally calculated by processing commands, 2D processing also allows to predefine the scaling factor with the argument **nc_proc**, for example, **xfb nc_proc 2**, scales down the data twice. However, you can only scale the data more down (or less up) than the command would have done without the argument **nc_proc**. The latter is shown by the processing status parameter NC_proc (type **dpp**). Smaller (more negative) values of **nc_proc** are ignored to avoid data overflow. The command **xfb nc_proc last** takes the current value of the processing status parameter NC_proc (type **dpp**) as input value.

**PPARMOD - dimensionality of the processed data**

- takes one of the values 1D, 2D,..., 8D
- interpreted by TopSpin display, parameter editor **edp** and processing commands that access processed data like **abs** and **apk**.
- can be set by changing the dimension from the parameter editor (**edp**) toolbar.
- The status parameter PPARMOD defines the dimensionality of the processed data. Note the following restriction: PPARMOD <= PARMODE.

**PHC0 - zero order phase correction value (frequency independent)**

- used in 1D, 2D and 3D datasets in all directions
- takes a float value (degrees)
- set by **apk**, **apks**, **apkf**, **apk0**, **apk0f, apkm** in 1D datasets
- set interactively in Phase correction mode in 1D and 2D datasets
- also exists as processing parameter (**edp**)
- PHC0 is one of the few examples where a processing parameter is set by a processing command. For example, **apk** sets both the processing and processing status parameter PHC0. **pk** reads the processing parameter and updates the processing status parameter. After multiple phase corrections, the processing status parameter PHC0 shows the total zero order phase correction.

**PHC1 - first order phase correction value (frequency dependent)**

- used in 1D, 2D and 3D datasets in all directions
- takes a float value (degrees)

- set by **apk**, **apks**, **apkf**, **apk1, apkm** in 1D datasets
- set interactively in Phase correction mode in 1D and 2D datasets
- also exists as processing parameter (**edp**)
- PHC1 is one of the few examples where a processing parameter is set by a processing command. For example, **apk** sets both the processing and processing status parameter PHC1. **pk** reads the processing parameter and updates the processing status parameter. For multiple phase corrections, the processing status parameter PHC1 shows the total first order phase correction.

**S_DEV - standard deviation of the processed data**

- used in 2D and 3D datasets in the first direction
- takes a float value
- set by all processing commands, e.g. **xfb**, **xfbp**, **abs2**, **tf***, **tabs***
- interpreted by **levcalc**
- only exists as processing status parameter (**dpp**)

**SINO - signal to noise ratio**

- used in 1D datasets
- takes a float value
- set by **sino**
- also exists as processing parameter
- The signal is determined in the region between SIGF2 and SIGF1. The noise is determined in the region between NOISF2 and NOISF1. Note that SINO also exists as a processing parameter (**edp**) which has a different purpose (see chapter List of processing parameters)

**SW_p - spectral width of the processed data**

- used in 1D, 2D and 3D datasets in all directions
- takes a double value
- set by all processing commands
- only exists as processing status parameter
- Normally, SW_p will be the same as the acquisition status parameter SW. However, in case of stripped data (see processing commands STSR and STSI), the processing spectral width differs from the acquired spectral width.

**SYMM - 2D symmetrization type done**

- used in 2D datasets in the F2 direction
- takes the value *no*, *sym*, *syma* or *symj*
- set by **sym**, **syma** and **symj**
- only exists as processing status parameter (**dpp**)
- SYMM shows the (last) kind of symmetrization that was done.

**STSI - strip size; the number of output points of a strip transform**

- used in 1D, 2D and 3D datasets in all directions
- takes an integer value between 0 and SI (default 0)
- also exists as processing parameter (**edp**)
- rounded by **ft**, **trf**, **xfb**, **xf2**, **xf1**, **xtrf**, **xtrf2**, **tf3**, **tf2**, **tf1**

- During strip transform, only the region determined by STSI and STSR is stored. Processing commands round the value of the processing parameter STSI; in 1D to the next lower multiple of 4, in 2D and 3D to the next higher multiple of 16 (see processing command STSI). Furthermore, when the 2D (3D) data are stored in submatrix (subcube) format, STSI is rounded to the next multiple of the submatrix (subcube) size. The rounded value is stored as the processing status parameter STSI. If no strip transform is done (STSI = 0), the status STSI is set to the value of SI.

**TDeff - number of raw data points that were used for processing**

- used in 1D, 2D and 3D datasets in all directions
- set by **ft**, **xfb**, **xf2**, **xf1**, **trf***, **xtrf***
- also exists as processing parameter (**edp**)
- Normally, all raw data points are used as input. However, the number of input points can be decreased with the processing parameter TDeff or increased by doing linear forward or backward prediction with TDoff < 0. The number of raw data points that were actually used is stored in the processing status parameter TDeff.

**TILT - flag indicating whether a tilt command has been performed**

- used in 2D datasets in the F2 direction
- takes the value TRUE or FALSE
- set by **ptilt**, **ptilt1** or **tilt**
- only exists as processing status parameter (**dpp**)

**XDIM - submatrix or subcube size**

- used in 2D and 3D datasets in all directions
- takes an integer value
- set by **xfb**, **xf2**, **xf1**, **xtrf**, **xtrf2**, **tf3**
- also exists as processing parameter
- Although XDIM is normally calculated by processing commands, 2D and 3D processing also allow to predefine the submatrix sizes, using the argument **xdim**:
  - On a 2D dataset, the command **xfb xdim** interprets the processing parameter XDIM in both F2 and F1.
  - On a 3D dataset, the command **tf3 xdim** interprets the processing parameter XDIM in F3, F2 and F1.

**FTSIZE - Fourier transform size**

- used in 1D, 2D and 3D datasets in all directions
- takes an integer value
- set by all processing command that perform Fourier transform
- Normally, the status parameter FSIZE has the same value as the status parameter SI. Only in case of strip transform (STSR > 0 and/or STSI > 0), they are different. FTSIZE then represents the size with which the raw data were Fourier transformed whereas SI represents the size with which the processed data are stored.

**YMAX_p - maximum intensity of the processed data**

- used in 1D, 2D and 3D datasets in the first direction
- takes an integer value
- set by all processing commands
- only exists as processing status parameter (**dpp**)

**YMIN_p - minimum intensity of the processed data**

- used in 1D, 2D and 3D datasets in the first direction
- takes an integer value
- set by all processing commands
- only exists as processing status parameter (**dpp**)

## 2.6 Relaxation Parameters

Relaxation parameters can be set with the command **edt1** which can be entered from the Relaxation menu.

**COMPNO - number of components contributing to the relaxation curve**

- used in pseudo 2D relaxation data sets
- takes an integer value (default is 1)
- interpreted by **simfit**
- Peak positions are determined on a row which is specified by the parameter START (usually the first row). These positions are then used by **pd** for each row of the 2D data. However, peak positions sometimes drifts in the course of the experiment, i.e. they might shift one or more points in successive rows. Therefore, **pd** searches for the maximum intensity at the predefined peak position plus or minus DRIFT.

**DRIFT - drift of the peak positions in the course of the experiment**

- used in pseudo 2D relaxation data sets
- takes an integer value (must be 1 or greater, default is 5)
- interpreted by **pd**
- Relaxation analysis is usually done with a series of relaxation curves, one for each peak in the spectrum. One curve shows the intensity distribution of one peak over a series of experiments, i.e. a series of rows in a pseudo 2D data set. First the peak positions are determined on one row, for example with **ppt1**. Then the command **pd** determines the intensity at these positions in each row. However, peak positions sometimes drifts in the course of the experiment, i.e. they can be slightly different in different rows. Therefore, **pd** searches for the maximum intensity in a range around a each peak position. This range is determined by the parameter DRIFT.

**EDGUESS - table of initial values and step rates of the function variables**

- used in pseudo 2D relaxation data sets
- interpreted by **simfit**
- The EDGUESS table shows all variables of the function specified by FCTTYPE. For each variable, the initial guess (G) and step rate (S) can be set for each component (C). The table below shows the EDGUESS table for an inversion recovery experiment, with 2 components. The initial guess for I[0] must be such that the total value of all components does not exceed 1. If there is only one component, I[0] is usually set to 1. The step rate is usually set to about one tenth or the initial guess. If the step rate of a variable is set to zero, then this variable is not changed during the iterations. Note that the commands **ct1**, **ct2**, **dat1** or **dat2** do not use the EDGUESS table. They calculate the initial values and step rates of the T1/T2 function variables I[0], P and T1.

| | | | |
|---|---|---|---|
| GC1I0 | 0.5 | SC1I0 | 0.05 |
| GC1A | 1.0 | SC1A | 0.1 |

| | | | |
|---|---|---|---|
| GC1T1 | 2.0 | SC1T1 | 0.2 |
| GC2I0 | 0.5 | SC2I0 | 0.05 |
| GC2A | 1.0 | SC2A | 0.1 |
| GC2T1 | 2.0 | SC2T1 | 0.2 |

**FCTTYPE - function type used for fitting the relaxation curve**

- used in pseudo 2D relaxation data sets
- takes one of the values listed in the next table.
- interpreted by **simfit**
- The table below shows the experiment types which **simfit** can handle and the corresponding fit functions. Note that **ct1**, **ct2**, **dat1** and **dat2** do not evaluate FCTTYPE because they can only handle T1/T2 experiments. They do, however, set FTCTYPE to the value *t1/t2.*

| Exp. Type | Comp | Fit function |
|---|---|---|
| uxnmrt1t2 | 1 | I[t] = I[0]+P*exp(t/T1) |
| invrec | 1 - 4 | I[t] = I[0]*(1-2A*exp(-t/T1)) |
| satrec | 1 - 6 | I[t] = I[0]*(1-exp(-t/T1)) |
| cpt1rho | 1 - 4 | I[t] = I[0]/(1-TIS/T1rho)*(exp(-t/T1rho)-exp(t/TIS)) |
| expdec | 1 - 6 | I[t] = I[0]*exp(-t/T) |
| gaussdec | 1 - 6 | I[t] = I[0]*exp(-SQR(t/T)) |
| lorgauss | 1 - 3 | I[t] = IL*exp(-t/TL)+IG*exp(-SQR(t/TG)) |
| linear | 1 - 6 | I[t] = A+B*t |
| varbigdel | 1 - 6 | I = I[0]*exp(-D*SQR(2*PI*gamma*G*LD)*(BD-LD/3)*1e4) |
| varlitdel | 1 - 6 | I = I[0]*exp(-D*SQR(2*PI*gamma*G*LD)*(BD-LD/3)*1e4) |
| vargrad | 1 - 6 | I = I[0]*exp(-D*SQR(2*PI*gamma*G*LD)*(BD-LD/3)*1e4) |
| raddamp | 1 - 6 | MZ[t]=A0+MZ[0]*tanh((t-T0)/TRD) |

- used in pseudo 2D relaxation data sets
- takes the value *area* or *intensity* (default is *intensity*)
- interpreted by **pd**, **ct1**, **dat1** and **simfit**
- Before running **pd**, both the integral ranges and peak positions should be determined (see **rspc** and **ppt1**). **pd** then picks the points storing both their integrals and intensities but it only displays one curve; the one defined by FITTYP. **ct1** or **simfit** then calculate the relaxation value for one peak according to FITTYPE. You can change FITTYP and recalculate the relaxation value without running **pd** again. The same counts for the commands **dat1** and **simfit all** which fit all peaks.

**INC - point (1D) or row (2D) increment**

- used in 1D and pseudo 2D relaxation data sets
- takes an integer value (default is 1)
- interpreted by **pft2** (1D data)
- interpreted by **pd** (pseudo 2D data)

- Starting with START, every INC point (1D) or row (pseudo 2D) is used for relaxation analysis.

**NUMPNTS - number of data points used for relaxation analysis**

- used in 1D and pseudo 2D relaxation data sets
- takes an integer value (default is TD)
- interpreted by **pft2** (1D)
- interpreted by **pd** (pseudo 2D)
- The default value of NUMPNTS is the number of available points, i.e. TD (1D) or F1 TD (pseudo 2D). TD is the acquisition status parameter which can be viewed with **dpa** or **s td**. Note that if you increase INC, you must reduce NUMPNTS such that INC*NUMPNTS does not exceed TD.

**START - first point (1D) or row (2D) used for relaxation analysis**

- used in 1D and pseudo 2D relaxation data sets
- takes an integer value (default is 1)
- interpreted by **pft2** (1D data)
- interpreted by **pd** (pseudo 2D data)
- Note that the default value (1) is not the first but the second point of a 1D data set. It is, however, the first row of a pseudo 2D data set. The point or row used is START + n*INC.

# 3 1D Processing Commands

This chapter describes all TopSpin 1D processing commands. Several of them can also be used to process one row of 2D or 3D data. They store their output in processed data files and do not change the raw data.

For each command, the relevant input and output parameters are mentioned. Furthermore, the relevant input and output files and their location are mentioned. Although file handling is completely transparent, it is sometimes useful to know which files are involved and where they reside. For example, if you have permission problems or if you want to process or interpret your data with third party software.

## 3.1 abs, absf, absd, bas

### NAME

abs - Automatic baseline correction (1D)

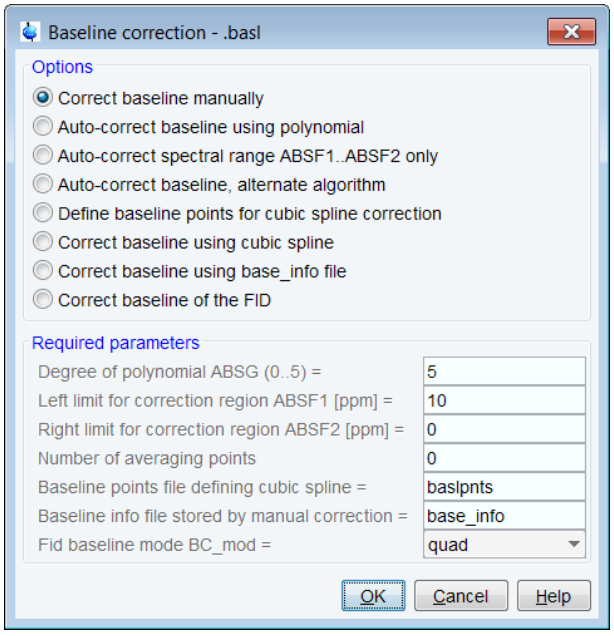
absf - Automatic baseline correction of the plot region (1D)

absd - Automatic baseline correction, special algorithm (1D)

bas - Open baseline correction dialog box (1D)

### DESCRIPTION

Baseline correction commands can be started on the command line or from the baseline correction dialog box.




The latter is opened with the command **bas**.

This dialog box offers several options, each of which selects a certain command for execution.

### Auto-correct baseline using polynomial

This option selects the command **abs** for execution. It performs an automatic baseline correction of the spectrum by subtracting a polynomial. The degree of the polynomial is determined by the parameter ABSG which has a value between 0 and 5, with a default of 5. **abs** first determines which parts of the spectrum contain spectral information and stores the result in the file *intrng* (integral regions). The remaining part of the spectrum is considered baseline and used to fit the polynomial function.

**abs** also interprets the parameters ABSL, AZFW, AZFE and ISEN. Since these parameters apply to integration rather than baseline correction, they do not appear in the **bas** dialog box. They do appear in the integration dialog box (command **int**). Data points greater than ABSL*( standard deviation) are considered spectral information, all other points are considered noise. If two peaks are more than AZFW apart, they are treated independently. If they are less than AZFW ppm apart, they are considered to be overlapping. Integral regions are extended at both sides by AZFE ppm. If this extension causes adjacent regions to overlap, the centre of the overlap is used as the limit of the two regions. Only regions whose integrals are larger (area) than the largest integral divided by ISEN are considered.

**abs n** does not store the integral regions.

The command abs only stores the integral regions of positive peaks. To store the integral regions of both positive and negative peaks, following command sequence can be used: **ef**, **mc**, **abs**, **efp**, **abs n**.

### Auto-correct spectral range ABSF1 .. ABSF2 only

This option selects the command **absf** for execution. It works like **abs**, except that it only corrects the spectral region which is determined by the processing parameters ABSF1 and ABSF2.

### Auto-correct baseline, alternate algorithm

This option selects the command **absd** for execution. It works like **abs**, except that it uses a different algorithm (It uses the same algorithm as the command abs in DISNMR). It is, for example, used when a small peak lies on the foot of a large peak. In that case, **absd** allows to correct the baseline around the small peak which can then be integrated. Usually **absd** is followed by **abs**.

To display the integral regions determined by one of the above commands:

1. Right-click inside the data window and select *Display Properties*
2. Check the entry *Integrals* and click **OK**

The integral regions are also used by various commands which calculate spectral integrals like **li**, **lipp** and **plot**.

If you run a command like **abs** from the command line, you have to make sure that the required parameters are already set. Click the Procpars tab or enter **edp** to do that.

If automatic baseline correction does not give satisfactory results, you can apply an interactively determined polynomial, exponential, sine or spline baseline correction. This can be started with the first entry of the **bas** dialog box, by clicking the  button in the toolbar or by entering **.basl** on the command line.

The **bas** command can be used on 1D or 2D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

### INPUT PARAMETERS

Set from the **bas** dialog box, with **edp** or by typing **absg**, **absf1** etc.:

ABSG - degree of the polynomial (input of **abs**, **absf**, **absd**)

ABSF1 - low field (left) limit of the region corrected by **absf**

ABSF2 - high field (right) limit of the region corrected by **absf**

Set from the **int** dialog box, with **edp** or by typing **absl**, **azfw** etc.:

ABSL - integral sensitivity factor with reference to the noise

AZFW - minimum distance between peaks for independent integration

AZFE - integral extension factor

ISEN - integral sensitivity factor with reference to the largest integral

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r* - real processed 1D data

*proc* - processing parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r* - real processed 1D data

*procs* - processing status parameters

*intrng* - integral regions (output of **abs**, **absf**, **absd**)

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

ABS

ABSD

ABSF

**SEE ALSO**

*bcm* [▷ 55], *sab* [▷ 84], *bc* [▷ 53]; *apbk* [▷ 48], (.basl), *sigreg* [▷ 75]

## 3.2 add, duadd, addfid, addc, adsu

**NAME**

add - Add two data sets point-wise, multiply 2nd with DC (1D)

duadd - Add two data sets ppm/Hz-wise, mult. 2nd with DC (1D)

addfid - Add two FIDs, multiply 2nd with DC (1D)

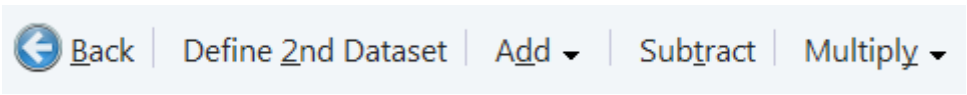
addc - Add the constant DC to the current data set

adsu - Open add/subtract/multiply dialog box (1D, 2D)

**DESCRIPTION**

Addition commands can be entered on the command line or started from the add/subtract/multiply dialog box. The latter is opened with the command **adsu**.

This dialog box offers several options, each of which selects a certain command for execution.



### Add a 1D spectrum point-wise

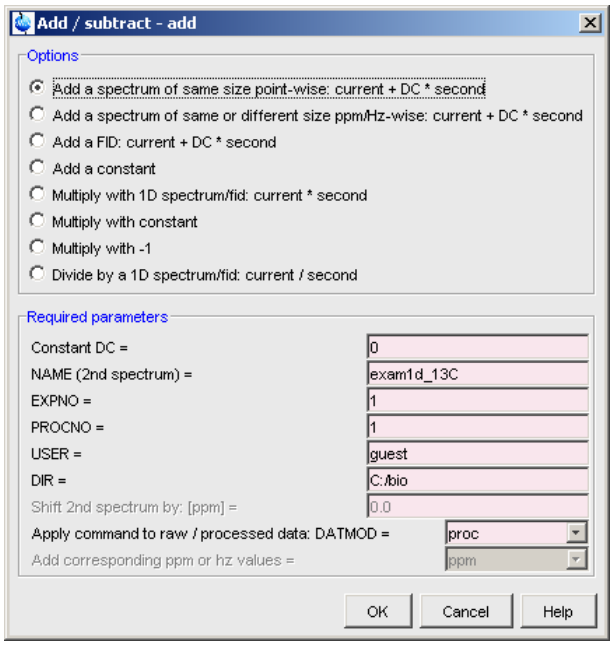
This option selects the command **add** for execution. It adds the second data set, multiplied with the constant DC, to the current data set. **add** performs a point to point addition which is independent of the spectrum calibration.The result is stored in the current data set. DC can be set by entering **dc** on the command line or in the *Procpars* pane. If the second data set has not been defined yet, the add/subtract dialog box is opened. Here you can define the second data set and start the **add** command. **add** works on raw or on processed data, depending on the value of DATMOD. For DATMOD = raw, **add** adds the raw data of the current and second data set but stores the result as processed data in the current data set. The raw data of the current data set are not overwritten.

### Add a 1D spectrum ppm/Hz-wise

This option selects the command **duadd** for execution. It works like **add**, except that it adds two data sets according to their chemical shift values. Each ppm value of one data set is added to the same ppm value of a second data set.

**duadd** is useful when the two input spectra are:

- of different size
- referenced differently
- acquired with different frequencies (i.e. on different spectrometers)




For data with equal size, reference and spectrometer frequency, **add** and **duadd** give the same result.

Furthermore, **duadd** allows to shift the second spectrum by a user defined number of ppm. The parameter *ppm* or *hz* is only relevant if the input data were acquired with different basic frequencies, i.e. when they come from different spectrometers. **duadd** only works on processed data, independent of the value of DATMOD.

### Add an FID

This option selects the command **addfid** for execution. It adds two 1D raw data sets multiplying one of them with the factor DC. The result is stored in the current data set. It works like **add** with DATMOD = raw, except that it overwrites the raw data.

### Add a constant

This option selects the command **addc** for execution. It adds the value of DC to the current data set. It works on raw or processed data, depending on the value of DATMOD. The result is stored as processed data in the current data set.

If you run a command like **add** from the command line, it behaves slightly different. It adds the second and the third data set, as specified with **edc2** and stores the result in the current data set. You have to make sure that the required parameters are already set. Click the Procpars tab or enter **edp** to do that.

The **adsu** command can be used on 1D or 2D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

### INPUT PARAMETERS

Set from the **adsu** dialog box, with **edp** or by typing **dc**, **datmod** etc.:

DC - multiplication factor

DATMOD - data mode: work on raw or processed data

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - current raw data (input of **add**/**addc** if DATMOD = raw)

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - current processed data (input of **add**/**addc** if DATMOD = proc)

*proc* - processing parameters

curdat2 - definition of the second data set

*<dir2>/data/<user2>/nmr/<name2>/<expno2>/*

*fid* - second raw data (input of **add** if DATMOD = raw, **addfid**)

*<dir2>/data/<user2>/nmr/<name2>/<expno2>/pdata/<procno2>/*

*1r*, *1i* - second processed data (input of **add** if DATMOD = proc)

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - current raw data (output of **addfid**)

*audita.txt* - acquisition audit trail (output of **addfid**)

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - current processed data (output of **add** and **addc**)

*procs* - processing status parameters

*auditp.txt* - processing audit trail (output of **add** and **addc**)

### USAGE IN AU PROGRAMS

ADD

ADDFID

ADDC

### SEE ALSO

*add2d, mul2d, addser [▶ 99]*, *mul, mulc, nm, div [▶ 69]*

## 3.3 accumulate

### NAME

accumulate - Accumulate 1D datasets ppm/Hz-wise (1D)

### SYNTAX

accumulate [start] offset scale Hz|ppm procno [expno [name [user [dir]]]]

### DESCRIPTION

The command **accumulate** accumulates 1D datasets. It adds a specified processed dataset to the current dataset. **accumulate** has the following features:

- the specified data can be shifted and scaled with respect to the current data.
- addition can be performed ppm-wise or hz-wise
- the specified data can overwrite the current data or can be added to the current data

All required information must be specified by command line arguments. As such, **accumulate** takes 4 to 9 arguments. Here are some examples of its usage:

**accumulate <offset> <scale> ppm |hz <procno>**

Add the processed data of the specified *procno* to the current *procno* as follows:

- shift the added data by <offset> ppm
- scale added data by the value <scale>
- perform the addition ppm-wise or hz-wise as specified

Example: **accumulate 0.0 1.0 ppm 3**

**accumulate start <offset> <scale> ppm |hz <procno>**

Same as above, except that the processed data of the specified *procno* are copied to the current *procno*, overwriting possibly existing data.

Example: **accumulate start 0.0 1.0 ppm 3**

Note that here, the arguments **offset** and **ppm |hz** do not affect the data but do affect the status parameter OFFSET.

In the examples above, the accumulated dataset has the same datapath as the original data except for the *procno*. To accumulate data with a different datapath, you can specify other parts of the datapath as arguments. Parts that are not specified are taken from the current dataset.

Examples:

**accumulate <offset> <scale> ppm |hz <procno> <expno>**

**accumulate start <offset> <scale> ppm |hz <procno> <expno> <user> <dir>**

**accumulate** works like the command **duadd**, except that all information is specified on the command line. **accumulate** is typically used repeatedly to accumulate a series of 1D processed data. The first instance of **accumulate** overwrites the current data with the specified data, defining the accumulation start. All further instances add the specified data to the current data.

### OUTPUT PARAMETERS

OFFSET - the ppm value of the first data point of the spectrum

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - current processed data

*proc* - processing parameters

*<dir2>/data/<user2>/nmr/<name2>/<expno2>/pdata/<procno2>/*

*1r*, *1i* - second processed data

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - current processed data

*procs* - processing status parameters

*auditp.txt* - processing audit trail

**SEE ALSO**

*add, duadd, addfid, addc, adsu [▶ 44]*

## 3.4 apbk

**Name**

**apbk** – Combined baseline and phase correction

**DESCRIPTION**

The command **apbk** provides simultaneous linear phase and baseline correction of 1D spectra. The implementation consists of different algorithms that were developed to ensure optimal performance on spectra of different nuclei. The algorithm selection is performed automatically when the **apbk** command is run.

The **apbk** algorithm that was developed and tested for $^{13}C$, $^{19}F$, $^{31}P$, $^{11}B$, $^{15}N$ and $^{29}Si$ spectra consists of a phase correction combined with a model-free baseline correction method. This allows better and more flexible baseline correction with respect to the **abs** command which uses a polynomial for baseline modeling.

For proton spectra, this approach was not successful due to the high signal density and the smaller spectral width typical of $^{1}H$ spectra. Therefore, another algorithm was developed and tested for $^{1}H$ spectra acquired without solvent suppression that relies on a deep neural network trained for baseline detection.

When the **apbk** command is applied to any other 1D spectrum that does not fall into the categories listed above, **apk** and **abs** are executed instead to correct the phase and baseline.

**USAGE**

**apbk** - Correct baseline and phase, write integration regions to disk.

**apbk –bo** - Correct only baseline.

**apbk –po** - Correct only phase.

**apbk –n -** Correct baseline and phase, do not write integration regions to disk.

**apbk –intrng** - Correct baseline and phase using pre-defined integration regions as input. Note: the integration regions will be replaced with new integration regions unless the -n option is added.

**apbk –f** - Enforce the use of the **apbk** algorithms. If this option is chosen for a $^{1}H$ spectrum, the deep learning based algorithm is applied, even if the spectrum was acquired with solvent suppression. If used on spectra of any other nucleus, the other **apbk** algorithm is enforced, even if the nucleus is not in the list of supported X-nuclei.

Options only available for $^{1}H$ spectra:

**apbk -apk0** - Correct only the zero-order phase.

**INPUT FILES**

*<dir>/<name>/<expno>/pdata/<procno>/*

1r, 1i - processed 1D data (frequency domain)

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r, 1i* - processed 1D data (real, imaginary)

*proc* - processing parameters

*procs* - processing status parameters

*intrng* - integral regions

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

ABPK

**SEE ALSO**

abs, absf, absd, bas, apk, apks, apkm, apkf, ph

## 3.5 apk0, apk1, apk0f

**NAME**

apk0 - Zero-order automatic phase correction (1D)

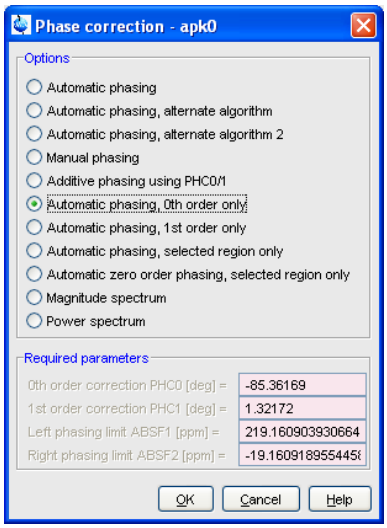
apk1 - First-order automatic phase correction (1D)

apk0f - Customized zero-order automatic phase correction (1D)

ph - Open phase correction dialog box (1D/2D)

**DESCRIPTION**

Phase correction commands can be entered on the command line or started from the phase correction dialog box:




This dialog is opened with the command **ph**. It offers several options, each of which selects a certain command for execution.

### Automatic phasing, 0th order only

This option selects the command **apk0** for execution. It works like **apk**, except that it only performs the zero order phase correction.

### Automatic phasing, 1st order only

This option selects the command **apk1** for execution. It works like **apk**, except that it only performs the first order phase correction.

### Automatic zero order phasing, selected region order only

This option selects the command **apk0f** for execution. It works like **apkf**, except that it only performs the zero order phase correction.

If you run a command like **apk0f** from the command line, you have to make sure that the required parameters are already set. Click the Procpars tab or enter **edp** to do that.

If automatic phase correction does not give satisfactory results, you can perform interactive phase correction. This can be started with the entry *Manual phasing* in the **ph** dialog box, by clicking the button in the toolbar or by entering **.ph** on the command line.

The **ph** command can be used on 1D, 2D or 3D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

### INPUT PARAMETERS

Set from the **ph** dialog box, with **edp** or by typing **absf1**, **absf2** etc.:

ABSF1 - low field (left) limit of the region used by **apk0f**

ABSF2 - high field (right) limit of the region used by **apk0f**

### OUTPUT PARAMETERS

Can be viewed with **edp**, **dpp** or by typing **phc0**, **sphc0** etc.:

PHC0 - zero order phase correction value (output of **apk0** and **apk0f**)

PHC1 - first order phase correction value (output of **apk1**)

Note that this is one of the rare cases where the output parameters of a command are stored as processing (**edp**) and as processing status parameters (**dpp**).

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

*proc* - processing parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

*proc* - processing parameters

*procs* - processing status parameters

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

APK0

APK1

APK0F

**SEE ALSO**

*apk, apks* [▸ 51], *pk* [▸ 71], *mc* [▸ 68], *ps* [▸ 74]; *apbk* [▸ 48], (.ph)

## 3.6 apk, apks, apkm, apkf, ph

**NAME**

apk - Automatic phase correction (1D)

apks - Automatic phase correction with a different algorithm (1D)

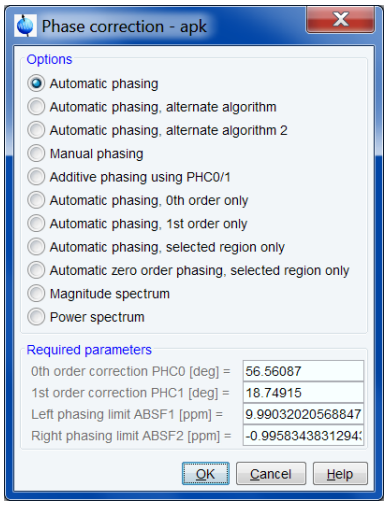
apkm - Automatic phase correction with a different algorithm 2 (1D)

apkf - Customized automatic phase correction (1D)

ph - Open phase correction dialog box (1D/2D)

**DESCRIPTION**

Phase correction commands can be can be entered on the command line or started from the phase correction dialog box. This dialog is opened with the command **ph**. It offers several options, each of which selects a certain command for execution.




### Automatic phasing

This option selects the command **apk** for execution. It calculates the zero and first order phase values and then corrects the spectrum according to these values. The phase values are stored in the parameters PHC0 and PHC1, respectively. Note that **apk** stores the calculated phase values both as processing parameters (**edp**) and as processing status parameters (**dpp**).

### Automatic phasing, alternate algorithm

This option selects the command **apks** for execution. It works like **apk**, except that it uses a different algorithm which gives better results on certain spectra, for instance polymer spectra where peaks are concentrated only in one area.

### Automatic phasing, alternate algorithm 2

This option selects the command **apkm** for execution. It uses symmetric isolated peaks, regions with positive/negative signals and regions of flat baseline for automated phase correction of 1D NMR spectra. The automated phasing is performed by means of minimization of certain penalty function with four terms. The first term is responsible for phases of symmetric isolated peaks, the second accounts for regions with positive/negative signals, the third accounts for baseline regions, and the fourth gives additional penalty for large values of first-order phase correction parameter PHC1. For a full description of **apkm**, enter the TopSpin command **help apkm**.

### Automatic phasing, selected region only

This option selects the command **apkf** for execution. It works like **apk**, except that it uses only a certain region of the spectrum for the calculation of the phase values. This region is determined by the parameters ABSF1 and ABSF2. The calculated phase values are then applied to the entire spectrum. Note that the parameters ABSF1 and ABSF2 are also used by the command **absf**.

If you run a command like **apkf** from the command line, you have to make sure that the required parameters are already set. Click the Procpars tab or enter **edp** to do that.

If automatic phase correction does not give satisfactory results, you can perform interactive phase correction. This can be started with the entry *Manual phasing* in the **ph** dialog box, by clicking the button in the toolbar or by entering **.ph** on the command line.

The **ph** command can be used on 1D or 2D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

### INPUT PARAMETERS

Set from the **ph** dialog box, with **edp** or by typing **absf1**, **absf2** etc.:

ABSF1 - low field (left) limit of the region used by **apkf**

ABSF2 - high field (right) limit of the region used by **apkf**

### OUTPUT PARAMETERS

Can be viewed with **edp**, **dpp** or by typing **phc0**, **s phc0** etc.:

PHC0 - zero order phase correction value (frequency independent)

PHC1 - first order phase correction value (frequency dependent)

Note that this is one of the rare cases where the output parameters of a command are stored as processing (**edp**) and as processing status parameters (**dpp**).

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

*proc* - processing parameters

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

*proc* - processing parameters

*procs* - processing status parameters

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

APK

APKF

APKS

### SEE ALSO

*apk0, apk1, apk0f, ph* [▷ 49]; *apbk* [▷ 48]

## 3.7 bc

### NAME

bc - Baseline correction of the FID (1D)

### DESCRIPTION

The command **bc** performs a baseline correction of raw 1D data. The type of correction is determined by the processing parameter BC_mod as shown in the following table:

| BC_mod | Function subtracted from the FID | Detection mode |
|---|---|---|
| no | no function | |
| single | average intensity of the last quarter of the FID | single channel |
| quad | average intensity of the last quarter of the FID | quadrature |
| spol | polynomial of degree 5 (least square fit) | single channel |

| | | |
|---|---|---|
| qpol | polynomial of degree 5 (least square fit) | quadrature |
| sfil | Gaussian function of width BCFW* | single channel |
| qfil | Gaussian function of width BCFW | Quadrature |
| *Marion, Ikura, Bax, J. Magn. Res. 84, 425-420 (1989) | | |

*spol/qpol* and *sfil/qfil* are especially used to subtract strong signals, e.g. a water signal at the centre of the spectrum. Note that *sfil/qfil* perform a better reduction at the risk of losing valuable signal. For reducing off-centre signal, you can set the parameter COROFFS to the offset frequency.

In this table, *s(ingle)* stands for single detection mode and *q(uad)* for quadrature detection mode. **bc** evaluates BC_mod for the function to be subtracted but not for the detection mode. The latter is evaluated from the acquisition status parameter AQ_mod. This means, for example, it does not matter if you set BC_mod to *single* or *quad*. The same counts for the values *spol*/*qpol* and *sfil/qfil*. Furthermore, for AQ_mod = DQD, no baseline correction is performed for BC_mod = *single* or *quad*. Note that the commands **trf** and **xtrf*** do evaluate the detection mode from BC_mod and perform the baseline correction for BC_mod = *single*/*quad* when AQ_mod = DQD.

The command **bc** is automatically executed as a part of the commands **em**, **gm**, **ft**, or any of the composite Fourier transform commands.

When executed on a 2D or 3D dataset, **bc** prompts you for the row and output *procno*. Alternatively, it can be entered with up to four arguments:

**bc <row> <procno> n y**

processes the specified row and stores it under the specified *procno*.

The last two arguments are optional: **n** prevents changing the display to the output 1D data, **y** causes a possibly existing data to be overwritten without warning.

When executed on a dataset with 2D or 3D raw data but 1D processed data (usually a result of rsr, rsc or a 1D processing command on that 2D or 3D data set), **bc** takes one argument **bc <row>** to process the specified row and stores it under the current *procno*.

**bc same** processes the same row as the previous processing command and stores it under the current *procno*. The **same** option is automatically used by the AU program macro BC. When used on a regular 1D dataset (i.e. with 1D raw data) it has no effect.

**bc** can also be started from the baseline dialog box which is opened with the command **bas**.

## INPUT PARAMETERS

Set from the **bas** dialog box, with **edp** or by typing **bc_mod**, **bcfw** etc.:

BC_mod - FID baseline correction mode

BCFW - filter width for BC_mod = sfil or qfil

COROFFS - correction offset in Hz, for BC_mod = spol or qpol and sfil/qfil

## INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - raw data (time domain)

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*proc* - processing parameters

## OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed data (time domain)

*procs* - processing status parameters

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

BC

### SEE ALSO

(bas)

## 3.8 bcm

### NAME

bcm - User defined spectrum baseline correction (1D)

### DESCRIPTION

The command **bcm** performs a spectrum baseline correction by subtracting a polynomial, sine or exponential function.

This involves the following steps:

1. Click or enter**.basl** to change to baseline correction mode.
2. Fit the baseline of the spectrum with a *polynomial*, *exponential* or *sine* function. Click-hold the button **A** and move the mouse to determine the zero order correction. Do the same with the buttons **B**, **C** etc. for higher order corrections until the line matches the baseline of the spectrum.
3. Click to return. The command **bcm** is automatically executed.

The interactively determined baseline function is stored in the file *base_info*. This file can be stored for general usage with the command **wmisc**. After that, you can read it with **rmisc** on another dataset and run **bcm** to perform the same baseline correction.In this case, **bcm** can be started from the command line or from the baseline dialog box which is opened with the command **bas**.

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r* - real processed 1D data

*proc* - processing parameters

*base_info* - baseline correction coefficients

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r* - real processed 1D data

*procs* - processing status parameters

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

BCM

### SEE ALSO

*abs, absf [▶ 42]*, *sab [▶ 84]*, (.basl)

## 3.9 dt

### NAME

dt - Calculate the first derivative of the data (1D)

### DESCRIPTION

The command **dt** calculates the first derivative of the current dataset. Depending on the value of DATMOD, **dt** works on the raw or on the processed data.

### INPUT PARAMETERS

Set by the user with **edp** or by typing **datmod** :

DATMOD - data mode: work on raw or processed data

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - raw data (input if DATMOD = raw)

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (input if DATMOD = proc)

*proc* - processing parameters

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

*procs* - processing status parameters

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

DT

## 3.10 ef, efp

### NAME

ef - Exponential window multiplication + Fourier transform (1D)

efp - Exponential window multiplication + FT + phase correction (1D)

### DESCRIPTION

The composite processing command **ef** is a combination of **em** and **ft**, i.e. it performs an exponential window multiplication and a Fourier transform.

**efp** is a combination of **em**, **ft** and **pk**, i.e. it does the same as **ef** but, in addition, performs a phase correction.

**ef** and **efp** automatically perform an FID baseline correction according to BC_mod.

All composite processing commands can be found in the menu:

**Process | Advanced | Special Transforms**

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - raw data (input if *1r*, *1i* do not exist or are Fourier transformed)

*acqus* - acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed data (input if they exist but are not Fourier transformed)

*proc* - processing parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

*procs* - processing status parameters

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

EF

EFP

**SEE ALSO**

*gf, gfp* [▷ 65], *fp, fmc* [▷ 60]

## 3.11 em, gm, wm

**NAME**

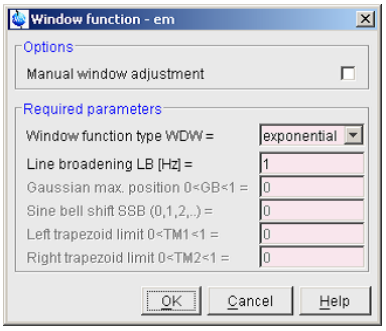
em - Exponential window multiplication of the FID (1D)

gm - Gaussian window multiplication of the FID (1D)

wm - Open window function dialog box (1D, 2D)

**DESCRIPTION**

Window multiplication commands can be entered on the command line or started from the window function dialog box. The latter is opened with the command **wm**.




The parameter section of this dialog box offers several window functions, each of which selects a certain command for execution.

**Exponential multiplication**

This function selects the command **em** for execution. It performs an exponential window multiplication of the FID. It is the most used window function for NMR spectra. **em** multiplies each data point *i* with the factor:

$$\exp\left(-\frac{(i-1)\cdot LB\cdot\pi}{2\cdot SWH}\right)$$

Where LB (the line broadening factor) is a processing parameter and SWH (the spectral width) an acquisition status parameter.

### Gaussian multiplication

This function selects the command **gm** for execution. It performs a Gaussian window multiplication of the FID. The result is a Gaussian line shape after Fourier transform. This line shape has sharper edges than the line shape caused by **em**. **gm** multiplies the FID with the function:

$$\exp(((-at))-(-bt^2))$$

Where *t* is the acquisition time in seconds and the parameters a and b are defined by:

$$a=\pi\cdot LB \quad \text{and} \quad b=\frac{\mathrm{a}}{2\,GB\cdot AQ}$$

In this equation, LB and GB are processing parameters which represent the exponential broadening factor and the Gaussian broadening factor, respectively. AQ is an acquisition status parameter which represents the acquisition time.

**gm** allows to separate overlapping peaks. The quality of the separation depends on the choice of the parameters LB and GB. Suitable values can be determined with *Manual window adjustment*. The value of LB must be negative, typically the half line width of the spectral peaks. Note that for exponential window multiplication (**em**), LB must be positive. The value of GB must lie between 0 and 1. It determines the position of the top of the Gaussian function. For example, for GB = 0.5 the top lies in the middle of the FID. Note that for large values of GB (close to 1), peaks can become negative at the edges which can impair quantitative analysis of the spectrum.

**em** and **gm** implicitly perform a baseline correction of the FID, according to the processing parameter BC_mod. Furthermore, they perform linear prediction according to the parameters ME_mod, NCOEF and LPBIN.

When executed on 2D or 3D data, **em** and **gm** take up to four arguments, e.g. **em <row> <procno> n y** process the specified row and store it under the specified *procno*. The last two arguments are optional: **n** prevents changing the display to the output 1D data, **y** causes a possibly existing data to be overwritten without warning.

When executed on a dataset with 2D or 3D raw data but 1D processed data (usually a result of rsr, rsc or a previous 1D processing command on that 2D or 3D data, **em** and **gm** take one argument, e.g. **em <row>** processes the specified row and stores it under the current *procno*.

**em same** processes the same row as the previous processing command and stores it under the current *procno*. The **same** option is automatically used by the AU program macros EM and GM. When used on a regular 1D dataset (i.e. with 1D raw data) it has no effect.

If you run a command like **em** from the command line, you have to make sure that the required parameters are already set. Click the Procpars tab or enter **edp** to do that.

The **wm** command can be used on 1D or 2D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

### INPUT PARAMETERS

Set from the **wm** dialog box, with **edp** or by typing **lb**, **bc_mod** etc.:

LB - Lorentzian broadening factor

GB - Gaussian broadening factor

BC_mod - FID baseline correction mode

Set by the acquisition, can be viewed with **dpa** or **s swh**:

SWH - spectral width

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - raw data (input if *1r*, *1i* do not exist or are Fourier transformed)

*acqus* - acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed data (input if they exist but are not Fourier transformed)

*proc* - processing parameters

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

*procs* - processing status parameters

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

EM

GM

### SEE ALSO

*sinm, qsin, sinc, qsinc* [▷ 78], *tm, traf, trafs* [▷ 87]

## 3.12 filt

### NAME

filt - Digital filtering of the data (1D)

### DESCRIPTION

The command **filt** smoothes the data by replacing each point with a weighted average of its surrounding points. By default, **filt** uses the weighting coefficients 1-2-1 which means that the intensity *p(i)* of data point *i* is replaced by:

$1 * p(i - 1) + 2 * p(i) + 1 * p(i + 1)$.

Different weighting algorithms can be set up by creating a new file in the directory:

*<tshome>/exp/stan/nmr/filt/1d*

Just copy the default file *threepoint* to a different name and modify it with a text editor. The file must look like:

3,1,2,1

or

5,1,2,3,2,1

Where the first number represents the number of points used for smoothing and must be odd. The other numbers are the weighting coefficients for the data points. The processing parameter DFILT determines which file is used by **filt**.

This is one of the few cases where file handling cannot be done from TopSpin and needs to be done on operating system level.

### INPUT PARAMETERS

Set by the user with **edp** or by typing **dfilt**, **datmod** etc. :

DFILT - digital filter filename

DATMOD - data mode: work on raw or processed data

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

*proc* - processing parameters

*<tshome>/exp/stan/nmr/filt/1d/**

digital filtering file(s)

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

*procs* - processing status parameters

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

FILT

## 3.13 fp, fmc

**NAME**

fp - Fourier transform +phase correction (1D)

fmc - Fourier transform + magnitude calculation (1D)

**DESCRIPTION**

The composite processing command **fp** is a combination of **ft** and **pk**, i.e. it performs a 1D Fourier transform and a phase correction.

**fmc** is a combination of **ft** and **mc**, i.e. it performs a 1D Fourier transform and a magnitude calculation.

**fp** and **fmc** automatically perform an FID baseline correction according to BC_mod.

All composite processing commands can be found in the menu:

**Process | Advanced | Special Transforms**

**INPUT AND OUTPUT PARAMETERS**

See the commands **ft**, **pk** and **mc**.

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - raw data (input if *1r*, *1i* do not exist or are Fourier transformed)

*acqus* - acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed data (input if they exist but are not Fourier transformed)

*proc* - processing parameters

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

*procs* - processing status parameters

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

FP

FMC

### SEE ALSO

*ef, efp* [▸ 56], *gf, gfp* [▸ 65]

## 3.14 ft, ftf

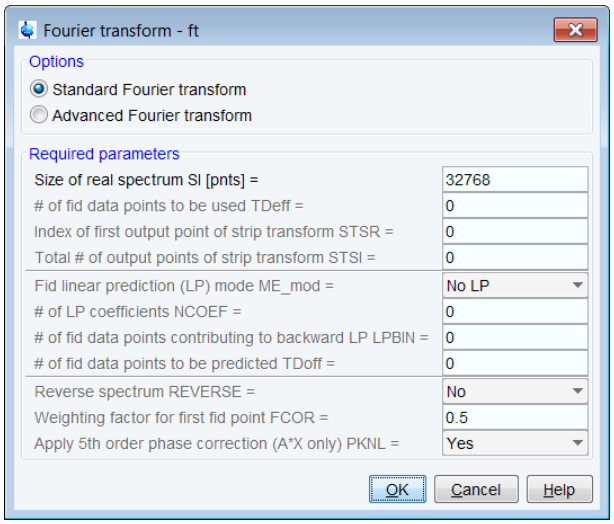
### NAME

ft - Fourier transform (1D)

ftf - Open the Fourier transform dialog box (1D, 2D)

### DESCRIPTION

The command **ft** Fourier transforms a 1D dataset or a row of a dataset with dimension ≥ 2. It can be started from the command line or from the Fourier transform dialog box. The latter is opened with the command **ftf**




This dialog box offers two options both of which select the **ft** command for execution.

#### Standard Fourier Transform

This option only allows to set the parameter SI, the size of the real spectrum.

#### Advanced Fourier Transform

This option allows to set all FT related parameters.

Fourier transform is the main step in processing NMR data. The time domain data (FID) which are created by acquisition are transformed into frequency domain data (spectrum). Usually, Fourier transform is preceded by other processing steps like FID baseline correction (**bc**) and window multiplication (**em**, **gm**, etc.) and followed by steps like phase correction ( **apk**) and spectrum baseline correction (**abs**).

The size of the resulting spectrum is determined by the parameter SI. An FID of TD time domain points is transformed to a spectrum of SI real and SI imaginary data points. A typical value for SI is TD/2. In that case, all points of the FID are used by the Fourier transform and no zero filling is done.

The size of the spectrum and the number of FID points which are used can be determined in the following ways:

- SI > TD/2: the FID is zero filled
- SI < TD/2: only the first 2*SI points of the FID are used
- 0 < TDeff < TD: only the first TDeff points of the FID are used

In the latter two cases, the spectrum will contain less information then the FID. Note that the parameter TDoff only plays a role for linear prediction and in 2D and 3D Fourier transform.

You can also perform a so-called strip transform which means that only a certain region of the spectrum is stored. This can be done by setting the parameters STSR and STSI which represent the strip start and strip size, respectively. They can take values between 0 and SI. The processing status parameters STSI and SI are both set to this value. You can check this by entering **dpp** or clicking the Procpars tab.

The Fourier transform mode depends on the acquisition mode; *single*, *sequential* or *simultaneous*. For this purpose, **ft** evaluates the acquisition status parameter AQ_mod as shown in the table below:

| AQ_mod | FT_mod | Fourier transform mode |
|---|---|---|
| qf | fsr | forward, single channel, real |
| qsim | fqc | forward, quadrature, complex |
| qseq | fqr | forward, quadrature, real |
| DQD | fqc | forward, quadrature, complex |

Note that **ft** does not evaluate the processing parameter FT_mod but it does store the Fourier transform mode, as evaluated from the acquisition mode, in the processing status parameter FT_mod. However, the command **trf** determines the Fourier transform mode from the processing parameter FT_mod and not from the acquisition mode (see **trf**).

**ft** evaluates the parameter FCOR. The first point of the FID is multiplied with FCOR which is a value between 0.0 and 2.0. However, on Avance spectrometers, the FID of digitally filtered data starts with a group delay of which the first points are zero so that the value of FCOR is irrelevant. On A*X data, FCOR allows to control the DC offset of the spectrum.

**ft** evaluates the parameter PKNL. On A*X spectrometers, PKNL = true causes a non linear 5th order phase correction of the raw data. This corrects possible errors caused by non linear behaviour of the analog filters. On Avance spectrometers, PKNL must always be set to TRUE. For digitally filtered data, it causes **ft** to handle the group delay of the FID. For analog data it has no effect.

**ft** evaluates the parameter REVERSE. If REVERSE = TRUE, the spectrum will be reversed, i.e. the first output data point becomes the last and the last point becomes the first. The same effect is attained by using the command **rv** after **ft**.

**ft** automatically performs an FID baseline correction according to BC_mod.

**ft** performs linear prediction according to ME_mod. This parameter can take the following values:

*no* : no linear prediction

*LPfr* : forward LP on real data

*LPfc* : forward LP on complex data

*LPbr* : backward LP on real data

*LPbc* : backward LP on complex data

*LPmifr* : mirror image forward LP on real data

*LPmifc* : mirror image forward LP on complex data

Forward prediction can, for example, be used to extend truncated FIDs. Backward prediction can be used to improve the initial data points of the FID. **ft** determines the detection mode (real or complex) from the acquisition status parameter AQ_mod, not from ME_mod. As such, **ft** does not distinguish between ME_mod = LPfr and ME_mod = LPfc. The same counts for backward prediction. Note that the command **trf** does determine the detection mode from ME_mod. Linear prediction is only performed for NCOEF > 0. Furthermore, LPBIN and, for backward prediction, TDoff play a role (see these parameters in chapter *List of processing parameters [▷ 20]*). By default, ME_mod is set to *no* which means no linear prediction is done.

When executed on a 2D or 3D dataset, **ft** takes up to four arguments, e.g. **ft <row> <procno> y n**, process the specified *row* and store it under the specified *procno*. The last two arguments are optional: **y** causes a possibly existing data to be overwritten without warning, **n** prevents TopSpin from changing to the destination dataset. Note that the oder of the last two arguments, **y** and **n**, is irrelevant.

If you run a command like **ft** from the command line, make sure that the required parameters are already set. Click the Procpars tab or enter **edp** to do that.

The **ft** command can be used on multidimensional data. In that case it automatically recognizes the dimensionality of the data and prompts for the row to be processed and the output *procno*. It only applies to the acquisition direction.

The **ftf** command can be used on 1D and 2D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

**INPUT PARAMETERS**

Set from the **ftf** dialog box, with **edp** or by typing **si**, **stsr** etc.:

SI - size of the processed data

STSR - strip start: first output point of strip transform

STSI - strip size: number of output points of strip transform

TDeff - number of raw data points to be used for processing

FCOR - first (FID) data point multiplication factor (0.0-2.0, default 0.5)

REVERSE - flag indicating to reverse the spectrum

PKNL - group delay compensation (Avance) or filter correction (A*X)

ME_mod - FID linear prediction mode

NCOEF - number of linear prediction coefficients

LPBIN - number of points for linear prediction

TDoff - number of raw data points predicted for ME_mod = LPb*

Set by the acquisition, can be viewed with **dpa** or by typing **s aq_mod** etc.:

AQ_mod - acquisition mode (determines the Fourier transform mode)

TD - time domain; number of raw data points

BYTORDA - byteorder or the raw data

NC - normalization constant

### OUTPUT PARAMETERS

Can be viewed with **dpp** or by typing **s ft_mod**, **s tdeff** etc.:

FT_mod - Fourier transform mode

TDeff - number of raw data points that were used for processing

STSR - strip start: first output point of strip transform

STSI - strip size: number of output points of strip transform

NC_proc - intensity scaling factor

YMAX_p - maximum intensity of the processed data

YMIN_p - minimum intensity of the processed data

BYTORDP - data storage order

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - raw data (input if *1r*, *1i* do not exist or are Fourier transformed)

*acqus* - acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed data (input if they exist but are not Fourier transformed)

*proc* - processing parameters

### OUTPUT FILES

*1r*, *1i* - processed 1D data (real, imaginary)

*procs* - processing status parameters

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

FT

### SEE ALSO

*ift* [ 67], *ht* [ 66], *trf, trfp* [ 89]

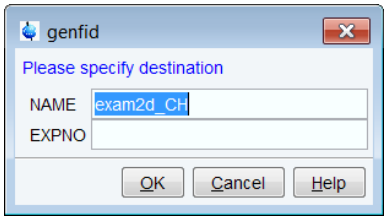
## 3.15 genfid

### NAME

genfid - Generate pseudo-raw data (1D)

### DESCRIPTION

The command **genfid** generates pseudo-raw data from processed data. When entered without arguments, it opens a dialog box where you can specify the destination dataset.



**genfid** is normally used in combination with the command **ift** which performs an inverse Fourier transform, converting a spectrum into an FID. Actually, **ift** transforms processed frequency domain data into processed time domain data. **genfid** converts these processed time domain data into pseudo-raw time domain data and stores them under a new name or experiment number (*expno*).

Note that **genfid** does not modify the data, but only stores them in a different format. The number of data points of the pseudo-raw data, is twice the size (SI) of the processed data they are created from. The acquisition status parameter TD (types **s td** or **dpa**) is set accordingly; TD = 2*SI.

**genfid** takes arguments and can be used as follows:

1. **genfid <expno>**
2. The FID will be stored under the specified *expno*.
3. **genfid <expno> <name> y**
4. The FID will be stored under the specified *name* and *expno*. The last argument (*y*) causes **genfid** to overwrite possibly existing data.

You can use any other combination of arguments as long they are entered in the correct order. The processed data number (*procno*) of the output dataset is always set to 1.

**genfid** can be used if you want to reprocess a 1D spectrum, for example with different processing parameters, but the raw data do not exist any more. An example of such a procedure is:

**ift** (if the data are Fourier transformed)

**genfid** (to create the pseudo-raw data)

**edp** (to set the processing parameters)

**ef** (to process the pseudo-raw data)

If the input data are processed but not Fourier transformed, you can skip the first step.

#### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed time domain data (real, imaginary)

#### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - pseudo-raw data

*audita.txt* - acquisition audit trail

#### USAGE IN AU PROGRAMS

GENFID(expno)

Overwrites possibly existing raw data in the specified *expno*

#### SEE ALSO

*ift* [▶ 67], *genser* [▶ 108]

## 3.16 gf, gfp

#### NAME

gf - Gaussian window multiplication + Fourier transform (1D)

gfp - Gaussian window multiplication + FT + phase correction (1D)

### DESCRIPTION

The composite processing command **gf** is a combination of **gm** and **ft**, i.e. it performs a Gaussian window multiplication and a Fourier transform.

**gfp** is a combination of **gm**, **ft** and **pk**, i.e. it does the same as **gf** but, in addition, performs a phase correction.

**gf** and **gfp** automatically perform an FID baseline correction according to BC_mod.

All composite processing commands can be found under the menu:

**Process | Advanced | Special Transforms**

### INPUT AND OUTPUT PARAMETERS

See **gm**, **ft** and **pk**

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - raw data (input if *1r*, *1i* do not exist or are Fourier transformed)

*acqus* - acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed data (input if they exist but are not Fourier transformed)

*proc* - processing parameters

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

*procs* - processing status parameters

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

GF

GFP

### SEE ALSO

*ef, efp* [▸ 56], *fp, fmc* [▸ 60]

## 3.17 ht

### NAME

ht - Hilbert transform (1D)

### DESCRIPTION

The command **ht** performs a Hilbert transform which means the imaginary part of a spectrum is calculated from the real part. This is only useful when the real data have been created from zero filled raw data, with SI ≥ TD. Only then they will contain the entire spectral information.

Imaginary data are required for phase correction. They are normally created together with the real data by Fourier transform. Directly after the Fourier transform, real and imaginary data are consistent and can be used for phase correction. If, however, the real data are manipulated, e.g. by **abs**, they are no longer consistent with the imaginary data. In that case, or when the imaginary data have been deleted, **ht** can be used to create new imaginary data.

Hilbert transform is based on the so called dispersion relations or Kramers-Kronig relations (see, for example, R. R. Ernst, G. Bodenhausen and A. Wokaun, Principles of nuclear magnetic resonance in one and two dimensions, Clarendon Press, Oxford, 1987).

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*1r* - real processed 1D data

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1i* - imaginary processed data

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

HT

**SEE ALSO**

*ift* [▷ 67], *ft, ftf* [▷ 61], *trf, trfp* [▷ 89]

## 3.18 ift

**NAME**

ift - Inverse Fourier transform (1D)

**DESCRIPTION**

The command **ift** performs an inverse Fourier transform of a 1D spectrum, thus creating an artificial FID. Normally, **ift** is done when the raw data do not exist any more. If, however, raw data do exist, they are not overwritten. **ift** stores the resulting FID as processed data, i.e. it overwrites the current spectrum.

After **ift**, you can create pseudo-raw data with the command **genfid** which creates a new dataset. Note that the number of data points of the pseudo-raw data, is twice the size of the processed data they are created from. The acquisition status parameter TD (**dpa**) is set accordingly; TD = 2*SI.

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (frequency domain)

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (time domain)

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

IFT

**SEE ALSO**

*genfid* [▷ 64], *ft, ftf* [▷ 61], *trf, trfp* [▷ 89]

## 3.19 ls, rs

### NAME

ls - Left shift data NSP points (1D)

rs - Right shift data NSP points (1D)

### DESCRIPTION

The command **ls** shifts 1D data to the left. The number of points shifted is determined by the parameter NSP. The right end of the data is filled with NSP zeroes.

**rs** shifts 1D data to the right. The number of points shifted is determined by the parameter NSP. The left end of the data is filled with NSP zeroes.

Depending on the parameter DATMOD, **rs** and **ls** work on raw or processed data.

The value of NSP is the number of the real plus imaginary data points that are shifted. As such, the real data are shifted NSP/2 points and the imaginary data are shifted NSP/2 points. For odd values of NSP the real and imaginary data points are interchanged. As such the displayed spectrum is not only shifted but also changes from real (absorption) to imaginary (dispersion) or vice versa. Note that his only plays a role for DATMOD = proc.

### INPUT PARAMETERS

Set by the user with **edp** or by typing **nsp**, **datmod** etc.:

NSP - number of points to be shifted

DATMOD - data mode: work on raw or processed data

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - raw data (input if DATMOD = raw)

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (input if DATMOD = proc*)*

*proc* - processing parameters

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

*procs* - processing status parameters

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

LS

RS

### SEE ALSO

*pk* [▶ 71]

## 3.20 mc

### NAME

mc - Magnitude calculation (1D)

### DESCRIPTION

The command **mc** calculates the magnitude spectrum of a 1D dataset. The intensity of each point *i* is replaced by its absolute value according to the formula:

$$ABS(i) = \sqrt{(R(i)^2 + I(i)^2)}$$

Where R and I are the real and imaginary part of the spectrum, respectively. If no processed input data exist, **mc** works on the raw data.

**mc** can also be started from the phase correction dialog box which is opened with **ph**.

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - raw 1D data (input if *1r*, *1i* do not exist)

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (input if they exist)

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

MC

### SEE ALSO

*ps* [▶ 74], *pk* [▶ 71], *apk, apks, apkm, apkf, ph* [▶ 51], *trf, trfp* [▶ 89]

## 3.21 mul, mulc, nm, div

### NAME

mul - Multiply two datasets (1D)

mulc - Multiply data with a constant (1D)

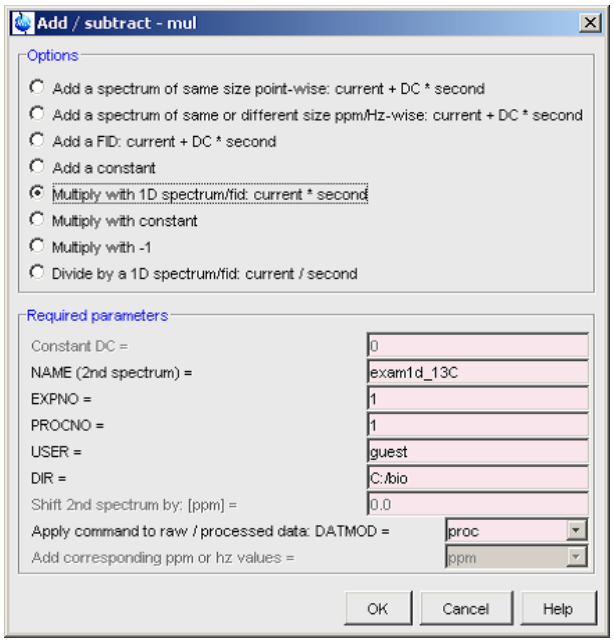
nm - Negate data (1D)

div - Divide two datasets (1D)

adsu - Open add/subtract/multiply dialog box (1D, 2D)

### DESCRIPTION

Multiplication commands can be entered on the command line or started from the add/subtract/multiply dialog box. The latter is opened with **adsu**.




This dialog box offers several options, each of which selects a certain command for execution.

### Multiply with 1D spectrum/fid

This option selects the command **mul** for execution. It multiplies the second dataset with the third dataset. The result is stored in the current dataset.

### Multiply with constant

This option selects the command **mulc** for execution. It multiplies the current data with the value of DC.

### Multiply with -1

This option selects the command **nm** for execution. It negates the current data which means all data points are multiplied by -1.

### Divide by 1D spectrum/fid

This option selects the command **div** for execution. It divides the second dataset by the third dataset. The result is stored in the current dataset.

**mul**/**div** perform a complex multiplication/division on complex spectra. This requires that for both the second and third dataset:

- the status parameter FT_mod = fqc or fsc
- real (file *1r*) and imaginary (file *1i*) data exist

This is the case for most data that have been acquired in Avance spectrometers. If the above requirements are not fulfilled, real and imaginary data are multiplied/divided pointwise. When a complex operation has been performed, this is reported in the audit trail output file.

Please note in addition that deleting the imaginary data enforces a pointwise multiplication for the command **mul** instead of a complex multiplication.

**mul**, **div**, **mulc** and **nm** work on raw or on processed data, depending on the value of DATMOD. The result is always stored as processed data in the current dataset. The raw data are not overwritten.

When **mul** and **div** are started from the command line, they will run without user interaction if the second dataset is already defined (file *curdat2*). If this is not defined, the **adsu** dialog box will be opened. When you run a multiplication or division command from the command line, make sure that the required parameters are set. Click the Procpars tab or enter **edp** to do that.

The **adsu** command can be used on 1D or 2D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

**INPUT PARAMETERS**

Set from the **adsu** dialog box, with **edp** or by typing **dc**, **datmod** etc.:

DC - multiplication factor (input of **mulc**)

DATMOD - data mode: work on raw or processed data

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - raw data (input if DATMOD = raw)

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (input if DATMOD = proc)

*proc* - processing parameters

curdat2 - definition of the second dataset

*<dir2>/data/<user2>/nmr/<name2>/<expno2>/*

*fid* - second raw data (input if DATMOD = raw)

*<dir2>/data/<user2>/nmr/<name2>/<expno2>/pdata/<procno2>/*

*1r*, *1i* - processed 1D data (input if DATMOD = proc*)*

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

*procs* - processing status parameters

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

MUL

MULC

NM

DIV

**SEE ALSO**

*add [▶ 44]*

## 3.22 pk

**NAME**

pk - Phase correction according to PHC0/PHC1 (1D)

## DESCRIPTION

The command **pk** performs a zero and first order phase correction according to user defined phase values. These phase values are read from the processing parameters PHC0 and PHC1.

The data, consisting of real points R(i) and imaginary points I(i) are phase corrected according to the formula:

$R0(i) = R(i)\cos a(i) - I(i)\sin a(i)$

$I0(i) = I(i)\cos a(i) + R(i)\sin a(i)$

Where:

$a(i) = PHC0 + (i-1)PHC1$

Where i > 0, R0 and I0 represent the corrected values and PHC0 and PHC1 are processing parameters.

**pk** does not calculate the phase values but uses the preset values. Therefore, **pk** is only useful when these values are known. They can be determined, interactively, in Phase correction mode or, automatically, with **apk** or **apks**.

**pk** is typically used in a series of experiments where the first spectrum is corrected with **apk** and each successive spectrum with **pk**, using the same values (see for example AU program **proc_noe**).

**pk** applies but does not change the processing parameters PHC0 and PHC1 (**edp**). It does, however, change the corresponding processing status parameters PHC0 and PHC1 (**dpp**), by adding the applied phase values.

**pk** is a part of the composite processing commands **efp**, **fp** and **gfp**.

**pk** can also be used to perform a phase correction on an FID rather than a spectrum. This is automatically done if you enter **pk** on a dataset which does not contain processed data. Phase correction on an FID is used prior to Fourier transform to induce a shift in the resulting spectrum. The spectrum is shifted according to the value of PHC1; one real data point to the left for each 360°. A negative value of PHC1 causes a right shift. The points which are cut off on one side of the spectrum are appended on the other side. Note the difference with performing a left shift (**ls**) or right shift (**rs**) after Fourier transform. This appends zeroes at the opposite side. If processed data do exist and you still want to do a phase correction on the FID, you can do this with the command **trf**.

The command **pk** can also be started from the phase correction dialog box which is opened with **ph**.

## INPUT PARAMETERS

Set from the **ph** dialog box, with **edp** or by typing **phc0**, **phc1** etc.:

PHC0 - zero order phase correction value (frequency independent)

PHC1 - first order phase correction value (frequency dependent)

## OUTPUT PARAMETERS

Can be viewed with **dpp** or by typing **s phc0**, **s phc1** etc.:

PHC0 - zero order phase correction value (frequency independent)

PHC1 - first order phase correction value (frequency dependent)

## INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - raw data (input if no processed data exist)

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (input if they exist)

*proc* - processing parameters

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

*procs* - processing status parameters

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

PK

### SEE ALSO

*mc* [▷ 68], *ps* [▷ 74], *apk, apks, apkm, apkf, ph* [▷ 51], *trf, trfp* [▷ 89]

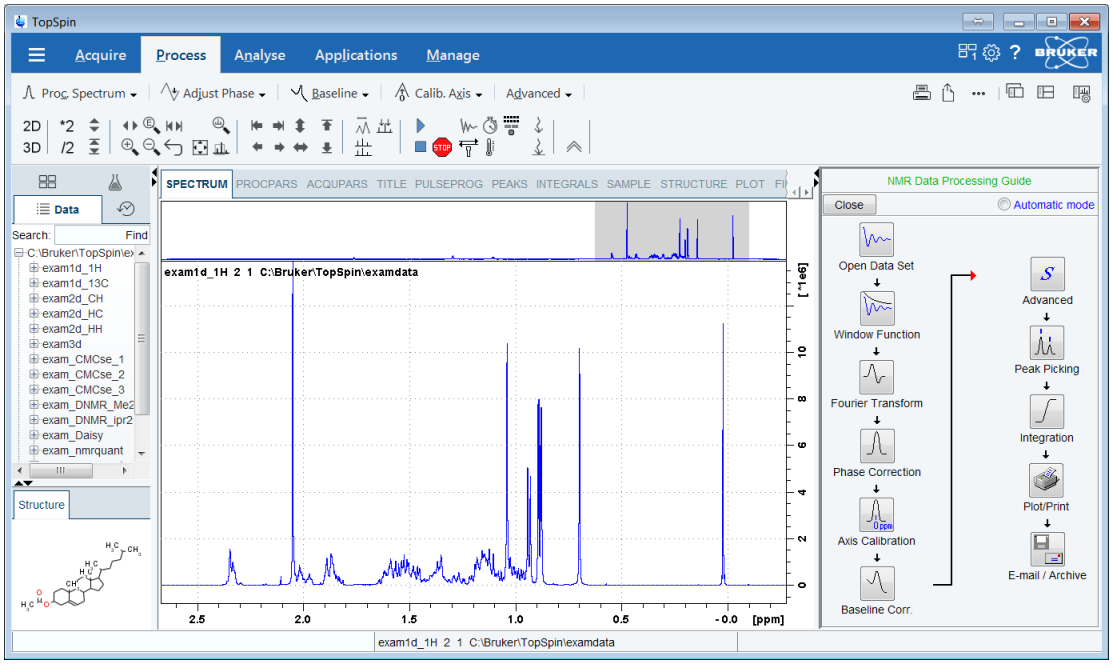
## 3.23 prguide

### NAME

prguide - Open the Processing Guide (1D,2D)

### DESCRIPTION

The command **prguide** opens the TopSpin Processing Guide:




This contains a workflow for processing data, especially suited for new or occasional users. In *Automatic mode*, the Processing Guide will simply execute a processing command when you click the corresponding button. This requires the processing parameters to be set correctly. In interactive mode (*Automatic mode* unchecked), the Processing Guide will, at each step, open a dialog box offering you the available options and required parameters. For example, the phase correction button offers various automatic algorithms as well as an option to switch to interactive phasing mode.

Experienced users normally enter the individual processing commands from the command line. This requires that, for each command, the processing parameters are set correctly.

The Processing Guide can be used for 1D and 2D processing.

### SEE ALSO

*managuide* [▶ 224], *solaguide* [▶ 242], *t1guide* [▶ 243], (aqguide)

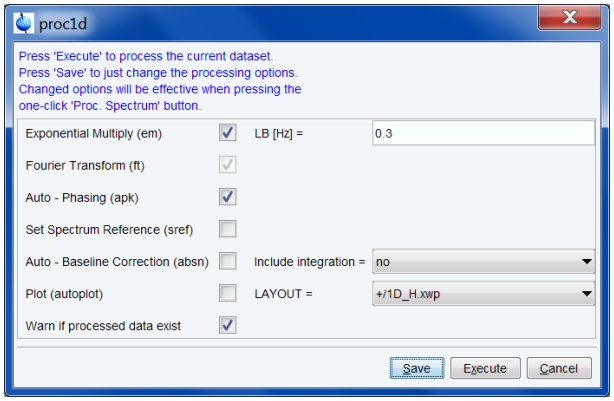
## 3.24 proc1d

### NAME

proc1d - Open 1D Processing dialog

### DESCRIPTION

The command **proc1d** opens a 1D processing dialog:




This dialog can be used for standard 1D processing, including exponential multiplication, Fourier transform, phase correction, referencing, baseline correction and plotting. Processing steps can be switched on or off and two parameters, line broadening and plot layout, can be set.

The command takes one argument:

**proc1d y**

Which will process the current dataset without opening the dialog, using the last settings.

### SEE ALSO

*prguide* [▶ 73]

## 3.25 ps

### NAME

ps - Calculate power spectrum (1D)

### DESCRIPTION

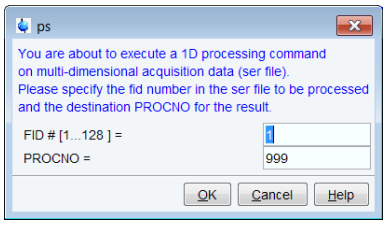
The command **ps** calculates the power spectrum of the 1D current dataset, replacing the intensity of each data point *i* according to the formula:

$PS(i) = R(i)^2 + I(I)^2$

Where R and I are the real and imaginary part of the spectrum, respectively. If no processed input data exist, **ps** works on the raw data. The result is always stored as the real processed data.

**ps** can also be started from the phase correction dialog box which is opened with **ph**.

If **ps** is typed on a 2D or 3D spectrum, the following warning message is displayed. Enter the appropriate values.




### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - raw data (input if no processed data exist)

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

PS

### SEE ALSO

*mc* [▸ 68], *pk* [▸ 71], *apk, apks, apkm, apkf, ph* [▸ 51], *trf, trfp* [▸ 89]

## 3.26 sigreg

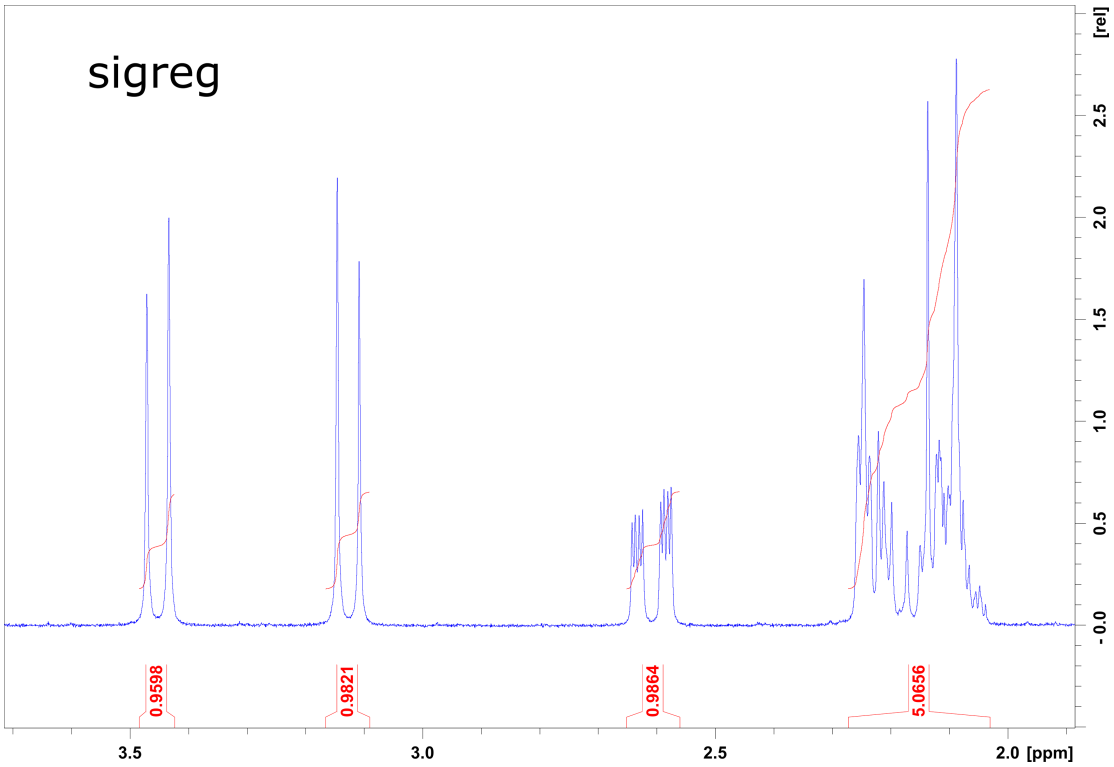
### NAME

**sigreg - Automatic signal region detection in 1D $^{1}$H spectra**

### DESCRIPTION

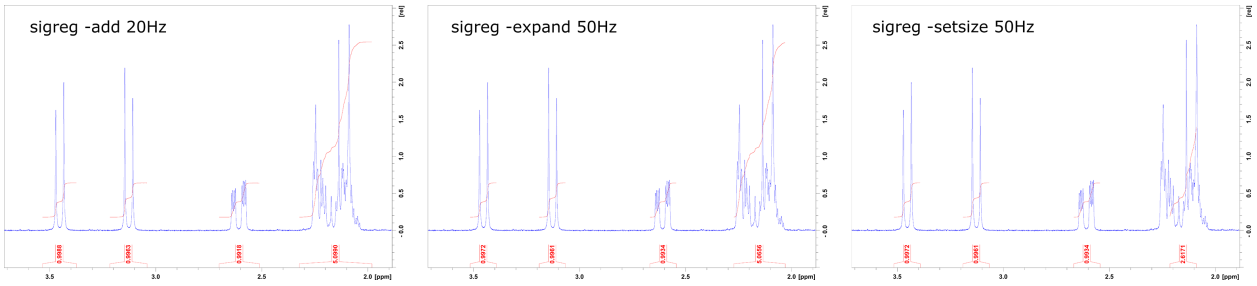
The processing command **sigreg** finds signal regions in proton spectra using a machine learning approach. The algorithm was developed and tested for 1D $^{1}$H spectra without solvent suppression.

To ensure optimal performance the spectrum should be phase- and baseline corrected before running **sigreg**.

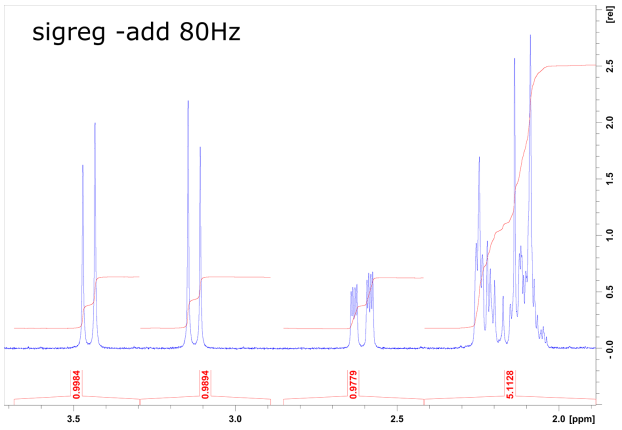


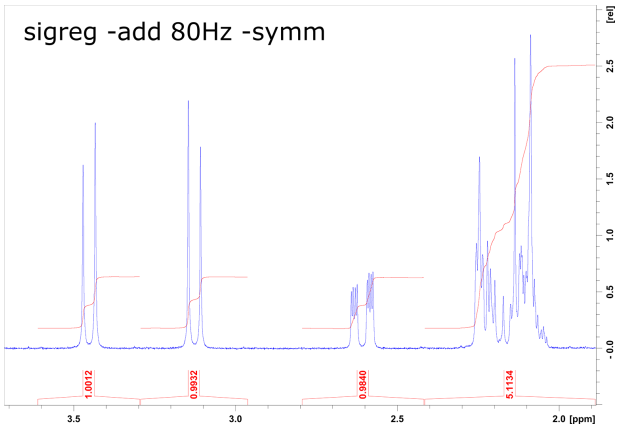

In case that the width of the signal regions detected by **sigreg** is not satisfactory, the command can be customized using the **-add**, **-expand**, and **-setsize** options.

- The **-add** option extends signal regions on both sides by a given amount of Hz or ppm.
- The **-expand** option extends all signal regions narrower than a given amount of Hz or ppm to that value of Hz or ppm. All signal regions broader than the given value remain untouched.
- The **-setsize** option sets the size of all signal regions to a given amount of Hz or ppm.

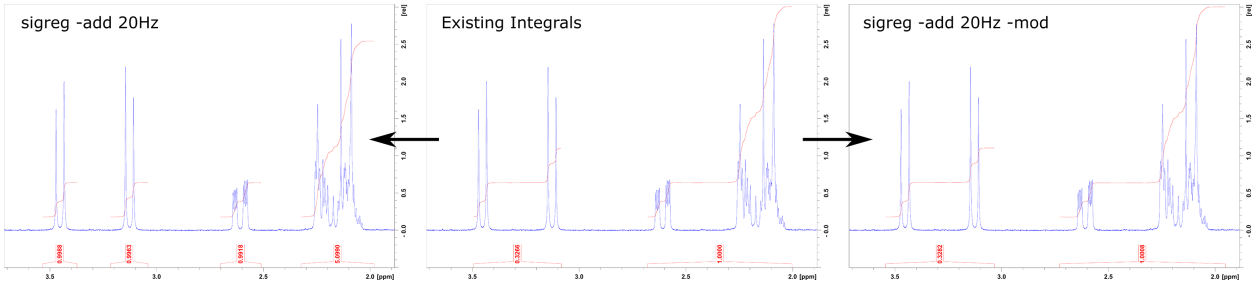



If the requested extension of the signal regions would cause adjacent regions to overlap, they are not merged, but kept separate, with the center of the overlap defining the limit of the two regions. As a result, the left and the right side of these two regions are extended by different values causing their centers to be shifted. This behavior can be prevented by using the **-symm** option that forces the changes to be symmetric with respect to the center of the signal region, and thus ensuring that the centers of all signal regions remain unchanged (example: **sigreg -setsize 0.1ppm -symm**).




By default, the signal regions are detected using **sigreg** before modification. By using the **-mod** option, it is possible to suppress this initial detection of signal regions, in order to apply the requested modifications to previously defined signal regions instead (example: **sigreg -add=0.15ppm -symm -mod**).




## USAGE

**sigreg**: detects signal regions in 1D 1H spectra.

**sigreg -add <value>Hz|ppm**: detects signal regions in 1D 1H spectra and extends them on both sides by <value> Hz|ppm.

**sigreg -expand <value>Hz|ppm**: detects signal regions in 1D 1H spectra and extends all signal regions narrower than <value>Hz|ppm to <value> Hz|ppm.

**sigreg -setsize <value>Hz|ppm**: detects signal regions in 1D 1H spectra and sets their size to <value> Hz|ppm.

**sigreg -add|-expand|-setsize <value>Hz|ppm -symm**: detects signal regions in 1D 1H spectra, performs the **-add|-expand|-setsize** option, and forces the changes to be symmetric with respect to the center of the signal regions.

**sigreg -add|-expand|-setsize <value>Hz|ppm -mod**: performs the **-add|-expand|-setsize** option on existing signal regions.

**sigreg -add|-expand|-setsize <value>Hz|ppm -symm -mod**: performs the **-add|-expand|-setsize** option on existing signal regions, and forces the changes to be symmetric with respect to the center of the signal region.

**Note:**

- **sigreg -add** accepts both positive and negative values, while **sigreg -expand** and **sigreg -setsize** accept only positive values.
- When using these options, all required information must be specified by command line arguments. The **-add, -expand**, and **-setsize** options must be followed by (1) the value for the modification of the signal regions and (2) the unit (Hz or ppm). Between the

numerical value and the unit there must not be any space, while the numerical value can be separated by the option using a space (example: **sigreg -add 10Hz**) or an equal sign (example: **sigreg -expand=1ppm**).

### INPUT FILES

*<dir>/<name>/<expno>/pdata/<procno>/*

*1r, 1i* - processed 1D data (frequency domain)

### OUTPUT FILES

*<dir >/<name>/<expno>/pdata/<procno>/*

*intrng* - integral regions

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

SIGREG

### SEE ALSO

*abs* [▶ 42], *int* [▶ 218], *apbk* [▶ 48]

## 3.27 sinm, qsin, sinc, qsinc

### NAME

sinm - Sine window multiplication of the FID (1D)

qsin - Sine squared window multiplication of the FID (1D)

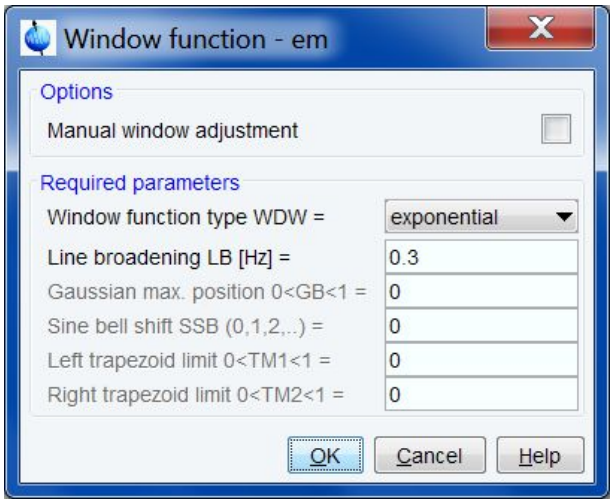
sinc - Sinc window multiplication of the FID (1D)

qsinc - Sinc squared window multiplication of the FID (1D)

wm - Open window multiplication dialog box (1D,2D)

### DESCRIPTION

Window multiplication commands can be started from the command line or from the window function dialog box. The latter is opened with the command **wm**:



This dialog box offers several window functions, each of which selects a certain command for execution.

### Sine bell

This window function selects the command **sinm** for execution. It performs a sine window multiplication, according to the function:

$SINM(t) = \sin((\pi - PHI) * (t / AQ) + PHI)$

where

$0 < t < AQ$ and $PHI = \pi / SSB$

Where AQ is an acquisition status parameter and SSB a processing parameter.

Typical values are SSB = 1 for a pure sine function and SSB = 2 for a pure cosine function. Values greater than 2 give a mixed sine/cosine function. Note that all values smaller than 2, for example 0, have the same effect as SSB = ¥, namely a pure sine function.

### Squared sine bell

This window function selects the command **qsin** for execution. It performs a sine squared window multiplication, according to the function:

$QSIN(t) = \sin((\pi - PHI) \times (t / AQ) + PHI)^2$

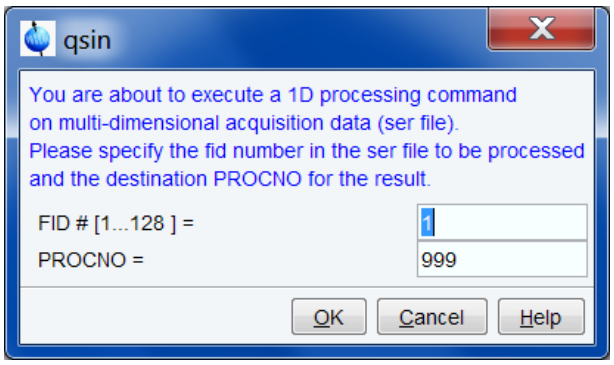
where

$0 < t <$ AQ and $PHI = \pi / SSB$

Where AQ is an acquisition status parameter and SSB a processing parameter.

Typical values are SSB = 1 for a pure sine function and SSB = 2 for a pure cosine function. Values greater than 2 give mixed sine/cosine functions. Note that all values smaller than 2 have the same effect as SSB = 1, namely a pure sine function.

If commands like **qsin** are typed on a 2D or 3D spectrum, the following warning message is displayed. Enter the appropriate values.




### Sinc

This window function selects the command **sinc** for execution. It performs a sinc window multiplication, according to the function:

$$SINC(t) = \frac{\sin t}{t}$$

Where

$-2\pi * SSB * GB < t < 2\pi * SSB * (1 - GB)$

and SSB and GB are processing parameters.

### Squared sinc

This window function selects the command **qsinc** for execution. It performs a sinc squared window multiplication, according to the function:

$$QSINC(t) = \left(\frac{\sin t}{t}\right)^2$$

Where

$-2\pi * SSB * GB < t < 2\pi * SSB * (1 - GB)$

and SSB and GB are processing parameters.

The ***sin*** commands implicitly perform a baseline correction of the FID, according to the processing parameter BC_mod. Furthermore, they perform linear prediction according to the parameters ME_mod, NCOEF and LPBIN.

If you run a command like **sinm** from the command line, you have to make sure that the required parameters are already set. Click the Procpars tab or enter **edp** to do that.

When executed on 2D or 3D data, the ***sin*** commands take up to four arguments, e.g. **sinm <row> <procno> n y**, process the specified row and store it under the specified *procno*. The last two arguments are optional: **n** prevents changing the display to the output 1D data, **y** causes a possibly existing data to be overwritten without warning.

When executed on a dataset with 2D or 3D raw data but 1D processed data (usually a result of rsr, rsc or a previous 1D processing command on that 2D or 3D data), the **sin*** commands take one argument **sinm <row>** to process the specified row and store it under the current *procno*.

**sinm same** process the same row as the previous processing command and store it under the current *procno*. The **same** option is automatically used by the AU program macros *SIN*. When used on a regular 1D dataset (i.e. with 1D raw data) it has no effect.

The **wm** command can be used on 1D or 2D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

### INPUT PARAMETERS

Set from the **wm** dialog box, with **edp** or by typing **ssb**, **gb** etc.:

SSB - sine bell shift

GB - Gaussian broadening factor (input of **sinc** and **qsinc**)

Set by the acquisition, can be viewed with **dpa** or **s aq**:

AQ - Acquisition time (input of **sinm** and **qsin**)

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - raw data (input if *1r*, *1i* do not exist or are Fourier transformed)

*acqus* - acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed data (input if they exist but are not Fourier transformed)

*proc* - processing parameters

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

*procs* - processing status parameters

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

SINM

QSIN

SINC

QSINC

### SEE ALSO

*em, gm, wm [▶ 57], tm, traf, trafs [▶ 87]*

## 3.28 refdcon

### NAME

refdcon – Reference deconvolution (1D)

.refdcon – Interactive reference deconvolution (1D)

### DESCRIPTION

Reference deconvolution is a simple and effective method to remove distortions caused by field inhomogeneities or modulations in nuclear magnetic resonance spectroscopy.

### USAGE

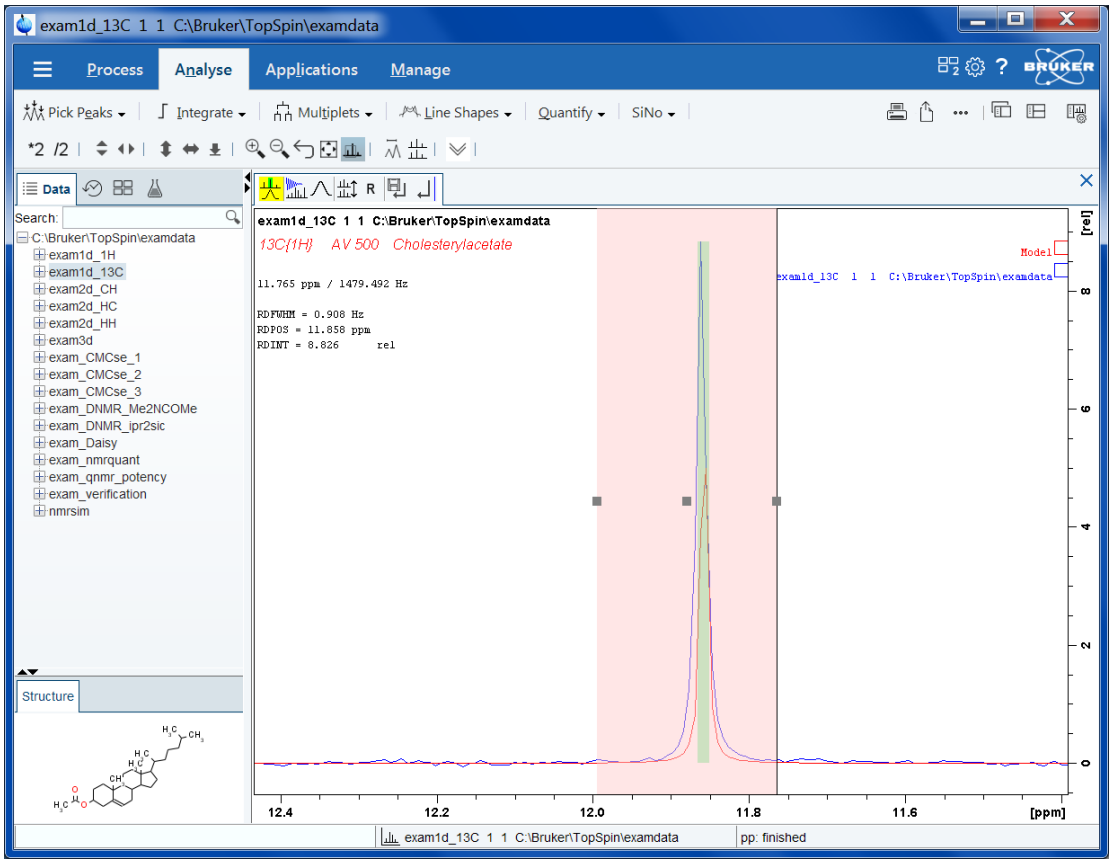
To start the reference deconvolution module call the menu item **Process | Advanced | Reference Deconvolution** or run the command **.refdcon**.

The reference deconvolution can also be used by the command **refdcon** on the command line. In this case all defined input parameters will be used to calculate the deconvolved spectra automatically. In a minimal usage the interval has to be defined. Additional parameters are optional. In case no parameter is set before, it is possible to set all parameters as arguments, shown in the following example:

**refdcon rdf1=”-1.3” rdf2=”2” rdfwhm=”2.2”**

#### Define and adapt the Lorentzian model

Starting the interactive reference deconvolution module it is in a first step necessary to define the region [RDF1, RDF2] including the peak of interest for the Lorentz model calculation. The interval can be defined with a left mouse click and dragging till the end of the desired region. As a consequence a Lorentzian model for the maximum peak in the interval will be calculated and shown on the screen (see the following figure):



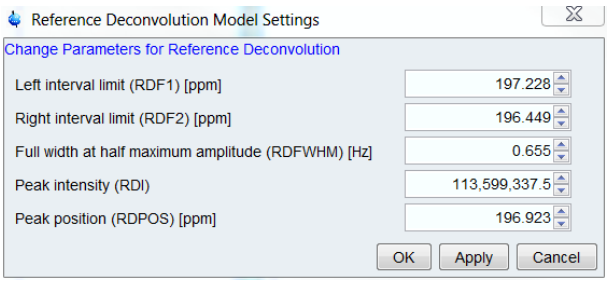

The calculated Lorentzian model and the selected region are marked with green and pink boxes.

Parameters can be changed using the associated adjustment handles and the model will update immediately.

The left and right handle of the model (green box) changes the half maximum amplitude parameter (RDFWHM) and the handle above adapt the intensity. The handle in the center changes the peak position.

While using the adjustment handles of the selected region (rosa box) the region size changes for the following deconvolution. The Lorentz will stay the same as long as the selected peak is in the region. By default the maximum peak in the interval is the selected peak.

Another possibility to change the model parameters is by using the parameter dialog. A double click in the selected region opens the following dialog:



**The menu bar**

-  - Start reference deconvolution of the spectra with the generated Lorentzian model
-  - Recalculate the default Lorentzian model in the current region.
-  - Change to stacked layout for a better comparison of the results (Lorentzian model or deconvolved spectra)
-  - Reset individual scaling
-  - Save deconvolved data

**INPUT PARAMETERS**

RDF1: Left interval limit for reference deconvolution [ppm]

RDF2: Right interval limit for reference deconvolution [ppm]

RDINT: Intensity for Lorentzian peak

RDPOS Position for Lorentzian peak [ppm]

RDFWHM: Full width at half maximum amplitude for Lorentzian peak [Hz]

**SEE ALSO**

REFDCON Manual

## 3.29 rv

**NAME**

rv - Reverse spectrum or FID (1D)

**DESCRIPTION**

The command **rv** reverses the data with respect to the middle data point, i.e. the leftmost data point becomes the rightmost point and vice versa. The real and imaginary parts of the spectrum are thereby interchanged. Depending on the value of DATMOD, **rv** works on the raw or on the processed data. The result is always stored as processed data.

A spectrum can also be reversed as a part of the Fourier transform by setting the processing parameter REVERSE to TRUE.

**INPUT PARAMETERS**

Set by the user with **edp** or by typing **datmod** :

DATMOD - data mode: work on raw or processed data

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - raw data (input if DATMOD = raw)

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (input if DATMOD = proc)

*proc* - processing parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data

*procs* - processing status parameters

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

RV

**SEE ALSO**

*ft, ftf [▷ 61]*, *trf, trfp [▷ 89]*

## 3.30 sab

**NAME**

sab - Spline baseline correction (1D)

**DESCRIPTION**

The command **sab** performs a spline baseline correction. This is based on a predefined set of data points which are considered to be a part of the baseline. The regions between these points are individually fitted. In order to execute **sab**, the baseline points must have been determined. You can do this as follows:

- Click  or enter **.basl** to change to baseline correction mode.
- Click  to switch to *Define baseline points* mode
- (if the baseline points have been defined before, you are first prompted to append to (a) or overwrite (o) the existing list of points)
- Move the cursor along the spectrum and click left at several positions which are part of the baseline.
- Click  to return. The command **sab** is automatically executed.

The set of baseline points is saved in the file *baslpnts*. This file can be stored for general usage with the command **wmisc.** After that, you can read it with **rmisc** on another dataset and run **sab** to perform the same baseline correction.

**sab** can be started from the command line or from the baseline dialog box which is opened with the command **bas**.

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r* - real processed 1D data

*baslpnts* - baseline points (points and ppm values)

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r* - real processed 1D data

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

SAB

### SEE ALSO

*bcm [*▸ 55], (bas, .basl)

## 3.31 sref, cal

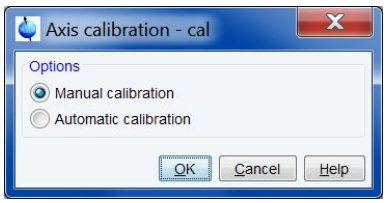
### NAME

sref - Calibrate the spectrum; set the TMS signal to 0 ppm (1D,2D)

cal - Open calibration dialog box (1D, 2D)

### DESCRIPTION

Spectrum calibration can be started from the command line with **sref** or from the calibration dialog box which is opened with the **cal** command.




This dialog box offers two options, one for manual and one for automatic calibration.

#### Manual calibration

This option selects the **.cal** command for execution. This is equivalent to clicking  in the toolbar and switches to interactive calibration mode. Click inside the data window at the reference peak, enter the frequency value in the appearing dialog box and click **OK**.

#### Automatic calibration

This option selects the **sref** command for execution. It calibrates the spectrum by setting the TMS signal of a spectrum to exactly 0 ppm. It works on 1D and 2D spectra.

**sref** makes use of the lock table. This must be set up once after installing TopSpin with the command **edlock**.

On 1D spectra, **sref** involves three steps which are discussed below.

During the first step **sref** sets the value of the processing parameter SF according to the formula:

SF=BF1/(1.0+RShift * 1e-6)

Where *RShift* is taken from the **edlock** table and BF1 is an acquisition status parameter. Changing SF automatically changes the processing parameters SR, the spectral reference, and OFFSET, the ppm value of the first data point, according to the following relations:

- SR = SF - BF1 - where BF1 is an acquisition status parameter.
- OFFSET = (SFO1/SF-1) * 1.0e6 + 0.5 * SW * SFO1/SF - where SW and SFO1 are acquisition status parameters

Actually, the relation for OFFSET depends on the acquisition mode. When the acquisition status parameter AQ_mod is *qsim*, *qseq* or *DQD*, which is usually the case, the above relation count. When AQ_mod is *qf*, the relation OFFSET = (SFO1/SF-1) * 1.0e6 is used.

**sref** then calculates which data point (between 0 and SI) in your spectrum corresponds to the ppm value *Ref.* from the **edlock** table. This data point will be used in the second step. The first step is independent of a reference substance.

During the second step, **sref** scans a region around the data point found in the first step for a peak. It will normally find the signal of the reference substance. The width of the scanned region is defined by the parameter *Width* in **edlock** table, so this region is *Ref. +/- 0.5*Width* ppm. This step is necessary because the lock substance (solvent) will not always resonate at exactly the same position relative to the reference shift. The absolute chemical shift of the lock substance (solvent) differs because of differences in susceptibility, temperature, concentration or pH, for instance.

The third step depends on whether or not a peak was found in the second step. If a peak was found, **sref** determines the interpolated peak top and shifts its ppm value to the *ref.* value from the **edlock** table. The processing parameters OFFSET, SF and SR are changed accordingly. As such, the result of the default (step 1) is slightly corrected in order to set the peak of the reference substance exactly to 0. You can check this by putting the cursor on this peak. If no peak was found, you will get the message: **sref: no peak found default calibration done**. The result of the default calibration (step 1) is stored without any further correction.

The three cases below show the calibration of a 1H, 13C and 31P spectrum with C6D6 as a solvent. The following table shows the corresponding entry in the **edlock** table:

| Solvent | Field | Lockpower | Nucleus | Distance [ppm] | Ref. [ppm] | Width [ppm] | Rshift [ppm] |
|---|---|---|---|---|---|---|---|
| C6D6 | -150 | -15.0 | | | | | |
| | | | 1H | 7.28 | 0.0 | 0.5 | 0.000 |
| | | | 2H | 7.28 | 0.0 | 0.5 | 0.000 |
| | | | 13C | 128.0 | 0.0 | 5.0 | 0.220 |
| | | | 31P | 0.00 | 10.5 | 5.0 | 13.356 |

**Case #1** - Calibration of a 1H spectrum: A spectrum was acquired while being locked on C6D6. **sref** will do a default calibration and look for a signal at 0.0 ppm ( *Ref.*) in a window of +/- 0.25 ppm. If a peak is found, its chemical shift will be set to 0 ppm.

**Case #2** - Calibration of a 13C spectrum: A spectrum was acquired while being locked on C6D6. **sref** will do a default calibration and look for a signal at 0.0 ppm ( *Ref.*) in a window of +/- 2.5 ppm. If a peak is found, its chemical shift will be set to 0 ppm.

**Case #3** - Calibration of a 31P spectrum: A spectrum was acquired while being locked on C6D6. **sref** will do a default calibration and look for a signal at 10.5 ppm ( *Ref.*) in a window of +/- 2.5 ppm. If a peak is found, its chemical shift will be set to exactly 10.5 ppm.

On 2D spectra, **sref** calibrates the F2 and F1 direction and this involves the same steps as described above for 1D spectra.

Please note that the purpose of **sref** is the following:

- If TMS (or any other reference substance) is found, the value is ignored.
- If there is no TMS (or any other reference substance), than **sref** sets SR to 0, which basically makes BF1 = SF. This is normally pragmatic, but in special cases it is necessary to enter a different value, in order to get a useful resulting chemical shift. Entering a value here will correct the chemical shift by the amount specified.

**INPUT PARAMETERS**

Set by the acquisition, can be viewed with **dpa** or by typing **s solvent** etc.:

SOLVENT - the solvent of the sample

INSTRUM - configuration name (entered during **cf**) of the spectrometer

LOCNUC - lock nucleus

SFO1 - spectral frequency

NUC1 - measured nucleus

SW - sweep width

**OUTPUT PARAMETERS**

Processing parameters which can be viewed with **edp**

Processing status parameters which can be viewed with **dpp**

SF - spectral reference frequency

OFFSET - the ppm value of the first data point of the spectrum

SR - spectral reference

**INPUT FILES**

*<tshome>/conf/instr/<instrum>/*

*2Hlock* - edlock table for 2H locked samples

*19Flock* - edlock table for 19F locked samples

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*proc* - processing parameters

*procs* - processing status parameters

**USAGE IN AU PROGRAMS**

SREF

## 3.32 tm, traf, trafs

**NAME**

tm - Trapezoidal window multiplication of the FID (1D)

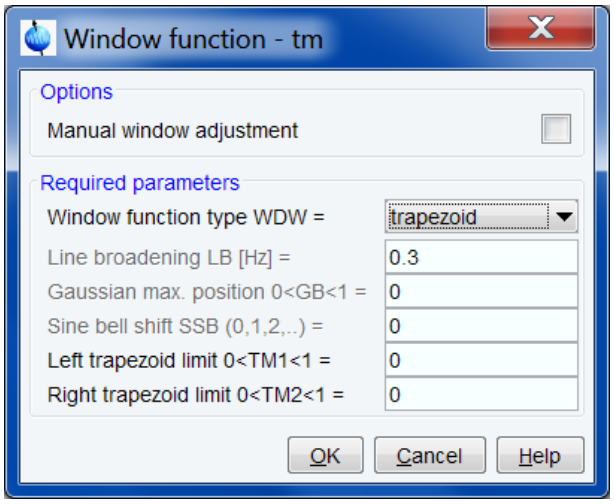
traf - Traficante window multiplication of the FID (1D)

trafs - Similar to traf, but trafs additionally retains optimum signal-to-noise.

wm - Open window function dialog box (1D,2D)

**DESCRIPTION**

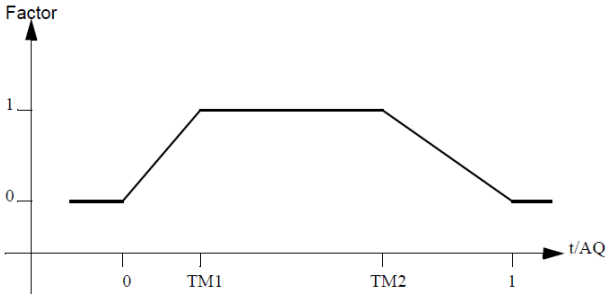
Window multiplication can be executed from the command line or from the window function dialog box. The latter is opened with the command **wm**:




This dialog box offers several window functions, each of which selects a certain command for execution.

### Trapezoid

This function selects the command **tm** for execution. It performs a trapezoidal window multiplication of the FID. The rising and falling edge of this function are defined by the processing parameters TM1 and TM2. These represent a fraction of the acquisition time as displayed below.




### Traficante and trafic.s/n

This function selects the commands **traf** and **trafs** , respectively, for execution. The algorithms used by these commands are described by D. D. Traficante and G. A. Nemeth in J. Magn. Res., 71**,** 237 (1987).

**tm**, **traf** and **trafs** implicitly perform a baseline correction of the FID, according to the processing parameter BC_mod. Furthermore, they perform linear prediction according to the parameters ME_mod, NCOEF and LPBIN.

When executed on 2D or 3D data, **tm** and **traf*** take up to four arguments, e.g. **tm <row> <procno> n y** process the specified row and store it under the specified *procno*. The last two arguments are optional: **n** prevents changing the display to the output 1D data, **y** causes a possibly existing data to be overwritten without warning.

If you run a command like **tm** from the command line, you have to make sure that the required parameters are already set. Click the Procpars tab or enter **edp** to do that.

When executed on a dataset with 2D or 3D raw data but 1D processed data (usually a result of rsr, rsc or a previous 1D processing command on that 2D or 3D data), **tm** and **traf*** take one argument, e.g. **tm <row>** process the specified row and store it under the current *procno*.

**tm same** process the same row as the previous processing command and store it under the current *procno*. The **same** option is automatically used by the AU program macro TM. When used on a regular 1D dataset (i.e. with 1D raw data) it has no effect.

The **wm** command can be used on 1D or 2D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

#### INPUT PARAMETERS

Set from the **wm** dialog box, with **edp** or by typing **tm1**, **lb** etc.:

TM1 - the end of the rising edge of a trapeziodal window (input of **tm**)

TM2 - the start of the falling edge of a trapezoidal window (input of **tm**)

LB - Lorentzian broadening factor (input of **traf***)

Set by the acquisition, can be viewed with **dpa** or **s aq**:

AQ - acquisition time (input of **tm**)

#### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - raw data (input if *1r*, *1i* do not exist or are Fourier transformed)

*acqus* - acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed data (input if they exist but are not Fourier transformed)

*proc* - processing parameters

#### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

*procs* - processing status parameters

*auditp.txt* - processing audit trail

#### USAGE IN AU PROGRAMS

TM

#### SEE ALSO

*em, gm, wm [▶ 57]*, *sinm, qsin, sinc, qsinc [▶ 78]*

## 3.33 trf, trfp

#### NAME

trf - User defined processing of raw data (1D)

trfp - User defined processing of processed data (1D)

#### DESCRIPTION

The command **trf** processes the raw data performing the following steps:

- baseline correction according to BC_mod
- linear prediction according to ME_mod
- window multiplication according to WDW
- Fourier transform according to FT_mod

- phase correction according to PH_mod

**trf** offers the following features:

- when all parameters mentioned above are set to *no*, the raw data (file *fid*) are simply stored as processed data (files *1r*, *1i*). The even points are stored as real data (file *1r*) and the odd points as imaginary data (file *1i*). The size of these processed data and the number of input FID points are determined by the parameters SI and TDeff, as described for the command **ft**. For example, if 0 < TDeff < TD, the processed data are truncated. This allows to create an FID with a smaller size than the original one (see also the command **genfid** *genfid [▷ 64]*).
- **trf** evaluates BC_mod for the baseline correction mode (e.g. quad, qpol or qfil) and detection mode (e.g. single or quad, spol or qpol, sfil or qfil). Note that the command **bc** evaluates the acquisition status parameter AQ_mod for the detection mode and ignores the BC_mod detection mode (see parameter BC_mod).
- **trf** evaluates WDW for the window multiplication mode (*em*, *gm*, *sine*, *qsine*, *trap*, *user*, *sinc*, *qsinc*, *traf* or *trafs*). This allows to vary the window multiplication by varying the value of WDW rather than the window multiplication command. This can be useful in AU programs.
- the Fourier transform is performed according to FT_mod. Normally, the Fourier transform is done with the command **ft** which determines the Fourier transform mode from acquisition status parameter AQ_mod. However, for some datasets, no value of AQ_mod translates to a correct Fourier transform mode. An example of this is when you read a column (with **rsc**) from a 2D dataset which was measured with FnMODE (or MC2) = States-TPPI and Fourier transformed in the F2 direction only. The resulting FID can only be Fourier transformed correctly with **trf**. The parameter FT_mod is automatically set to the correct value by the **rsc** command. **trf** can also be used manipulate the acquisition mode of raw data by Fourier transforming the data with one FT_mod and inverse Fourier transforming them with a different FT_mod. From the resulting data you could create pseudo-raw data (using **genfid**) with a different acquisition mode than the original raw data. Finally, **trf** allows to process the data without Fourier transform (FT_mod = no). The following table shows a list of FT_mod values:

| FT_mod | Fourier transform mode |
|---|---|
| no | no Fourier transform |
| fsr | forward, single channel, real |
| fqr | forward, quadrature, real |
| fsc | forward, single channel, complex |
| fqc | forward, quadrature, complex |
| isr | inverse, single channel, real |
| iqr | inverse, quadrature, real |
| isc | inverse, single channel, complex |
| iqc | inverse, quadrature, complex |

The command **trfp** works like **trf**, except that it always works on processed data. If no processed data exist, **trfp** stops with an error message.

**trfp** can be used to perform multiple additive baseline corrections, to remove multiple frequency baseline distortions. This cannot be done with **bc** or **trf** because these commands always work on the raw data, i.e. they are not additive. Note that the window multiplication commands (e.g. **em**, **gm**, **sine** etc.) are additive. The same counts for linear prediction (part of **ft**) and phase correction (**pk**).

**trf** can be used to do a combination of forward and backward prediction. Just run **trf** with ME_mod = LPfc and then **trfp** (or **ft**) with ME_mod = LPbc.

When executed on a 2D or 3D dataset, **trf** takes up to four arguments:

**trf <row> <procno> n y**

process the specified row and store it under the specified *procno*. The last two arguments are optional: **n** prevents changing the display to the output 1D data, **y** causes a possibly existing data to be overwritten without warning.

When executed on a dataset with 2D or 3D raw data but 1D processed data (usually a result of rsr, rsc or a previous 1D processing command on that 2D or 3D data), **trf** takes one argument **trf <row>** process the specified row and store it under the current *procno*.

**trf same** process the same row as the previous processing command and store it under the current *procno*. The **same** option is automatically used by the AU program macro TRF. When used on a regular 1D dataset (i.e. with 1D raw data), it has no effect.

## INPUT PARAMETERS

Set by the user with **edp** or by typing **si**, **tdeff** etc.:

SI - size of the processed data

TDeff - number of raw data points to be used for processing

FCOR - first (FID) data point multiplication factor (0.0-2.0, default 0.5)

BC_mod - FID baseline correction mode

BCFW - filter width for BC_mod = sfil or qfil

COROFFS - correction offset for BC_mod = spol/qpol or sfil/qfil

ME_mod - FID linear prediction mode

NCOEF - number of linear prediction coefficients

LPBIN - number of points for linear prediction

TDoff - number of raw data points predicted for ME_mod = LPb*

WDW - FID window multiplication mode

LB - Lorentzian broadening factor for WDW = em or gm

GB - Gaussian broadening factor for WDW = gm, sinc or qsinc

SSB - Sine bell shift for WDW = sine, qsine, sinc or qsinc

TM1, TM2 - limits of the trapezoidal window for WDW = trap

FT_mod - Fourier transform mode

REVERSE - flag indicating to reverse the spectrum

PKNL - group delay compensation (Avance) or filter correction (A*X)

STSR - strip start: first output point of strip transform

STSI - strip size: number of output points of strip transform

PH_mod - phase correction mode

PHC0 - zero order phase correction value for PH_mod = pk

PHC1 - first order phase correction value for PH_mod = pk

Set by the acquisition, can be viewed with **dpa** or by typing **s td** :

TD - time domain; number of raw data points

## OUTPUT PARAMETERS

Can be viewed with **dpp** or by typing **s tdeff** etc.:

TDeff - number of raw data points that were used for processing

STSR - strip start: first output point of strip transform

STSI - strip size: number of output points of strip transform

NC_proc - intensity scaling factor

YMAX_p - maximum intensity of the processed data

YMIN_p - minimum intensity of the processed data

BYTORDP - data storage order

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - raw data (input of **trf**)

*acqus* - F2 acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (input of **trfp**)

*proc* - processing parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data

*procs* - processing status parameters

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

TRF

TRFP

**SEE ALSO**

*bc* [▷ 53], *em, gm* [▷ 57], *pk* [▷ 71], *ft, ftf* [▷ 61]

## 3.34 zf

**NAME**

zf - Zero all data points (1D)

**DESCRIPTION**

The command **zf** sets the intensity of all data points to zero. Depending on the value of the parameter DATMOD, **zf** works on raw or processed data. The result is always stored as processed data, the raw data are never overwritten.

The output of **zf** is usually the same for DATMOD = raw or processed, namely SI processed data points with zero intensity. However, for DATMOD = proc, the existing processed data are set to zero whereas for DATMOD = raw, new processed data are created according to the current processing parameters. The result is different when the data have been Fourier transformed with STSI < SI. **zf** with DATMOD = proc creates STSI zeroes whereas **zf** with DATMOD = raw creates SI zeroes. The reason is that **zf** with DATMOD = raw reprocesses the raw data but does not interpret STSI since no Fourier transform is done.

**INPUT PARAMETERS**

Set by the user with **edp** or by typing **datmod**, **si** etc.:

DATMOD - data mode: work on raw or processed data

SI - size of the processed data

STSI - strip size (input if DATMOD = proc)

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - raw data (input if DATMOD = raw)

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (input if DATMOD = proc)

*proc* - processing parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data

*procs* - processing status parameters

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

ZF

**SEE ALSO**

*zp* [ 93]

## 3.35 zp

**NAME**

zp - Zero the first NZP data points (1D)

**DESCRIPTION**

The command **zp** sets the intensity of the first NZP points of the dataset to zero. It works on raw or processed data depending on the value of the parameter DATMOD. The parameter NZP can take a value between 0 and the size of the FID or spectrum.

The value of NZP is the number of the real plus imaginary data points that are zeroed. As such, the first (NZP+1)/2 real points and the first NSP/2 imaginary data points are zeroed.

**INPUT PARAMETERS**

Set by the user with **edp** or by typing **nzp**, **datmod** etc.:

NZP - number of data points set to zero intensity

DATMOD - data mode: work on raw or processed data

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - raw data (input if DATMOD = raw)

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (input if DATMOD = proc)

*proc* - processing parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data (real, imaginary)

*procs* - processing status parameters

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

ZP

**SEE ALSO**

*zf* [▸ 92]

# 4 2D Processing Commands

This chapter describes all TopSpin 2D processing commands. Most of them only work on 2D data but some, e.g. **xfb**, can also be used to process a plane of 3D data. They store their output in processed data files and do not change the raw data.

We will often refer to the two directions of a 2D dataset as the F2 and F1 direction. F2 is the acquisition direction which is displayed horizontally and F1 the orthogonal direction which is displayed vertically. The names of most 2D processing commands express the direction in which they work, e.g. **xf2** works in F2, **xf1** in F1 and **xfb** in both directions. F2 traces are usually referred to as rows, F1 traces as columns. Some commands express this terminology, e.g. **rsr** reads and stores rows and **rsc** reads and stores columns of a 2D spectrum.

For each command, the relevant input and output parameters are mentioned. Furthermore, the relevant input and output files and their location are mentioned. Although file handling is completely transparent, it is sometimes useful to know which files are involved and where they reside. For example, if you have permission problems or if you want to process or interpret your data with third party software.

## 4.1 abs2, abst2, absd2, absot2

### NAME

abs2 - Automatic baseline correction in F2 (2D)

abst2 - Automatic selective baseline correction in F2 (2D)

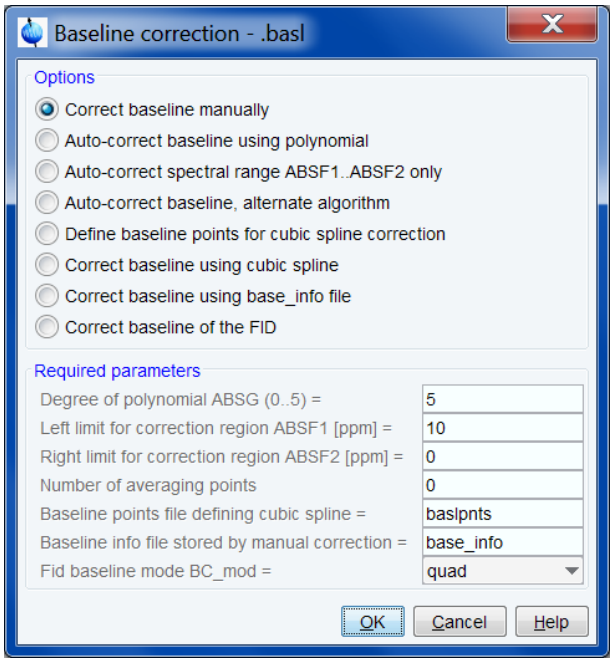
absd2 - Automatic baseline correction in F2, diff. algorithm (2D)

absot2 - Automatic selective baseline correction in F2, diff. algorithm (2D)

bas - Open baseline correction dialog box (1D,2D)

### DESCRIPTION

Baseline correction commands can be started from the command line, by entering **abs2**, **abst2** etc. or from the baseline dialog box. The latter is opened with the command **bas**:




This dialog box offers several options, each of which selects a certain command for execution. The command further depends on the selected direction. Here we describe the commands for the F2 direction.

### F2 Auto-correct baseline using polynomial

This option selects the command **abs2** for execution. It performs an automatic baseline correction in the F2 direction. This means it subtracts a polynomial from the rows of the processed 2D data. The degree of the polynomial is determined by the parameter ABSG which has a value between 0 and 5, with a default of 5. It works like **absf** in 1D which means it only corrects the spectral region between ABSF1 and ABSF2.

### F2 Auto-correct baseline, shift correction region

This option selects the command **abst2** for execution. It performs an automatic selective baseline correction in the F2 direction. This means it corrects the rows of the processed 2D data. It works like **abs2**, except for the following:

- only the rows between F1-ABSF2 and F1-ABSF1 are corrected
- the part (region) of each row which is corrected shifts from row to row. The first row is corrected between F2-ABSF2 and F2-ABSF1. The last row is corrected between F2-SIGF2 and F2-SIGF1. For intermediate rows, the low field limit is an interpolation of F2-ABSF2 and F2-SIGF2 and the high field limit is an interpolation of F2-ABSF1 and F2-SIGF1.

### F2 Auto-correct baseline, alternate algorithm

This option selects the command **absd2** for execution. It works like **abs2**, except that it uses a different algorithm (it uses the same algorithm as the command abs in DISNMR). It is, for example, used when a small peak lies on the foot of a large peak. In that case, **absd2** allows to correct the baseline around the small peak which can then be integrated. Usually **absd2** is followed by **abs2**.

### F2 Auto-correct baseline, shift correction region, alternate algorithm

This option selects the command **absot2** for execution. It works like **abst2**, except that it has a different algorithm which applies a larger correction.

If you run a command like **abs2** from the command line, you have to make sure that the required parameters are already set. Click the Procpars tab or enter **edp** to do that.

The **bas** command can be used on 1D, 2D or 3D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

### INPUT PARAMETERS

Set from the **bas** dialog box, with **edp** or by typing **absg**, **absf1** etc.:

ABSG - degree of the polynomial to be subtracted (0 to 5, default is 5)

ABSF1 - low field limit of the region which is baseline corrected

ABSF2 - high field limit of the region which is baseline corrected

SIGF1 - low field limit of the correction region in the last row

SIGF2 - high field limit of the correction region in the last row

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

*proc* - F2 processing parameters

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

*procs* - F2 processing status parameters

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

ABS2

ABST2

ABSD2

ABSOT2

### SEE ALSO

*abs1, abst1, absd1, absot1, bas [▷ 97]*

## 4.2 abs1, abst1, absd1, absot1, bas

### NAME

abs1 - Automatic baseline correction in the F1 (2D)

abst1 - Automatic selective baseline correction in the F1 (2D)

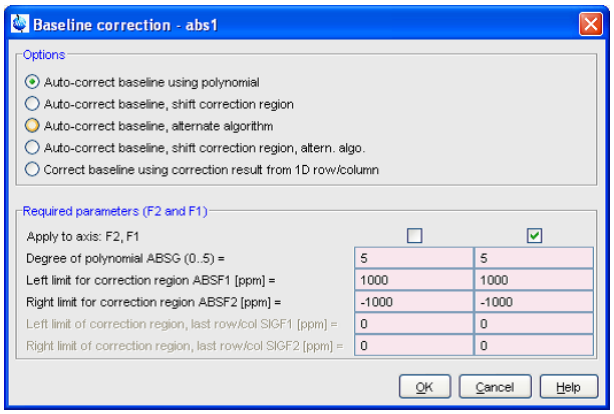
absd1 - Automatic baseline correction in F1, diff. algorithm (2D)

absot1 - Automatic selective baseline correction in F1, diff. algorithm (2D)

bas - Open baseline correction dialog box (1D,2D)

### DESCRIPTION

Baseline correction can be started from the command line, with **abs1**, **abst1** etc., or from the baseline dialog box. The latter is opened with the command **bas**




This dialog box offers several options, each of which selects a certain command for execution. The command further depends on the selected direction. Here we describe the commands for the F1 direction.

### F1 Auto-correct baseline using polynomial

This option selects the command **abs1** for execution. It performs an automatic baseline correction in the F1 direction. This means it subtracts a polynomial from the columns of the processed 2D data. The degree of the polynomial is determined by the parameter ABSG which has a value between 0 and 5, with a default of 5. It works like **absf** in 1D which means it only corrects the spectral region between ABSF1 and ABSF2.

### F1 Auto-correct baseline, shift correction region

This option selects the command **abst1** for execution. It performs an automatic selective baseline correction in the F1 direction. This means it corrects the columns of the processed 2D data. It works like **abs1**, except for the following:

- only the columns between F2-ABSF2 and F2-ABSF1 are corrected
- the part (region) of each column which is corrected shifts from column to column. The first column is corrected between F1-ABSF2 and F1-ABSF1. The last column is corrected between F1-SIGF2 and F1-SIGF1. For intermediate columns, the low field limit is an interpolation of F1-ABSF2 and F1-SIGF2 and the high field limit is an interpolation of F1-ABSF1 and F1-SIGF1.

### F1 Auto-correct baseline, alternate algorithm

This option selects the command **absd1** for execution. It works like **abs1**, except that it uses a different algorithm. It is, for example, used when a small peak lies on the foot of a large peak. In that case, **absd1** allows to correct the baseline around the small peak which can then be integrated. Usually **absd1** is followed by **abs1**.

### F1 Auto-correct baseline, shift correction region, alternate algorithm

This option selects the command **absot1** for execution. It works like **abst1**, except that it has a different algorithm which applies a larger correction.

If you run a command like **abs1** from the command line, you have to make sure that the required parameters are already set. Click the Procpars tab or enter **edp** to do that.

The **bas** command can be used on 1D or 2D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

### INPUT PARAMETERS

Set from the **bas** dialog box, with **edp** or by typing **absf1**, **absf2** etc.:

ABSG - degree of the polynomial to be subtracted (0 to 5, default is 5)

ABSF1 - low field limit of the correction region in the first row

ABSF2 - high field limit of the correction region in the first row

SIGF1 - low field limit of the correction region in the last row

SIGF2 - high field limit of the correction region in the last row

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

*proc2* - F1 processing parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

*proc2s* - F1 processing status parameters

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

ABS1

ABST1

ABSD1

ABSOT1

**SEE ALSO**

*abs2, abst2, absd2, absot2 [* 95]

## 4.3 add2d, mul2d, addser

**NAME**

add2d - Add or subtract two datasets (2D)

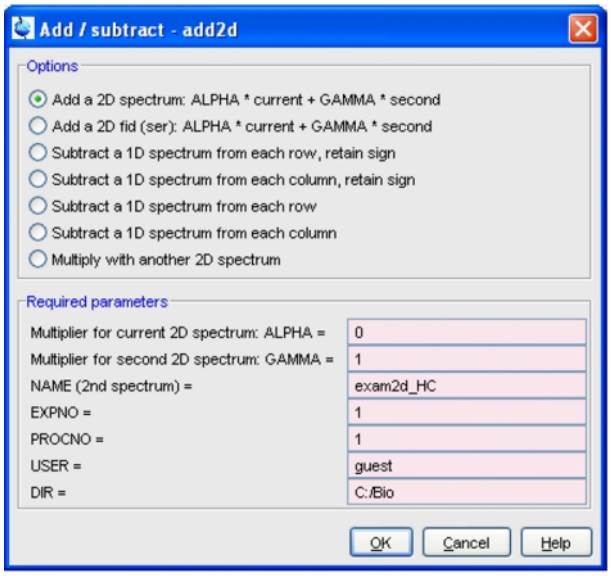
mul2d -Multiply two datasets (2D)

addser - Add two raw datasets (2D, 3D)

adsu - Open add/subtract/multiply dialog box (1D, 2D)

**DESCRIPTION**

Addition commands can be started from the command line or from the add/subtract dialog box. The latter is opened with the command **adsu**.




This dialog box offers several options, each of which selects a certain command for execution.

### Add a 2D spectrum

This option selects the command **add2d** for execution. It adds the processed data of the second dataset to those of the current 2D dataset, according to the following formula:

current = ALPHA*current + GAMMA*second

Where ALPHA and GAMMA are processing parameters. Both real and imaginary data are added. The result overwrites the current processed data. For APLHA = 1 and GAMMA = -1, the spectra are subtracted.

### Multiply with another 2D spectrum

This option selects the command **mul2d** for execution. It multiplies the processed data of the second dataset with those of the current 2D dataset. Both real and imaginary data are multiplied. The result overwrites the current processed data.

### Add 2D fid (ser)

This option selects the command **addser** for execution. It adds the raw data of the second dataset to those of the current 2D dataset. The result overwrites the current raw data. Note that **addser** also works on 3D data.

The two 2D datasets to be added or multiplied must have equal sizes.

If you run a command like **add2d** from the command line, you have to make sure that the required parameters are already set. Click the Procpars tab or enter **edp** to do that. If the second dataset has not been defined yet, **add2d** opens the add/subtract (**adsu**) dialog box.

The **adsu** command can be used on 1D or 2D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

### INPUT PARAMETERS

Set from the **adsu** dialog box, with **edp** or by typing **alpha**, **gamma** etc.:

ALPHA - multiplication factor of the current spectrum

GAMMA - multiplication factor of the second spectrum

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *2ir*, *2ri*, *2ii* - processed data of the current dataset

*proc* - F2 processing parameters

*<dir2>/data/<user2>/nmr/<name2>/<expno2>/pdata/<procno2>/*

*2rr*, *2ir*, *2ri*, *2ii* - processed data of the second dataset

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *2ir*, *2ri*, *2ii* - processed data

*procs* - F2 processing status parameters

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

ADD2D

ADDSER

MUL2D

**SEE ALSO**

*add, duadd, addfid, addc, adsu [ 44]*, *mul, mulc, nm, div [ 69]*

## 4.4 bcm2, bcm1

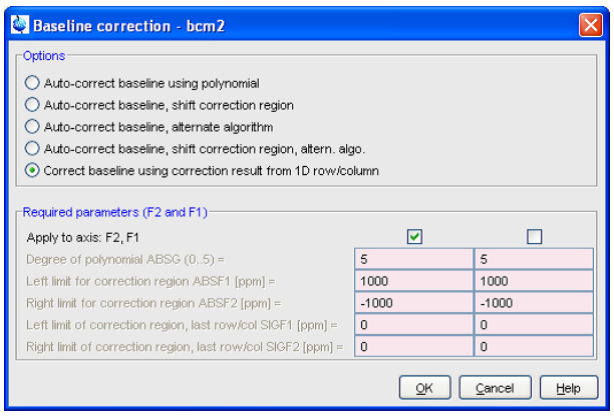
**NAME**

bcm2 - User defined baseline correction in F2 (2D)

bcm1 - User defined baseline correction in F1 (2D)

**DESCRIPTION**

Baseline correction commands can be started from the command line or from the baseline dialog box. The latter is opened with the command **bas**:



This dialog box offers several options, each of which selects a certain command for execution.

**Correct baseline, using correction result from 1D row/column (F2)**

This option selects the command **bcm2** for execution. It performs a baseline correction in the F2 direction by subtracting a polynomial, sine or exponential function. Before you can use **bcm2**, you must first do the following:

1. Read a row with **rsr** (TopSpin will switch to the 1D data window)
2. Click or enter **.basl** to switch to baseline mode.
3. Click , or to select the baseline correction function.
4. Fit the baseline of the spectrum with the function you selected in step 2 (initially represented by a straight horizontal line). Click-hold button *A* and move the mouse to determine the zero order correction. Do the same with the buttons *B*, *C* for higher order corrections until the line matches the baseline of the spectrum.
5. Click to save the baseline correction to the 2D dataset and leave baseline mode.
6. Select the 2D data window.

Then you can enter **bcm2** to perform the baseline correction.

**Correct baseline, using correction result from 1D row/column (F1)**

This option selects the command **bcm1** for execution. It works like **bcm2**, except that it performs a baseline correction in the F1 direction (columns). Before you can use **bcm1**, you must read a column with **rsc** and define the baseline on it (see above).

**bcm*** commands only works on the real data. After applying them, the imaginary data no longer match the real data and cannot be used for phase correction.

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

*base_info* - baseline correction coefficients

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

BCM2

BCM1

**SEE ALSO**

*abs1, abst1, absd1, absot1, bas* [▷ 97], *abs2, abst2, absd2, absot2* [▷ 95], *bcm2, bcm1* [▷ 101]

## 4.5 f2disco, f1disco

**NAME**

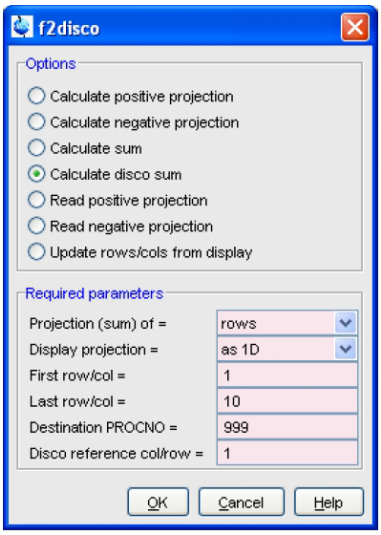
f2disco - Calculate disco projection in F2 (2D)

f1disco - Calculate disco projection in F1 (2D)

proj - Open projections dialog box (2D,3D)

## DESCRIPTION

The disco projection commands open the projections dialog box the corresponding command:




This dialog box has several options, each of which selects a certain command for execution.

### Calculate disco sum (of rows)

This option selects the command **f2disco** for execution. Like **f2sum**, it calculates the sum of all rows between *firstrow* and *lastrow*. However, for each row, the intensity at the intersection with the reference column is determined. If this intensity is positive, the row is added to the total. If it is negative, the row is subtracted from the total.

### Calculate disco sum (of columns)

This option selects the command **f1disco** for execution. It works like **f2disco**, except that it calculates the sum of the specified columns considering the intensities at the intersections with a reference row.

The calculated disco sum is stored under the specified *Destination procno.*

The Required parameter *Display projection* can be set to:

- *on 2D* to display the calculated projection with the 2D dataset. The current 2D dataset remains the active dataset.
- *as 1D* to display the calculated projection as a 1D dataset. The active dataset changes to the destination *procno*.

The required parameters can also be specified as arguments on the command line. As an example we use the command **f2disco** here.

**f2disco <firstrow>** prompts for *lastrow* and *refrow* and stores the disco projection under data *name* ~TEMP

**f2disco <firstrow> <lastrow> <refrow>** stores the specified disco projection under data *name* ~TEMP

**f2disco <firstrow> <lastrow> <refrow> <procno>** stores the specified disco projection under the specified *procno* of the current data *name*

**f2disco <firstrow> <lastrow> <refcol> <procno> n** stores the specified disco projection under the specified *procno* of the current data *name* but does not change the display to this *procno*

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *2ir*, *2ri*, *2ii* - processed data

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i*- 1D spectrum containing the F1 disco projection

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

F2DISCO(firstrow, lastrow, refcol, procno)

F1DISCO(firstcol, lastcol, refrow, procno)

For *procno* = -1, the disco projection is written to the dataset ~TEMP

### SEE ALSO

*f2projn, f2projp* [ 104], *f2sum, f1sum* [ 106], *rhpp, rhnp* [ 111]

## 4.6 f2projn, f2projp, f1projn, f1projp

### NAME

f2projn - Calculate negative partial projection in F2 (2D)

f2projp - Calculate positive partial projection in F2 (2D)

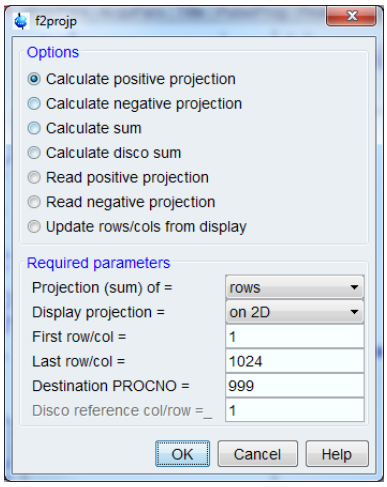
f1projn - Calculate negative partial projection in F1 (2D)

f1projp - Calculate positive partial projection in F1 (2D)

proj - Open projections dialog box

### DESCRIPTION

The projection commands open the projections dialog box selecting the corresponding command.



This dialog box has several options, each of which selects a certain command for execution.

**Calculate positive projection (of rows)**

This option selects the command **f2projp** for execution. It calculates the positive partial 1D projection of the 2D dataset in the F2 direction

**Calculate positive projection (of columns)**

This option selects the command **f1projp** for execution. It calculates the positive partial 1D projection of the 2D dataset in the F1 direction

**Calculate negative projection (of rows)**

This option selects the command **f2projn** for execution. It calculates the negative partial 1D projection of the 2D dataset in the F2 direction

**Calculate negative projection (of columns)**

This option selects the command **f1projn** for execution. It calculates the negative partial 1D projection of the 2D dataset in the F1 direction

The calculated projection is stored under the specified *Destination procno.*

The Required parameter *Display projection* can be set to:

- *on 2D* to display the calculated projection with the 2D dataset. The current 2D dataset remains the active dataset.
- *as 1D* to display the calculated projection as a 1D dataset. The active dataset changes to the destination PRONCNO.

The required parameters can also be specified as arguments on the command line. As an example we use the command **f2projn** here.

**f2projn <firstrow>** prompts for *lastrow* and stores the projection under data *name* ~TEMP

**f2projn <firstrow> <lastrow>** stores the specified projection under data *name* ~TEMP

**f2projn <firstrow> <lastrow> <procno>** stores the specified projection under the specified *procno* of the current data *name*

**f2projn <firstrow> <lastrow> <procno> n** stores the specified projection under the specified *procno* of the current data *name* but does not change the display to this *procno*

A projection is a 1D trace where every point has the highest intensity of all points of the corresponding orthogonal trace in the 2D spectrum. Partial means that only a specified range of rows (or columns) is are evaluated, i.e. only a part of the orthogonal trace is scanned for the highest intensity. Negative projections contain only negative intensities, positive projections contain only positive intensities.

A special case is the command **f1projp** or **f1projn** on a hypercomplex 2D dataset (MC2 ≠ QF) that has been processed in F2 only. Suppose you would perform the following command sequence:

**xf2** - to process the data in F2 only.

**s si** - to check the F1 size of the 2D data, click **Cancel**.

**s mc2** - to check status MC2 (≠ QF), click **Cancel**.

**f1projp** - to store the F1 projection in ~TEMP and change to that dataset.

**s si** - to check the size of the resulting 1D dataset, click **Cancel** .

You will see that the size of the 1D data is only half the F1 size of the 2D data. The reason is that **f1projp** unshuffles the input data (file *2rr*). As such, **f1projp** behaves like the command **rsc**. If you want to prevent the unshuffling of the input data (file *2rr*), you can use the following trick. Set the status parameter MC2 to QF before you run **f1projp**:

**s mc2** , click **QF**

Then, the size of the 1D data will be the same as the F1 size of the 2D data.

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - processed data

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*f2projn* - ascii file specifying the range of rows and the 1D data path

*f2projp* - ascii file specifying the range of rows and the 1D data path

*f1projn* - ascii file specifying the range of columns and the 1D data path

*f1projp* - ascii file specifying the range of columns and the 1D data path

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r* - 1D spectrum containing the projection

*auditp.txt* - processing audit trail

If the commands are used with less than three arguments, the files are stored in:

*<dir>/data/<user>/nmr/~TEMP/1/pdata/1/*

**USAGE IN AU PROGRAMS**

F2PROJN(firstrow, lastrow, procno)

F2PROJP(firstrow, lastrow, procno)

F1PROJN(firstcol, lastcol, procno)

F1PROJP(firstcol, lastcol, procno)

For all these macros counts that if procno = -1, the projection is written to the dataset ~TEMP

**SEE ALSO**

*f2disco, f1disco* [▶ 102], *f2sum, f1sum* [▶ 106], *rhpp, rhnp* [▶ 111]

## 4.7 f2sum, f1sum, proj

**NAME**

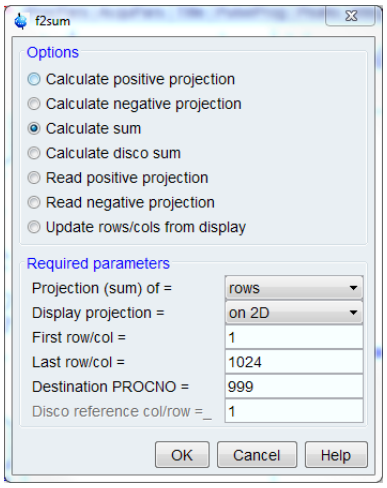
f2sum - Calculate partial sum in F2 (2D)

f1sum - Calculate partial sum in F1 (2D)

proj - Open the projections dialog box (2D,3D)

**DESCRIPTION**

The projection sum commands open the projections dialog box selecting the corresponding command.




This dialog box has several options, each of which selects a certain command for execution.

**Calculate sum (of rows)**

This option selects the command **f2sum** for execution. It calculates the sum of all rows within a region specified by the parameters. The result is divided by the number of rows. This means, that in the fact a mean row is calculated.

**Calculate sum (of columns)**

This option selects the command **f1sum** for execution. It calculates the sum of all columns within a region specified by the parameters. The result is divided by the number of columns. This means, that in the fact a mean column is calculated.

The calculated column is stored under the specified *Destination procno.*

The Required parameter *Display projection* can be set to:

- *on 2D* to display the calculated projection with the 2D dataset. The current 2D dataset remains the active dataset.
- *as 1D* to display the calculated projection as a 1D dataset. The active dataset changes to the destination *procno*.

The required parameters can also be specified as arguments on the command line. As an example we use the command **f2sum** here.

**f2sum <firstrow>** prompts for *lastrow* and stores the sum under data *name* ~TEMP

**f2sum <firstrow> <lastrow>** stores the specified sum under data *name* ~TEMP

**f2sum <firstrow> <lastrow> <procno>** stores the specified sum under the specified *procno* of the current data *name*

**f2sum <firstrow> <lastrow> <procno> n** stores the specified sum under the specified *procno* of the current data *name* but does not change the display to this *procno*

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *2ir*, *2ri*, *2ii* - processed data

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i*- 1D spectrum containing the sum

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

F2SUM(firstrow, lastrow, procno)

F1SUM(firstcol, lastcol, procno)

For both macros counts that if *procno* = -1, the sum is written to the dataset ~TEMP

### SEE ALSO

*f2disco, f1disco* [▷ 102], *f2projn, f2projp, f1projn, f1projp* [▷ 104], *rhpp, rhnp, rvpp, rvnp* [▷ 111]

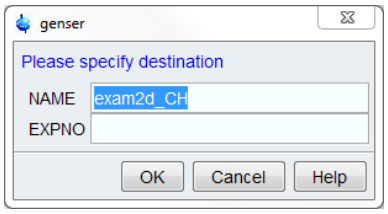
## 4.8 genser

### NAME

genser - Generate pseudo-raw data (2D)

### DESCRIPTION

The command **genser** generates pseudo-raw data from processed 2D data. When entered without arguments, **genser** opens the following dialog box:




Here, you specify the output dataset and click **OK** to actually execute the command. **genser** is normally used in combination with **xif2** and **xif1**. These commands perform an inverse Fourier transform, converting processed frequency domain data into processed time domain data. **genser** converts these processed time domain data into pseudo-raw time domain data and stores them under a new name or experiment number (*expno*).

Note that genser does not modify the data, but only stores them in a different format. The number of data points of the pseudo-raw data, is twice the size (SI) of the processed data they are created from. The acquisition status parameter TD (type dpa) is set accordingly; TD = 2*SI. This counts for both the F2 and F1 direction.

**genser** takes three arguments and can be used as follows:

- **genser** opens a dialog box where you can specify the output data.
- **genser <expno>** stores the output under the specified *expno* and opens a new data window displaying this expno.
- **genser <expno> n** stores the output under the specified *expno*, but does not open and display this expno.

If the specified expno already exists, you will be prompted to overwrite it or not. You can force the overwrite by specifying the extra argument **y** on the command line:

- **genser <expno> y n** stores the output under the specified *expno*, overwriting it if it exists, but does not open and display this expno.

The processed data number (*procno*) of the new dataset is always set to 1.

**genser** can be useful if you want to reprocess a 2D spectrum, for example with different processing parameters, but the raw data do not exist any longer. An example of such a procedure is:

**xif2** (if the data are Fourier transformed in F2)

**xif1** (if the data are Fourier transformed in F1)

**genser** (to create the pseudo-raw data)

**edp** (to set the processing parameters)

**xfb** (to process the pseudo-raw data)

If the input data are processed but not Fourier transformed, you can skip the first two steps.

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *2ir*, *2ri*, *2ii* - processed time domain data

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*ser* - pseudo-raw time domain data

*audita.txt* - acquisition audit trail

### USAGE IN AU PROGRAMS

GENSER(expno)

### SEE ALSO

*genfid* [▷ 64], *xif2, xif1* [▷ 153]

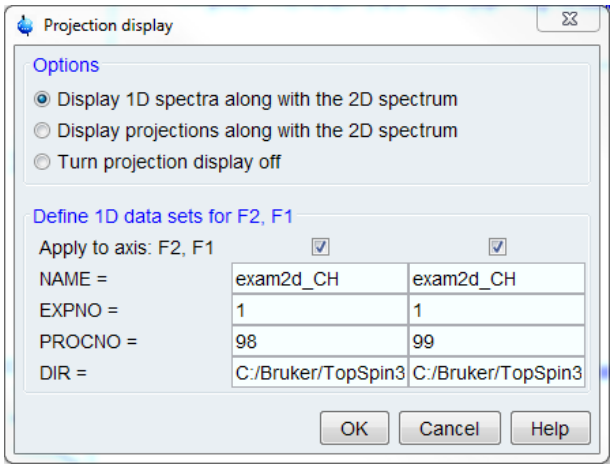
## 4.9 projd

### NAME

projd - Display projections along with the 2D spectrum (2D)

### DESCRIPTION

The **projd** command opens a dialog box where you can specify the projections to be displayed along with the 2D spectrum:



This dialog box offers the following three options:

- *Display 1D spectra along with the 2D spectrum*
- Displays the specified 1D dataset(s) as external projections
- *Display projections along with the 2D spectrum*
- Displays the internal projections.
- *Turn projection display off*
- Turns off the projection display.

In the lower part of the dialog you can specify the 1D datasets to be used for the first option. The checkboxes allow you to display the F2-projection, F1-projection or both. Clicking **OK** will show the projections according to the chosen option and close the dialog.

Note that the effect of the second and third option can also be reached by clicking the button of the toolbar or entering .**pr** on the command line.

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r* - 1D processed data (input for 1st option)

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*curdat2* - definition of the second and third dataset

### SEE ALSO

*f2projn, f2projp* [▷ 104], *rhpp, rhnp* [▷ 111]

## 4.10 rev2, rev1

### NAME

rev2 - Reverse spectrum in F2 (2D)

rev1 - Reverse spectrum in F1 (2D)

### DESCRIPTION

The command **rev2** reverses the spectrum in the F2 direction. This means, each row is mirrored about the central column.

The command **rev1** reverses the spectrum in the F1 direction. This means, each column is mirrored about the central row.

Note that the spectrum can also be reversed by during **xfb** by setting the F2 and/or F1 processing parameter REVERSE to TRUE.

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *2ir*, *2ri*, *2ii* - processed data

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *2ir*, *2ri*, *2ii* - processed data

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

REV2

REV1

**SEE ALSO**

*rv* [▶ 83]

## 4.11 rhpp, rhnp, rvpp, rvnp

**NAME**

rhpp - Calculate horizontal (F2) positive projection (2D)

rhnp - Calculate horizontal (F2) negative projection (2D)

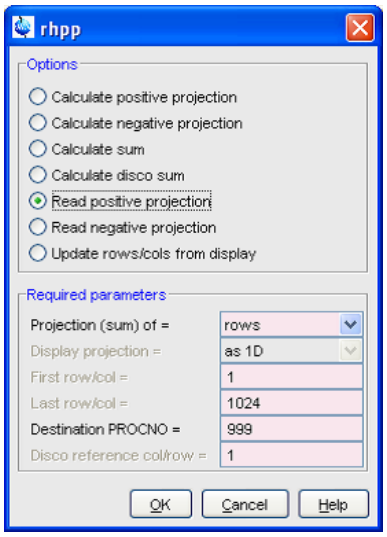
rvpp - Calculate vertical (F1) positive projection (2D)

rvnp - Calculate vertical (F1) negative projection (2D)

proj - Open the projections dialog box (2D,3D)

**DESCRIPTION**

The projection commands can be started from the command line or from the projection dialog box selecting the corresponding command.




This dialog box has several options, each of which selects a certain command for execution.

**Read positive projection (on rows)**

This option selects the command **rhpp** for execution. It calculates the full positive projection of a 2D spectrum in the F2 direction and stores it as a 1D dataset.

**Read positive projection (on columns)**

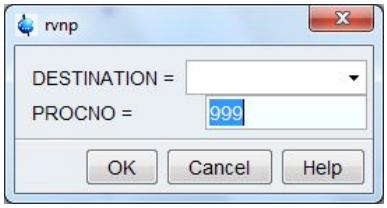
This option selects the command **rvpp** for execution. It calculates the full positive projection of a 2D spectrum in the F1 direction and stores it as a 1D dataset.

### Read negative projection (on rows)

This option selects the command **rhnp** for execution. It calculates the full negative projection of a 2D spectrum in the F2 direction and stores it as a 1D dataset.

### Read negative projection (on columns)

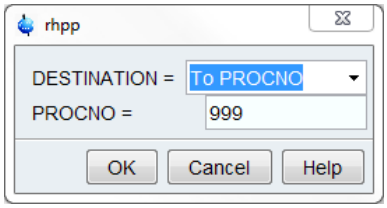
This option selects the command **rvnp** for execution. It calculates the full negative projection of a 2D spectrum in the F1 direction and stores it as a 1D dataset.




A projection is a 1D trace where every point has the highest intensity of all points of the corresponding orthogonal trace in the 2D spectrum.

**r*p** commands only take the projection of the first quadrant data (file *2rr*) and store it as real 1D data (file *1r*)

**r*p** commands can be started from the command line. When entered without arguments, a dialog window is displayed:




The required arguments can also be specified on the command line.

**rhpp <procno>** stores the projection under the specified *procno* of the current data *name*

**rhpp <procno> n** stores the projection under the specified *procno* but does not change the display to that *procno*

The three other **r*p** command have the same syntax.

A special case is the command **rvpp** or **rvnp** on a hypercomplex 2D dataset (MC2 ≠ QF) that has been processed in F2 only. Suppose you would perform the following command sequence:

**xf2** - to process the data in F2 only

**s si** - to check the F1 size of the 2D data, click **Cancel**.

**s mc2** - to check status MC2 (≠ QF), click **Cancel**.

**rvpp** - to store the F1 projection in ~TEMP and change to that dataset

**s si** - to check the size of the resulting 1D dataset, click **Cancel** .

You will see that the size of the 1D data is only half the F1 size of the 2D data. The reason is that **rvpp** unshuffles the input data (file *2rr*). As such, **rvpp** behaves like the command **rsc**. If you want to prevent the unshuffling of the input data (file *2rr*), you can use the following trick. Set the status parameter MC2 to QF before you run **rvpp** :

**s mc2** , click **QF**

Then, the size of the 1D data will be the same as the F1 size of the 2D data.

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r* - 1D spectrum containing the projection

*auditp.txt* - processing audit trail

If the commands are used without arguments, the files are stored in:

*<dir>/data/<user>/nmr/~TEMP/1/pdata/1/*

### USAGE IN AU PROGRAMS

RHPP(procno)

RHNP(procno)

RVPP(procno)

RVNP(procno)

For all these macros counts that if *procno* = -1, the projection is written to the dataset ~TEMP

### SEE ALSO

*f2projn, f2projp* [▷ 104], *f2sum, f1sum* [▷ 106], *f2disco, f1disco* [▷ 102]

## 4.12 rsc

### NAME

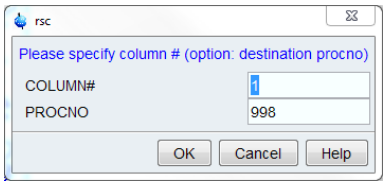
rsc - Read column from 2D data and store as 1D data

### SYNTAX

rsc [<column> [<procno>] [n]]

### DESCRIPTION

The command **rsc** reads a column from a 2D spectrum and stores it as a 1D spectrum. When entered on a 2D dataset without arguments, **rsc** opens a dialog box where you can specify the column number and the *procno* of the output data.




The column must be specified as a number between 1 and F2-SI. The latter is the F2 processing status parameter SI that can be viewed with **s si**. The *procno* can be any number other that the current *procno*. If the *procno* field is left empty, the output dataset is stored under data name ~TEMP.

When entered on a 2D dataset, **rsc** takes up to three arguments and can be used as follows:

**rsc** opens the above dialog box

**rsc <column>** stores the specified column under data *name* ~TEMP

**rsc <column> <procno>** stores the specified column under the current data *name*, the current *expno* and the specified *procno*. It changes the display to the output 1D data.

**rsc <column> <procno> n** stores the specified column under the current data *name*, the current *expno* and the specified *procno*. It does not change the display to the output 1D data.

After **rsc** has read a column and the display has changed to the destination 1D dataset, a subsequent **rsc** command can be entered on this 1D dataset. This takes two arguments and can be used as follows:

**rsc** opens the above dialog box

**rsc <column>** reads the specified column from the 2D dataset from which the current 1D dataset was extracted

**rsc <column> <procno>** reads the specified column from the 2D dataset that resides under the current data *name* (however, if the current data name is ~TEMP, rsc <column> <procno> reads from the specified *procno* in the dataset from which the current 1D dataset was extracted), the current *expno* and the specified *procno*. Specifying the *procno* allows to read a column from a 2D dataset other than the one from which the current 1D dataset was extracted. Furthermore, the AU macro RSC requires two arguments, no matter if it is used on a 1D or on a 2D dataset.

**rsr** can also be started from the dialog box that is opened with the command **slice**.

A special case is a 2D dataset that has been Fourier transformed in F2 but not in F1. **rsc** then stores 1D processed data that are in the time domain rather than the frequency domain. Below are five different examples of this case.

**Example 1**

A 2D dataset is Fourier transformed in F2, column 17 (time domain) is extracted and stored under the same name and *expno*, in *procno* 2. The resulting 1D dataset is Fourier transformed.

On the 2D dataset, enter the following commands:

**xf2** - to Fourier transform in F2 only

**rsc 17 2** - to read column 17 to *procno* 2 and switch to that dataset

**ft** - to Fourier transform the resulting 1D data according to FnMODE

The 1D data shares the expno, and the acquisition parameters in it, with the source 2D dataset. 1D processing commands automatically recognize that this 1D dataset is a column from a 2D dataset. The command **ft** interprets the F1 acquisition parameter FnMODE to determine the Fourier transform mode.

**Example 2**

A 2D dataset with F1 acquisition mode *States* is Fourier transformed in F2. Column 17 (time domain) is extracted and stored under data *name* ~TEMP. The resulting 1D dataset is Fourier transformed.

On the 2D dataset, enter the following commands:

**s fnmode** – to check the FnMODE value (*States*), click **Cancel**.

**xf2** - to Fourier transform in F2 only.

**s mc2** – to check the MC2 value (*States*), click **Cancel**.

**rsc 17** - read column 17 to ~TEMP and switch to that dataset.

**s aq_mod** – to check the AQ_mod value (*qsim*), click **Cancel**.

**ft** - Fourier transform the resulting 1D data according to AQ_mod.

The source 2D and the destination 1D have a separate a set of acquisition parameters. rsc reads the F1 status parameter MC2 of the 2D data and translates that to the corresponding AQ_mod of the 1D data. 1D processing commands recognizes this 1D dataset as regular 1D data. This means, for example, that **ft** interprets the AQ_mod to determine the Fourier transform mode.

**Example 3**

A 2D dataset with an F1 acquisition mode States-TPPI is Fourier transformed in F2. Column 17 (time domain) is extracted and stored under data name ~TEMP. The resulting 1D dataset is Fourier transformed.

On the 2D dataset, enter the following commands:

**s fnmode** – to check the FnMODE value (*States-TPPI*), click **Cancel**.

**xf2** - to Fourier transform in F2 only

**s mc2** - to check the MC2 value (*States-TPPI*), click **Cancel**.

**rsc 17** - to read column 17 to ~TEMP and switch to that dataset

**ft_mod** - to check the FT_mod value (*fsc*), click **Cancel**.

**trfp** - to Fourier transform the resulting 1D data according to FT_mod

The source 2D and the destination 1D have a separate a set of acquisition parameters. Since there is no value for AQ_mod that corresponds to States-TPPI, rsc sets the processing parameter FT_mod instead of the acquisition status parameter AQ_mod. As such, the resulting 1D dataset can only be Fourier transformed correctly with trfp.

**Example 4**

A 2D dataset with an F1 acquisition mode QF is Fourier transformed in F2. Column 17 (time domain) is extracted and stored under data name ~TEMP. From the 2D dataset, enter the following commands:

**s fnmode** – to check the FnMODE value (*QF*), click **Cancel**.

**xf2** - to Fourier transform in F2 only

**s mc2** – to check the MC2 value (*QF*), click **Cancel**.

**rsc 17** - to read column 17 to ~TEMP and switch to that dataset.

**s si** – to check the size of the 1D dataset, click **Cancel**.

For FnMODE = QF the 2D storage mode is different than for other values (see the description of xfb). As such, the size of the resulting 1D data is twice as large as for other values of FnMODE. If 2D imaginary data (file 2ii) exist, 1D imaginary (file 1i) are created. Only in that case, the 1D data can be Fourier transformed.

**Example 5**

From a 3D dataset, a plane is extracted and, from this plane a column is extracted.

On the 3D dataset, enter the following commands:

**xf2 s13 48 2** - to read the F3-F1 plane 48 to *procno* 2

**rsc 19 3** - to read, from plane 48, column 19 to *procno* 3

**ft** : to Fourier transform the resulting 1D data according to FnMODE

The 3D, 2D and 1D dataset are stored in three different procno's all under the same expno, i.e. they share the same acquisition parameters. 1D processing commands automatically recognize that the 1D dataset is a column from an F3-F1 plane that was extracted from a 3D dataset. As such, **ft** interprets the F1 parameter FnMODE to determine the Fourier transform mode. Note that F1 is the third direction of the 3D dataset. The parameter handling, however, is transparent to the user.

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *2ir*, *2ri*, *2ii* - 2D processed data

### OUTPUT FILES

If no output *procno* is specified:

*<dir>/data/<user>/nmr/~TEMP/1/pdata/1/*

*1r*, *1i* - 1D spectrum

*used_from* - data path of the source 2D data and the column no.

*auditp.txt* - processing audit trail

If the output *procno* is specified:

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - 1D spectrum

*used_from* - data path of the source 2D data and the column no.

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

RSC(column, procno)

If *procno* = -1, the column is written to the dataset ~TEMP

### SEE ALSO

*rsr* [▸ 116], *rtr* [▸ 202], *wsr* [▸ 130], *wsc* [▸ 126], *rser2d* [▸ 171], *wser* [▸ 127], *wserp* [▸ 128], *r12, r13* [▸ 167]

## 4.13 rsr

### NAME

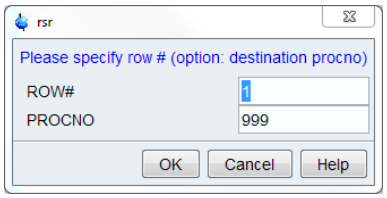
rsr - Read row from 2D data and store as 1D data (2D,1D)

### SYNTAX

rsr [<row> [<procno>] [n]]

### DESCRIPTION

The command **rsr** reads a row from a 2D spectrum and stores it as a 1D spectrum. When entered on a 2D dataset without arguments, **rsr** opens a dialog box where you can specify the row number and the *procno* of the output data.




The row must be specified as a number between 1 and F1-SI. The latter is the F1 processing status parameter SI that can be viewed with **s si**. The *procno* can be any number other that the current *procno*. If the *procno* field is left empty, the output dataset is stored under data name ~TEMP.

When entered on a 2D dataset, **rsr** takes up to three arguments and can be used as follows:

- **rsr <row>** stores the specified row under data *name* ~TEMP
- **rsr <row> <procno>** stores the specified row under the current data *name*, the current *expno* and the specified *procno*. It changes the display to the output 1D data.
- **rsr <row> <procno> n** stores the specified row under the current data *name*, the current *expno* and the specified *procno*. It does not change the display to the output 1D data.

After **rsr** has read a row and the display has changed to the destination 1D dataset, a subsequent **rsr** command can be entered on this 1D dataset. This takes two arguments and can be used as follows:

- **rsr** opens the dialog box where you can specify the row and *procno* of the 2D data
- **rsr <row>** reads the specified row from the 2D dataset from which the current 1D dataset was extracted
- **rsr <row> <procno>** reads the specified row from the 2D dataset that resides under the current data *name* (however, if the current data name is ~TEMP, rsr <row> <procno> reads from the specified *procno* in the dataset from which the current 1D dataset was extracted), the current *expno* and the specified *procno*. Specifying the *procno* allows to read a row from a 2D dataset other than the one from which the current 1D dataset was extracted. Furthermore, the AU macro RSR requires two arguments, no matter if it is used on a 1D or on a 2D dataset.

**rsr** can also be started from the dialog box that is opened with the command **slice**.

## INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *2ir*, *2ri*, *2ii* - 2D processed data

## OUTPUT FILES

If no *procno* is specified:

*<dir>/data/<user>/nmr/~TEMP/1/pdata/1/*

*1r*, *1i* - 1D spectrum

*used_from* - data path of the source 2D data and the row no.

*auditp.txt* - processing audit trail

If the output *procno* is specified:

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - 1D spectrum

*used_from* - data path of the source 2D data and the row no.

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

RSR(row, procno)

If *procno* = -1, the row is written to the dataset ~TEMP

### SEE ALSO

*r12, r13, r23, slice* [ 167], *rsc* [ 113], *rser2d* [ 171], *rtr* [ 202], *wsc* [ 126], *wser* [ 127], *wserp* [ 128], *wsr* [ 130]

## 4.14 rser

### NAME

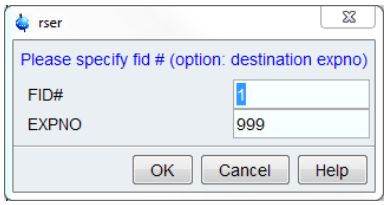
rser - Read row from 2D raw data and store as 1D FID (2D,1D)

### SYNTAX

rser [<row> [<expno>] [n]]

### DESCRIPTION

The command **rser** reads a row from 2D or 3D raw data (a series of FIDs) and stores it as a 1D dataset. It opens a dialog box where you can specify the FID number and the *expno* of the output data.




For 2D data, the row must be specified as a number between 1 and F1-TD. The latter is the F1 acquisition status parameter TD that can be viewed with **s td**.

**rser** is normally entered on the 2D dataset. It then takes up to three arguments and can be used as follows:

**rser** prompts for the row number and stores it under data *name* ~TEMP

**rser <row>** stores the specified row under data *name* ~TEMP

**rser <row> <expno>** stores the specified row under the current data *name* and the specified *expno* and then changes the display to this *expno*

**rser <row> <expno> n** stores the specified row under the current data *name* and the specified *expno* but does not change the display to this *expno*

**rser <row> <expno> eao** performes EA calculation in all dimensions with acquisition status parameter FnMODE = Echo-Antiecho and stores the specified row under the current data name and the specified expno.

After **rser** has read a row and the display has changed to the destination 1D dataset, a subsequent **rser** command can be entered on this 1D dataset. This takes two arguments and can be used as follows:

**rser** opens the above dialog box where you can specify the row number and the *procno* of the 2D dataset from which the current 1D dataset was extracted

**rser <row>** reads the specified row from the 2D dataset from which the current 1D dataset was extracted

**rser <row> <expno>** reads the specified row from the 2D dataset that resides under the current data *name* (however, if the current data name is ~TEMP, the input dataset is the one from which the current 1D dataset was extracted, except for the specified *expno (procno)*, the specified *expno* and *procno* 1.

Note that on 3D data, rser does not distinguish between the F2 and F1 direction and treats the 3D dataset as a large 2D dataset. This implies that the row number must lie between 1 and (F2-TD) * (F1-TD).

**rser** can also be started from the dialog box that is opened with the command **slice**.

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*ser* - 2D or 3D raw data

**OUTPUT FILES**

If the output *expno* is specified:

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - 1D FID

*audita.txt* - acquisition audit trail

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/1/*

*used_from* - data path of the source 2D data and the row no.

If no output *expno* is specified:

*<dir>/data/<user>/nmr/~TEMP/1/*

*fid* - 1D FID

*<dir>/data/<user>/nmr/~TEMP/1/pdata/1*

*used_from* - data path of the source 2D data and the row no.

**USAGE IN AU PROGRAMS**

RSER(row, expno, procno)

If expno = -1, the row is written to the dataset ~TEMP

**SEE ALSO**

*r12, r13, r23, slice [* 167], *rsc [* 113], *rser2d [* 171], *rsr [* 116], *wsc [* 126], *wser [* 127], *wserp [* 128], *wsr [* 130]

## 4.15 sub2, sub1, sub1d2, sub1d1

**NAME**

sub2 - Subtract 1D data from 2D data rows, keep sign (2D)

sub1 - Subtract 1D data from 2D data columns, keep sign (2D)

sub1d2 - Subtract 1D data from 2D data rows (2D)

sub1d1 - Subtract 1D data from 2D data columns (2D)

adsu - Open add/subtract/multiply dialog box (1D, 2D)

## DESCRIPTION

Subtracting a 1D data from a 2D data can be started from the command line or from the add/subtract dialog box. The latter is opened with the command **adsu**.

This dialog box offers several options, each of which selects a certain command for execution.

### Subtract a 1D spectrum from each row, retain sign

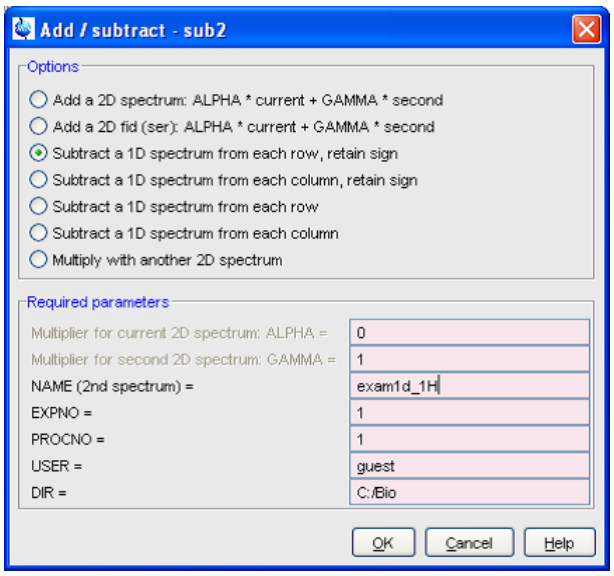
This option selects the command **sub2** for execution. It subtracts a 1D dataset from each row of the current 2D spectrum. It first compares the intensity of each data point of the 1D spectrum with the intensity of the corresponding data point in the 2D spectrum. If they have opposite signs, no subtraction is done and the 2D data point remains unchanged. If they have the same sign and the 1D data point is smaller than the 2D data point, the subtraction is done. If the 1D data point is greater than the 2D data point, the latter is set to zero. As such, the sign of the 2D data points always remains the same.

### Subtract a 1D spectrum from each column, retain sign

This option selects the command **sub1** for execution. It works like **sub2**, except that it subtracts the 1D second dataset from each column of the current 2D spectrum.

### Subtract a 1D spectrum from each row

This option selects the command **sub1d2** for execution. It subtracts a 1D dataset from each row of the current 2D spectrum. Unlike **sub2**, it does not compare intensities.




### Subtract a 1D spectrum from each column

This option selects the command **sub1d1** for execution. It subtracts a 1D dataset from each column of the current 2D spectrum. Unlike **sub1**, it does not compare intensities.

The **sub*** commands only work on the real data. After using them, the imaginary data no longer match the real data and cannot be used for phase correction.

If the second dataset has not been defined yet, the **sub*** commands open the add/subtract (**adsu**) dialog box.

The **adsu** command can be used on 1D or 2D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

*<dir2>/data/<user2>/nmr/<name2>/<expno2>/pdata/<procno2>/*

*1r* - real processed 1D data

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

SUB2

SUB1

SUB1D2

SUB1D1

**SEE ALSO**

*add2d, mul2d, addser [▷ 99]*

## 4.16 sym, syma, symj, symt

**NAME**

sym - Symmetrize spectrum about the diagonal (2D)

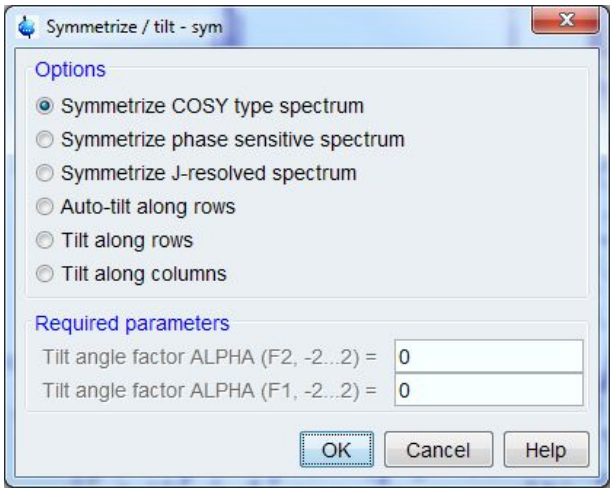
syma - Symmetrize spectrum about the diagonal, keep sign (2D)

symj - Symmetrize spectrum about central horizontal line (2D)

symt - Open symmetrization and tilt dialog box (2D)

**DESCRIPTION**

All **sym*** commands open the symmetrize/tilt dialog box:



This dialog box offers several options, each of which selects a certain command for execution.

### Symmetrize COSY type spectrum

This option selects the command **sym** for execution. It symmetrizes a 2D spectrum about a diagonal from the lower left corner (data point 1,1) to the upper right corner (data point F2-SI, F1-SI). It compares each data point with the corresponding data point on the other side of the diagonal and determines which one has the lowest (most negative) intensity. Then both data points are set to that intensity.The following table shows the intensities of four pairs of data points before and after **sym**:

| before **sym** | after **sym** |
|---|---|
| -370000, 12000 | -370000, -370000 |
| 1000, -700 | -700, -700 |
| 18000, 6000 | 6000, 6000 |
| -13000, -8000 | -13000, -13000 |

**sym** is typically used on magnitude cosy spectra.

### Symmetrize phase sensitive spectrum

This option selects the command **syma** for execution. It works like **sym**, except that it compares each data point with the corresponding data point on the other side of the diagonal and determines which one has the lowest absolute intensity. Then both data points are set to that intensity while each point keeps its original sign. The following table shows the intensities of four pairs of data points before and after **syma**:

| before **syma** | after **syma** |
|---|---|
| -370000, 12000 | -12000, 12000 |
| 1000, -700 | 700, -700 |
| 18000, 6000 | 6000, 6000 |
| -13000, -8000 | -8000, -8000 |

**syma** is typically used on phase sensitive cosy spectra.

### Symmetrize J-resolved spectrum

This option selects the command **symj** for execution. It symmetrizes a 2D spectrum about a horizontal line through the middle. It is similar to **sym**, i.e. it compares each data point with the corresponding data point on the other side of the horizontal line and determines which one has the lowest (most negative) intensity. Then both data points are set to that intensity. The following table shows the intensities of 5 pairs of data points before and after **symj**:

| before **symj** | after **symj** |
|---|---|
| -370000, 12000 | -370000, -370000 |
| 1000, -700 | -700, -700 |
| 18000, 6000 | 6000, 6000 |
| -13000, -8000 | -13000, -13000 |

**symj** is typically used on J-resolved spectra which have been tilted with the command **tilt**.

**sym*** commands only work on the real data. After using it, the imaginary data no longer match the real data and cannot be used for phase correction.

When executed from the command line, the command **sym**, **syma** and **symj** select the corresponding option in the dialog box. This means, you can just click **OK** or hit **Enter** to start the command. In contrast, **symt** selects the last used symmetrization command.

**OUTPUT PARAMETERS**

Can be viewed with **dpp** or by typing **s symm** :

SYMM - type of symmetrization (*no*, *sym*, *syma* or *symj*) done

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

SYM

SYMA

SYMJ

**SEE ALSO**

*tilt, ptilt, ptilt1 [▷ 123]*

## 4.17 tilt, ptilt, ptilt1

**NAME**

tilt - Tilt a 2D spectrum

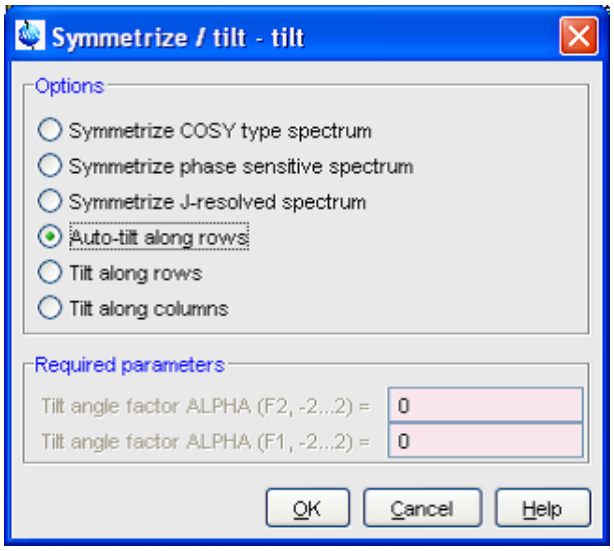
ptilt - Tilt a 2D spectrum by shifting the data in the F2 direction

ptilt1 - Tilt a 2D spectrum by shifting the data in the F1 direction

symt - Open the symmetrize/tilt dialog box

**DESCRIPTION**

All ***tilt*** commands open the symmetrize/tilt dialog box.




This dialog box offers several options, each of which selects a certain command for execution.

**Auto-tilt along rows**

This option selects the command **tilt** for execution. It tilts the 2D spectrum, shifting each row of the 2D spectrum by the value:

n = tiltfactor * (nsrow/2 - row)

The variables in this equation are defined as:

tiltfactor = (SW_p1/SI1) / (SW_p2/SI2)

nsrow = total number of rows

row = the row number

Where SW_p1, SI1, SW_p2 and SI2 represent the processing status parameters SW_p and SI in F1 and F2, respectively.

The upper half of the spectrum is shifted to the right, the lower half to the left. Furthermore, this is a circular shift, i.e. the data points which are cut off at the right edge of the spectrum are appended at the left edge and vice versa.

**Tilt along rows**

This option selects the command **ptilt** for execution. It tilts the 2D spectrum about a user defined angle, by shifting the data points in the F2 direction. It is typically used to correct possible magnet field drifts during long term 2D experiments. The tilt factor is determined by the F2 processing parameter ALPHA which can take a value between -2 and 2. Each row of the 2D matrix is shifted by *n* points where *n* is defined by:

n = tiltfactor * (nsrow/2 - row)

The variables in this equation are defined by:

tiltfactor = ALPHA*SI2 / SI1

nsrow = total number of rows

row = the row number

Where SI2 and SI1 are processing status parameter SI in F2 and F1, respectively.

### Tilt along columns

This option selects the command **ptilt1** for execution. It tilts the 2D spectrum about a user defined angle, by shifting the data points in the F1 direction. The tilt factor is determined by the F1 processing parameter ALPHA which can take a value between -2 and 2. Each column of the 2D matrix is shifted by *n* points where *n* is defined by:

n = tiltfactor * (nscol/2 - col)

The variables in this equation are defined by:

tiltfactor = ALPHA*SI1/ SI2

nscol = total number of columns

col = the column number

Where SI2 and SI1 are processing status parameter SI in F2 and F1, respectively.

For F2-ALPHA = 1 and F1-ALPHA = 1:

- the sequence **ptilt** - **ptilt1** rotates the spectrum by 90°
- the sequence **ptilt1** - **ptilt** rotates the spectrum by -90°.

When executed from the command line, the command **tilt**, **ptilt** and **ptilt1** select the corresponding option in the dialog box. This means, you can just click **OK** or hit **Enter** to start the command. In contrast, **symt** selects the last used tilt command.

### INPUT PARAMETERS

Set from the **symt** dialog box, with **edp** or by typing **alpha**:

ALPHA - tilt factor (used by **ptilt** and **ptilt1**)

Set by initial processing command, e.g. **xfb**, can be viewed with **dpp**:

SW_p - spectral width of the processed data (used by **tilt**)

SI - size of the processed data

### OUTPUT PARAMETERS

Can be viewed with **dpp**:

TILT - shows whether **tilt**, **ptilt** or **ptilt1** was done (true or false)

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

TILT

PTILT

PTILT1

### SEE ALSO

*sym, syma, symj, symt [▷ 121]*

## 4.18 wsc

### NAME

wsc - Replace column of 2D spectrum by 1D spectrum

### SYNTAX

wsc [<row> [<procno> ]]

### DESCRIPTION

The command **wsc** replaces one column of 2D processed data by 1D processed data. It is normally used in combination with **rsc** in the following way:

1. Run **rsc** to extract column *x* from a 2D spectrum
2. Manipulate the resulting 1D data with 1D processing commands
3. Run **wsc** to replace column *x* of the 2D data with the manipulated 1D data

**wsc** can be entered on the source 1D dataset or on the destination 2D dataset.

Examples of the usage of **wsc** on the source 1D dataset:

- **wsc** prompts for the column of the destination 2D data which must be replaced by the current 1D data. The 2D dataset is the one from which the 1D dataset was extracted.
- **wsc <column>** the specified column of the destination 2D data is replaced by the current 1D data. The 2D dataset is the one from which the current 1D dataset was extracted.
- **wsc <column> <procno>** the specified column of the destination 2D data is replaced by the current 1D data. The 2D dataset must reside under the current data *name* (however, if the current data name is ~TEMP, wsc <column> <procno> writes to the specified *procno* in the dataset from which the current 1D dataset was extracted), the current *expno* and the specified *procno*.

Examples of usage of **wsc** on the destination 2D dataset:

- **wsc <column>** the specified column of the current 2D processed data is replaced. The source 1D data must reside under the data *name* ~TEMP
- **wsc <column> <procno>** the specified column of the current 2D processed data is replaced. The source 1D data must reside under the current data *name*, the current *expno* and the specified *procno*.

Although **wsc** is normally used as described above, it allows to specify a full dataset path in the following way:

**wsc <column> <procno> <expno> <name> <user> <dir>**

When entered on a 1D dataset, the arguments specify the destination 2D dataset. When entered on a 2D dataset, the arguments specify the source 1D dataset. If only certain parts of the destination 2D data path are specified, e.g. the *expno* and *name*, the remaining parts are the same as in the current 1D data path. In AU programs, **wsc** must always have 6 arguments (see USAGE IN AU PROGAMS below).

**wsc** can also be started from the dialog box that is opened with the command **slice**.

### INPUT FILES

*<dir>/data/<user>/nmr/~TEMP/1/pdata/1*

*1r*, *1i* - 1D processed data

*used_from* - data path of the 2D data (input of **wsc** on a 1D dataset)

or

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*1r*, *1i* - 1D processed data

*used_from* - data path of the 2D data (input of **wsc** on a 1D dataset)

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*2rr*, *2ri* - processed 2D data

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

WSC(column, procno, expno, name, user, dir)

**SEE ALSO**

*r12, r13, r23, slice* [ 167], *rsc* [ 113], *rser2d* [ 171], *rsr* [ 116], *wser* [ 127], *wserp* [ 128], *wsr* [ 130]

## 4.19 wser

**NAME**

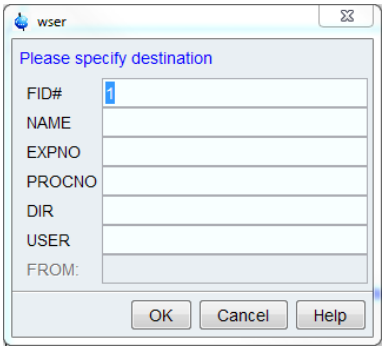
wser - Replace row of 2D raw data by 1D raw data (2D)

**SYNTAX**

wser [<row> [<expno> ]]

**DESCRIPTION**

The command **wser** replaces one row of 2D raw data by 1D raw data. It can be entered on the source 1D dataset or on the destination 2D dataset. When entered on a 1D dataset, **wser** opens the following dialog box:




Enter the FID number to be replaced and the destination data path.

Usage of **wser** with arguments on the source 1D dataset:

- **wser <row>** the specified row of the 2D raw data is replaced by the current 1D FID. The destination 2D dataset is the one from which the current 1D dataset was extracted.
- **wser <row> <expno>** the specified row of the 2D raw data is replaced by the current 1D FID. The 2D dataset must reside under the current data *name*, the specified *expno* and *procno* 1.

Usage of **wser** with arguments on the destination 2D dataset:

- **wser <row> <expno>** the specified row of the current 2D raw data is replaced. The source 1D dataset must reside under the current data *name*, specified *expno* and *procno* 1.

### INPUT FILES

*<dir>/data/<user>/nmr/~TEMP/1/*

*fid* - 1D raw data

*<dir>/data/<user>/nmr/~TEMP/1/pdata/1*

*used_from* - data path of the 2D data (input of **wser** on a 1D dataset)

or

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - 1D raw data

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*used_from* - data path of the 2D data (input of **wser** on a 1D dataset)

**wser** can also be started from the dialog box that is opened with the command **slice**.

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*ser* - raw 2D data

*audita.txt* - acquisition audit trail

### USAGE IN AU PROGRAMS

WSER(row, name, expno, procno, dir, user)

Note that the order of the arguments in AU programs is different from the order on the command line.

### SEE ALSO

*r12, r13, r23, slice* [▷ 167], *rsc* [▷ 113], *rser2d* [▷ 171], *rsr* [▷ 116], *wsc* [▷ 126], *wserp* [▷ 128], *wsr* [▷ 130]

## 4.20 wserp

### NAME

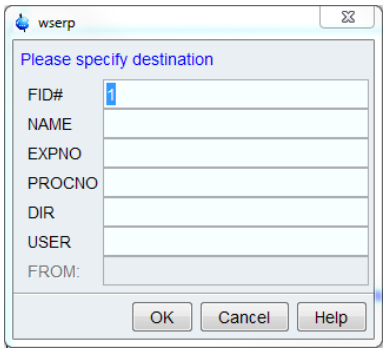
wserp - Replace row of 2D raw data by 1D processed data

### SYNTAX

wserp [<row> [<expno> ]]

### DESCRIPTION

The command **wserp** replaces one row of 2D raw data by processed 1D data. It can be entered on the source 1D dataset or on the destination 2D dataset. When entered on a 1D dataset, **wserp** opens the following dialog box:




Here, you can enter the FID number to be replaced and the destination data path.

Usage of **wserp** with arguments on the source 1D dataset:

- **wserp <row>** the specified row of the 2D raw data is replaced by the current 1D processed data. The 2D dataset is the one from which the current 1D dataset was extracted.
- **wserp <row> <expno>** the specified row of the 2D raw data under the specified *expno* is replaced by the current 1D processed data. The 2D dataset *name*, *user* and *dir* are the same as in the dataset as the current 1D data were extracted from.

Usage of **wserp** with arguments on the destination 2D dataset:

- **wserp <row> <expno>** the specified row of the current 2D raw data is replaced. The source 1D dataset must reside under the current data *name*, specified *expno* and *procno* 1.

**wserp** can also be started from the dialog box that is opened with the command **slice**.

## INPUT FILES

*<dir>/data/<user>/nmr/~TEMP/1/pdata/1/*

*1r*, *1i* - 1D processed data (real, imaginary)

*used_from* - data path of the 2D data (input of **wserp** on a 1D dataset)

or

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - 1D processed data (real, imaginary)

*used_from* - data path of the 2D data (input of **wserp** on a 1D dataset)

## OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*ser* - raw 2D data

*audita.txt* - acquisition audit trail

## USAGE IN AU PROGRAMS

WSERP(row, name, expno, procno, dir, user)

Note that the order of the arguments in AU programs is different from the order on the command line.

## SEE ALSO

*r12, r13, r23, slice* [▸ 167], *rsc* [▸ 113], *rser2d* [▸ 171], *rsr* [▸ 116], *wsc* [▸ 126], *wser* [▸ 127], *wsr* [▸ 130]

## 4.21 wsr

### NAME

wsr - Replace row of a 2D spectrum by 1D spectrum

### SYNTAX

wsr [<row> [<procno> ]]

### DESCRIPTION

The command **wsr** replaces one row of 2D processed data by 1D processed data. It is normally used in combination with **rsr** in the following way:

- run **rsr** to extract row *x* from a 2D spectrum
- manipulate the resulting 1D data with 1D processing commands
- run **wsr** to replace row *x* of the 2D data with the manipulated 1D data

**wsr** can be entered on the source 1D dataset or on the destination 2D dataset.

Examples of the usage of **wsr** on the source 1D dataset:

- **wsr** prompts for the row of the destination 2D data which must be replaced by the current 1D data. The 2D dataset is the one from which the current 1D dataset was extracted.
- **wsr <row>** the specified row of the destination 2D data is replaced by the current 1D data. The 2D dataset is the one from which the current 1D dataset was extracted.
- **wsr <row> <procno>** the specified row of the destination 2D data is replaced by the current 1D data. The 2D dataset must reside under the current data *name* (however, if the current data name is ~TEMP, wsr <row> <procno> writes to the specified procno in the dataset from which the current 1D dataset was extracted), the current *expno* and the specified *procno*.

Examples of usage of **wsr** on the destination 2D dataset:

- **wsr <row>** the specified row of the current 2D processed data is replaced. The source 1D data must reside under the data *name* ~TEMP.
- **wsr <row> <procno>** the specified row of the current 2D processed data is replaced. The source 1D data must reside under the current data *name*, the current *expno* and the specified *procno*.

**wsr** can also be started from the dialog box that is opened with the command **slice**.

### INPUT FILES

*<dir>/data/<user>/nmr/~TEMP/1/pdata/1*

*1r*, *1i* - 1D processed data

*used_from* - data path of the 2D data (input of **wsr** on a 1D dataset)

or

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*1r*, *1i* - 1D processed data

*used_from* - data path of the 2D data (input of **wsr** on a 1D dataset)

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*2rr*, *2ir* - processed 2D data

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

WSR(row, procno, expno, name, user, dir)

### SEE ALSO

*wsc* [▷ 126], *rsr* [▷ 116], *rsc* [▷ 113], *wser* [▷ 127], *wserp* [▷ 128], rser, *rser2d* [▷ 171], *r12, r13* [▷ 167]

## 4.22 xf1

### NAME

xf1 - Process data, including FT, in F1 (2D)

### DESCRIPTION

The command **xf1** processes a 2D dataset in the F1 direction. It can be started from the command line or from the Fourier transform dialog box. The latter is opened with the command **ftf**.

**xf1** Fourier transforms time domain data (FID) into frequency domain data (spectrum). Depending on the F1 processing parameters BC_mod, WDW, ME_mod and PH_mod, **xf1** also performs baseline correction, window multiplication, linear prediction and phase correction, respectively. These steps are described in detail for the command **xfb**.

Normally, 2D data are processed with the command **xfb** which performs a Fourier transform in both directions, F2 and F1. In some cases, however, it is useful to process the data in two separate steps using the sequence **xf2** - **xf1**, for example to view the data after processing them in F2 only.

If you run **xf1** without running **xf2** first, a warning that the F2 transform has not been done will appear. When the command has finished the data are in the time domain in F2 and in the frequency domain in F1. The opposite case, however, is more usual, i.e. data which have only been processed with **xf2**.

**xf1** takes the same options as **xfb**.

The F1 Fourier transform mode and data storage mode depends on the F1 acquisition mode (see INPUT PARAMETERS below and the description of **xfb**).

### INPUT PARAMETERS

**F2 and F1 parameters**

Set by **xf2**, can be viewed with **dpp** or by typing **s si**, **s stsr** etc.:

SI - size of the processed data

STSR - strip start: first output point of strip transform

STSI - strip size: number of output points of strip transform

TDeff - number of raw data points to be used for processing

TDoff - first point of the FID used for processing (default 0)

If **xf2** has not been done, **xf1** uses the **edp** parameters set by the user.

**F1 parameters**

Set from the **ftf** dialog box, with **edp** or by typing **bc_mod** etc.

BC_mod - FID baseline correction mode

BCFW - filter width for BC_mod = sfil or qfil

COROFFS - correction offset for BC_mod = spol/qpol or sfil/qfil

ME_mod - FID linear prediction mode

NCOEF - number of linear prediction coefficients

LPBIN - number of points for linear prediction
TDoff - number of raw data points predicted for ME_mod = LPb*
WDW - FID window multiplication mode
LB - Lorentzian broadening factor for WDW = em or gm
GB - Gaussian broadening factor for WDW = gm, sinc or qsinc
SSB - Sine bell shift for WDW = sine, qsine, sinc or qsinc
TM1, TM2 - limits of the trapezoidal window for WDW = trap
PH_mod - phase correction mode
PHC0 - zero order phase correction value for PH_mod = pk
PHC1 - first order phase correction value for PH_mod = pk
FCOR - first (FID) data point multiplication factor (0.0-2.0, default 0.5)
REVERSE - flag indicating to reverse the spectrum
Set by the **xf2**, can be viewed with **dpp** or by typing **s mc2** :
MC2 - Fourier transform mode (input of **xf1** on processed data)
Set by the acquisition, can be viewed with **dpa** or by typing **s fnmode**:
FnMODE - Acquisition mode (input of **xf1** on raw data)

**OUTPUT PARAMETERS**

**F1 parameters**

Can be viewed with **dpp** or by typing **s ft_mod** etc.:
FT_mod - Fourier transform mode
FTSIZE - Fourier transform size

**F2 parameters**

Can be viewed with **dpp** or by typing **s ymax_p**, **s ymin_p** etc.:
YMAX_p - maximum intensity of the processed data
YMIN_p - minimum intensity of the processed data
S_DEV - standard deviation of the processed data
NC_proc - intensity scaling factor

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*
*ser* - raw data (input if *2rr* does not exist or is Fourier transformed in F1)
*acqu2s* - F1 acquisition status parameters
*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*
*2rr* - real processed data (input if it exists but is not processed in F1)
*2ir* - second quadrant imaginary processed data (input if FnMODE ≠ QF)
*2ii* - second quadrant imaginary processed data (input if FnMODE = QF)
*proc* - F2 processing parameters
*proc2* - F1 processing parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*
*2rr* - real processed data
*2ir* - third quadrant imaginary processed data (output if FnMODE ≠ QF)

*2ii* - fourth quadrant imaginary processed data (output if FnMODE ≠ QF)

*2ii* - second quadrant imaginary processed data (output if FnMODE = QF)

*procs* - F2 processing status parameters

*proc2s* - F1 processing status parameters

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

XF1

### SEE ALSO

*xf2* [▶ 136], *xfb, ftf* [▶ 139], *xfb, ftf* [▶ 139], *xtrf, xtrf2* [▶ 154], *xtrfp, xtrfp2, xtrfp1* [▶ 157]

## 4.23 xfbm, xf2m, xf1m

### NAME

xfbm - Calculate magnitude spectrum in F2 and F1 (2D)

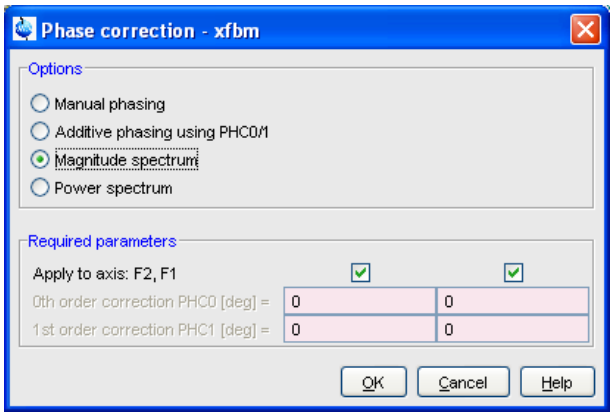
xf2m - Calculate magnitude spectrum in F2 (2D)

xf1m - Calculate magnitude spectrum in F1 (2D)

ph - Open phase correction dialog box (1D,2D)

### DESCRIPTION

The magnitude spectrum commands can be started from the command line or from the phase correction dialog box. The latter is started with the command **ph**:




This dialog box offers several options, each of which selects a certain command for execution.

**Magnitude spectrum (F2)**

This option selects the command **xf2m** for execution. It calculates the real and F2-imaginary data according to:

$$rr = \sqrt{rr^2 + ir^2}$$

$$ri = \sqrt{ri^2 + ii^2}$$

**Magnitude spectrum (F1)**

This option selects the command **xf1m** for execution. It calculates the real and F1-imaginary data according to according to:

$$rr = \sqrt{rr^2 + ri^2}$$

$$ir = \sqrt{ir^2 + ii^2}$$

**Magnitude spectrum (F12 and F1)**

This option selects the command **xfbm** for execution. It calculates the real andF1/F2-imaginary data according to according to:

$$rr = \sqrt{rr^2 + ir^2 + ri^2 + ii^2}$$

Where:

*rr* = real data (*2rr* file)

*ir* = F2-imaginary data (*2ir* file)

*ri* = F1- imaginary data (*2ri* file)

*ii* = F2/F1-imaginary data (*2ii* file)

The commands **xf*m** are, for example, used to convert a phase sensitive spectrum to magnitude spectrum. This is useful for data which cannot be phased properly or data which are not phase sensitive but have been acquired as such.

The **ph** command can be used on 1D or 2D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

## INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *2ir*, *2ri*, *2ii* - processed 2D data

## OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *2ir*, *2ri*, *2ii* - processed 2D data

*auditp.txt* - processing audit trail

## USAGE IN AU PROGRAMS

XFBM

XF2M

XF1M

### SEE ALSO

*xfbps, xf2ps, xf1ps [▶ 135]*

## 4.24 xfbps, xf2ps, xf1ps

### NAME

xfbps - Calculate power spectrum in F2 and F1 (2D)

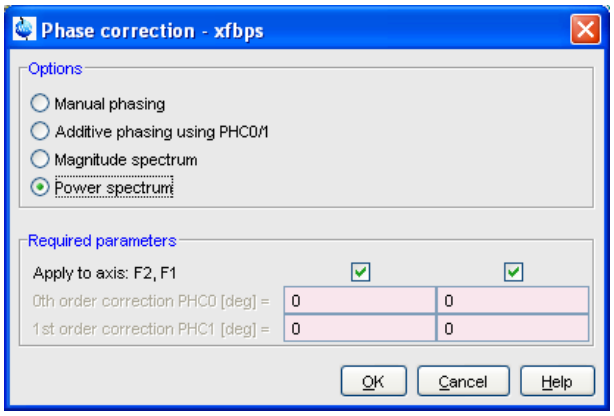
xf2ps - Calculate power spectrum in F2 (2D)

xf1ps - Calculate power spectrum in F1 (2D)

ph - Open phase correction dialog box (1D,2D)

### DESCRIPTION

The commands **xf*ps** calculate the magnitude spectrum. They can be started from the command line or from the phase correction dialog box. The latter is started with the command **ph**:




This dialog box offers several options, each of which selects a certain command for execution.

#### Power spectrum in F2

This option selects the command **xf2ps** for execution. It recalculates the real and F2-imaginary data according to:

$$rr = rr^2 + ir^2$$

$$ri = ri^2 + ii^2$$

#### Power spectrum (F1)

This option selects the command **xf1ps** for execution. It recalculates the real and F1-imaginary data according to:

$$rr = rr^2 + ri^2$$

$$ir = ir^2 + ii^2$$

**Power spectrum (F2 and F1)**

This option selects the command **xfbps** for execution. It recalculates the real according to:

$$rr = rr^2 + ir^2 + ri^2 + ii^2$$

Where:

*rr* = real data (*2rr* file)

*ir* = F2-imaginary data (*2ir* file)

*ri* = F1- imaginary data (*2ri* file)

*ii* = F2/F1-imaginary data (*2ii* file)

The commands **xf*ps** is, for example, used in special cases to convert a phase sensitive spectrum to a power spectrum. This is useful for data which cannot be phased properly or data which are not phase sensitive but have been acquired as such.

The **ph** command can be used on 1D or 2D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *2ir*, *2ri*, *2ii* - processed 2D data

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *2ir*, *2ri*, *2ii* - processed 2D data

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

XFBPS

XF2PS

XF1PS

**SEE ALSO**

*xfbm, xf2m, xf1m [▶ 133]*

## 4.25 xf2

**NAME**

xf2 - Process data, including FT, in F2 (2D)

## DESCRIPTION

The command **xf2** processes a 2D dataset in the F2 direction. It can be started from the command line or from the Fourier transform dialog box. The latter is opened with the command **ftf**.

**xf2** Fourier transforms time domain data (FID) into frequency domain data (spectrum). Depending on the F2 processing parameters BC_mod, WDW, ME_mod and PH_mod, **xf2** also performs baseline correction, window multiplication, linear prediction and phase correction, respectively. These steps are described in detail for the command **xfb**.

Normally, 2D data are processed with the command **xfb** which performs a Fourier transform in both directions, F2 and F1. In some cases, however, 2D data must only be processed in the F2 direction. Examples are T1, T2 or Dosy data, or a 2D dataset which has been created from a series on 1D datasets.

Even if a 2D dataset must be processed in both directions, it is sometimes useful to do that in two separate steps using the sequence **xf2** - **xf1**. The result is exactly the same as with **xfb** with one exception; **xfb** performs a quad spike correction (see **xfb**) and the sequence **xf2** - **xf1** does not.

**xf2** takes the same options as **xfb**. Furthermore, **xf2** takes the special option **nd2d** converting an nD dataset (n>2) to a 2D dataset processing it in the acquisition direction. The size in the orthogonal direction (F1-SI) of the destination 2D dataset, is the product of the TD values of the source nD dataset.

**xf2** can also be used to process one 2D plane of a 3D spectrum (see **xfb**).

## INPUT PARAMETERS

### F2 and F1 parameters

Set from the **ftf** dialog box, with **edp** or by typing **si**, **stsr** etc.:

SI - size of the processed data

STSR - strip start: first output point of strip transform

STSI - strip size: number of output points of strip transform

TDeff - number of raw data points to be used for processing

TDoff - first point of the FID used for processing (default 0)

XDIM - submatrix size (only used for the command **xf2 xdim**)

Set by the acquisition, can be viewed with **dpa** or by typing **s td**:

TD - time domain; number of raw data points

### F2 parameters

Set from the **ftf** dialog box, with **edp** or by typing **bc_mod** etc.

BC_mod - FID baseline correction mode

BCFW - filter width for BC_mod = sfil or qfil

COROFFS - correction offset for BC_mod = spol/qpol or sfil/qfil

ME_mod - FID linear prediction mode

NCOEF - number of linear prediction coefficients

LPBIN - number of points for linear prediction

TDoff - number of raw data points predicted for ME_mod = LPb*

WDW - FID window multiplication mode

LB - Lorentzian broadening factor for WDW = em or gm

GB - Gaussian broadening factor for WDW = gm, sinc or qsinc

SSB - Sine bell shift for WDW = sine, qsine, sinc or qsinc

TM1, TM2 - limits of the trapezoidal window for WDW = trap

PH_mod - phase correction mode
PHC0 - zero order phase correction value for PH_mod = pk
PHC1 - first order phase correction value for PH_mod = pk
FCOR - first (FID) data point multiplication factor (0.0-2.0, default 0.5)
REVERSE - flag indicating to reverse the spectrum
Set by the acquisition, can be viewed with **dpa** or by typing **s aq_mod**:
AQ_mod - acquisition mode (determines the Fourier transform mode)
BYTORDA - byteorder or the raw data
NC - normalization constant

**F1 parameters**

Set by the acquisition, can be viewed with **dpa** or by typing **s fnmode** :
FnMODE - Fourier transform mode

## OUTPUT PARAMETERS

**F2 and F1 parameters**

Can be viewed with **dpp** or by typing **s si**, **s tdeff** etc.:
SI - size of the processed data
TDeff - number of raw data points that were used for processing
STSR - strip start: first output point of strip transform
STSI - strip size: number of output points of strip transform
FTSIZE - Fourier transform size
XDIM - submatrix size

**F2 parameters**

Can be viewed with **dpp** or by typing **s ft_mod**, **s ymax_p** etc.:
FT_mod - Fourier transform mode
YMAX_p - maximum intensity of the processed data
YMIN_p - minimum intensity of the processed data
S_DEV - standard deviation of the processed data
NC_proc - intensity scaling factor
BYTORDP - byte order of the processed data

**F1 parameters**

Set by the acquisition, can be viewed with **dpp** or by typing **s mc2** :
MC2 - Fourier transform mode

## INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*
*ser* - raw data (input if *2rr* does not exist or is Fourier transformed in F2)
*acqus* - F2 acquisition status parameters
*acqu2s* - F1 acquisition parameters
*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*
*2rr* - processed data (input if it exists but is not Fourier transformed in F2)
*proc* - F2 processing parameters
*proc2* - F1 processing parameters

Note that if *2rr* is input, *2ri* is also input if xf1 has been done.

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - first quadrant real processed data

*2ir* - second quadrant imaginary processed data (output if FnMODE ≠ QF)

*2ii* - second quadrant imaginary processed data (output if FnMODE = QF)

*procs* - F2 processing status parameters

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

XF2

### SEE ALSO

*xf1 [ 131], xfb, ftf [ 139], xtrf, xtrf2 [ 154]*

## 4.26 xfb, ftf

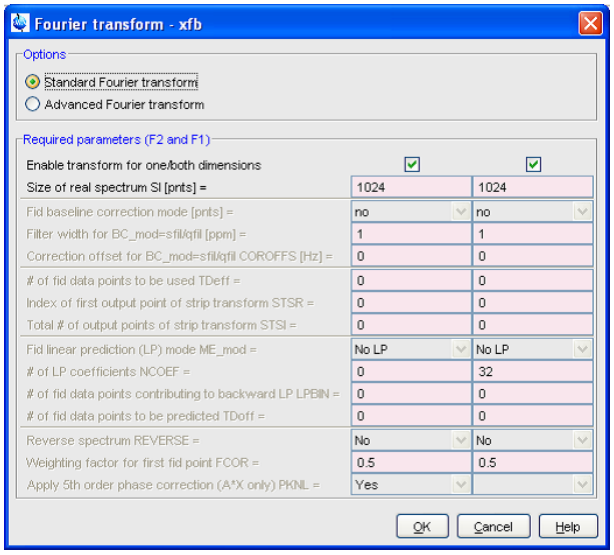
### NAME

xfb - Process data, including FT, in F2 and F1 (2D)

ftf - Open Fourier transform dialog box (1D,2D)

### DESCRIPTION

The command **xfb** processes a 2D dataset or a plane of a dataset with dimension ≥ 3. It can be started from the command line or from the Fourier transform dialog box. The latter is opened with the command **ftf**.




The **ftf** command recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters. For 2D data, two options appear, both of which select the **xfb** command for execution, provided the F2 and F1 direction are both enabled.

**Standard Fourier transform**

This option only allows to set the parameter SI, the size of the real spectrum.

**Advanced Fourier transform**

This option allows to set all Fourier transform related parameters.

**xfb** Fourier transforms time domain data into frequency domain data. Depending on the processing parameters BC_mod, WDW, ME_mod and PH_mod, **xfb** also performs baseline correction, window multiplication, linear prediction and spectrum phase correction.

The processing steps done by **xfb** can be described as follows:

1. Baseline correction of the 2D time domain data. Each row and/or column is baseline corrected according to BC_mod. This parameter takes the value *no*, *single*, *quad*, *spol*, *qpol sfil* or *qfil*. More details on BC_mod can be found in chapter *List of processing parameters* [▶ 20].
2. Linear prediction of the 2D time domain data. Linear prediction is done according to ME_mod. This parameter takes the value *no*, *LPfr*, *LPfc*, *LPbr*, *LPbc, LPmifr* or *LPmifc.* Usually, ME_mod = no, which means no prediction is done. Forward prediction (*LPfr*, *LPfc, LPmifr* or *LPmifc*) can, for example, be used to extend truncated FIDs. Backward prediction (*LPbr* or *LPbc*) can be used to improve the initial data points of the FID. Linear prediction is only performed for NCOEF > 0. Furthermore, LPBIN and, for backward prediction, TDoff play a role (see these parameters in chapter *List of processing parameters* [▶ 20]).
3. Window multiplication of the 2D time domain data. Each row and/or column is multiplied with a window function according to WDW. This parameter takes the value *em*, *gm*, *sine*, *qsine*, *trap*, *user*, *sinc*, *qsinc*, *traf* or *trafs*. More details on WDW can be found in chapter *List of processing parameters* [▶ 20].
4. Fourier transform of the 2D time domain data. Each row is Fourier transformed according to the acquisition status parameter AQ_mod as shown in the table below. Each column (F1) is Fourier transformed according to the acquisition status parameter FnMODE as shown in the table below. **xfb** does not evaluate the processing parameter FT_mod! However, it stores the Fourier transform mode as it was evaluated from AQ_mod (F2) or FnMODE (F1) in the processing status parameter FT_mod. If, for some reason, you want to Fourier transform a spectrum with a different mode, you can set the processing parameter FT_mod (with **edp**) and use the command **xtrf** (see **xtrf**). More details on FT_mod can be found in chapter *List of processing parameters* [▶ 20].
5. Phase correction of the 2D spectrum according to PH_mod. This parameter takes the value *no*, *pk*, *mc* or *ps*. For PH_mod = pk, **xfb** applies the values of PHC0 and PHC1. This is only useful if the phase values are known. If they are not, you can do an interactive phase correction in Phase correction mode after **xfb** has finished. More details on PH_mod can be found in chapter *List of processing parameters* [▶ 20].

| F2 AQ_mod | Fourier transform mode | F2 status FT_mod |
|---|---|---|
| qf | forward, single, real | fsr |
| qsim | forward, quad, complex | fqc |
| qseq | forward, quad, real | fqr |
| DQD | forward, quad, complex | fqc |

| F1 FnMODE | Fourier transform mode | F1 status FT_mod |
|---|---|---|
| QF | forward, quad, complex | fqc |
| QSEQ | forward, quad, real | fqr |
| TPPI | forward, single, real | fsr |
| States | forward, quad, complex | fqc |

| F1 FnMODE | Fourier transform mode | F1 status FT_mod |
|---|---|---|
| States-TPPI | forward, single, complex | fsc |
| Echo-AntiEcho | forward, quad, complex | fqc |

The size of the processed data is determined by the processing parameter SI; SI real and SI imaginary points are created. A typical value for SI is TD/2 in which case, all raw data points are used and no zero filling is done. In fact, several parameters control the number of input and output data points, for example:

1. SI > TD/2: the raw data are zero filled before the Fourier transform
2. SI < TD/2: only the first 2*SI raw data points are used
3. 0 < TDeff < TD: only the first TDeff raw data points are used
4. 0 < TDoff < TD: the first TDoff raw data points are cut off at the beginning and TDoff zeroes are appended at the end (corresponds to left shift).
5. TDoff < 0: -TDoff zeroes are prepended at the beginning. Note that:
   - for SI < (TD-TDoff)/2 raw data are cut off at the end
   - for DIGMOD=digital, the zeroes would be prepended to the group delay which does not make sense. You can avoid that by converting the raw data with **convdta** before you process them.
6. 0 < STSR < SI: only the processed data between STSR and STSR+STSI are stored (if STSI = 0, STSR is ignored and SI points are stored)
7. 0 < STSI < SI: only the processed data between STSR and STSR+STSI are stored.

Note that only in the first case the processed data contain the total information of the raw data. In all other cases, information is lost.

**xfb** performs a quad spike correction which means that the central data point of the spectrum is replaced by the average of the neighbouring data points in the F1 direction. Note that the quad spike correction is skipped if you process the data with the sequence **xf2** - **xf1**.

**xfb** evaluates the parameter FCOR. The first point of the FIDs is multiplied with the value of FCOR which lies between 0.0 and 2.0. For digitally filtered Avance data, FCOR is only used in the F1 direction. In F2, it has no effect because the first point is part of the group delay and, as such, is zero. However, A*X data or Avance data measured with DIGMOD = analog, FCOR is used in F1 and F2.

**xfb** evaluates the F2 parameter PKNL. On A*X spectrometers, PKNL = true causes a non linear 5th order phase correction of the raw data. This corrects possible errors caused by non linear behaviour of the analog filters. On Avance spectrometers, PKNL must always be set to TRUE. For digitally filtered data, it causes **xfb** to handle the group delay of the FID. For analog data it has no effect.

**xfb** evaluates the F2 and F1 parameter REVERSE. If REVERSE = TRUE, the spectrum will be reversed in the corresponding direction, i.e. the first data point becomes the last and the last data point becomes the first. The same effect can be obtained with the commands **rev2** and/or **rev1** after **xfb**.

**USAGE:**

**xfb** is normally used without options. There are, however, several options available:

- **n**
  - **xfb** normally stores real and imaginary processed data. However, the imaginary data are only needed for phase correction. If the parameters PHC0 and PHC1 are set correctly, then you don't need to store the imaginary data. The option **n** allows to do

that. This will save processing time and disk space. If you still want to do a phase correction, you can create imaginary data from the real data with a Hilbert transform (see **xht2** and **xht1**).

- **nc_proc value**
  - **xfb** scales the data such that, i.e. the highest intensity of the spectrum lies between $2^{28}$ and $2^{29}$. The intensity scaling factor is stored in the processing status parameter NC_proc and can be viewed with **dpp**. The option **nc_proc** causes **xfb** to use a specific scaling factor. However, you can only scale down the data by entering a greater (more positive) value than the one **xfb** would use without this option. If you enter a smaller (more negative) value, the option will be ignored to prevent data overflow. The option **nc_proc last** causes **xfb** to use the current value of the status processing parameter NC_proc, i.e. the value set by the previous processing step on this dataset.
- **raw/proc**
  - **xfb** works on raw data if no processed data exist or if processed data exist and have been Fourier transformed in F2 and/or F1. One of them is usually true, i.e. the data have not been processed yet or they have been processed, for example with **xfb**. If, however, the data have been processed with **xtrf** with FT_mod = no, they are not Fourier transformed and a subsequent **xfb** will work on the processed data. The **raw** option causes **xfb** to work on the raw data, no matter what. The **proc** option causes **xfb** to work on the processed data. If these do not exist or are Fourier transformed, the command stops and displays an error message. In other words, the option **proc** prevents **xfb** to work on raw data.
- **big/little**
  - **xfb** stores the data in the data byte order (big or little endian) of the computer it runs on e.g. little endian on Windows PCs. Note that TopSpin's predecessor XWIN-NMR on SGI UNIX workstations stores data in big endian. The byte order is stored in the processing status parameter BYTORDP which can be viewed with **s bytordp**. The option **big** or **little** allows to predefine the byte order. This, for example, is used to read processed data with third party software which can not interpret BYTORDP. This option is only evaluated when **xfb** works on the raw data.
- **xdim**
  - Large 2D spectra are stored in the so-called submatrix format. The size of the submatrices are calculated by **xfb** and depend on the size of the spectrum and the available memory. The option **xdim** allows to use predefined submatrix sizes. It causes **xfb** to interpret the F2 and F1 processing parameter XDIM which can be set by entering **xdim** on the command line. The actually used submatrix sizes, whether predefined or calculated, are stored as the F2 and F1 processing status parameter XDIM and can be viewed with **dpp**. Predefining submatrix sizes is, for example, used to read the processed data with third party software which can not interpret the processing status parameter XDIM. This option is only evaluated when **xfb** works on the raw data.

Normally, **xfb** stores the entire spectral region as determined by the spectral width. You can, however, do a so-called strip transform which means that only a certain region of the spectrum is stored. This can be done by setting the parameters STSR and STSI which represent the strip start and strip size, respectively. They both can take a value between 0 and SI. The values which are actually used can be a little different. STSI is always rounded to the next multiple of 16. Furthermore, when the data are stored in submatrix format (see below), STSI is rounded to the next higher multiple of the submatrix size. Type **dpp** to check this; if XDIM is smaller than SI, then the data are stored in submatrix format and STSI is a multiple of XDIM.

Depending on size of the processed data and the available computer memory, **xfb** stores the data in sequential or submatrix format. Sequential format is used when the entire dataset fits in memory, otherwise submatrix format is used. **xfb** automatically calculates the submatrix sizes such that one row (F2) of submatrices fits in the available memory. The calculated

submatrix sizes are stored in the processing status parameter XDIM (type **dpp**). The next two tables show the alignment of the data points for sequential and submatrix format, respectively. This example shows a dataset with the following sizes: F2 SI = 16, F1 SI = 16, F2 XDIM = 8, F1 XDIM = 4. The storage handling is completely transparent to the user and is only of interest when the data are interpreted by third party software.

F2 →
F1 ↓

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 32 | 33 | 34 | 35 | 36 | 38 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
| 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 |
| 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 |
| 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 |
| 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 |
| 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 |
| 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 |
| 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 |
| 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 | 250 | 251 | 252 | 253 | 254 | 255 |

*Figure 4.1: 2D data in sequential storage format*

F2 →
F1 ↓

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 |
| 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 |
| 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
| 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 |
| 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 |
| 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 | 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 |
| 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 |
| 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 |
| 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 | 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 |
| 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 |
| 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 | 248 | 249 | 250 | 251 | 252 | 253 | 254 | 255 |

*Figure 4.2: 2D data in 8*4 submatrix storage format*

As can be seen in the second table *F1 FnMODE* of this chapter, the acquisition mode in F1 (FnMODE) determines the Fourier transform mode. Furthermore, FnMODE determines the data storage mode. The description below demonstrates the difference in data storage between a data set with FnMODE = QF and one with FnMODE ≠ QF.

**FnMODE = QF**

**xfb** performs complex (two-quadrant) processing. In F2 the data are acquired phase sensitive, in F1 non-phase sensitive. In the example below, the following parameter settings are used:

In F2: TD = 8, SI is 4

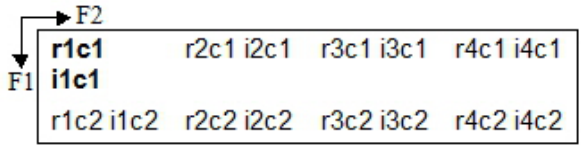
In F1: TD = 2, SI = 2

Furthermore, the following notation is used for individual data points:

**rncm** : point *n* of FID *m*. This point is real in F2 and complex in F1

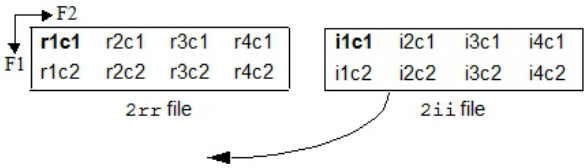
**incm** : point *n* of FID *m*. This point is imaginary in F2 and complex in F1

### Input F2 processing (raw data)



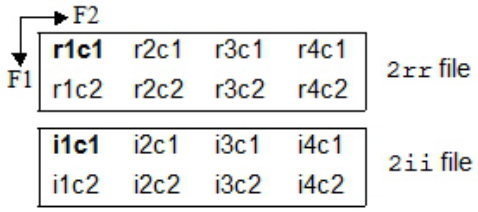

For F2 processing, **r1c1 i1c1** is the first complex input point, r2c1 i2c1 the second etc.

### Output F2 processing = Input F1 processing

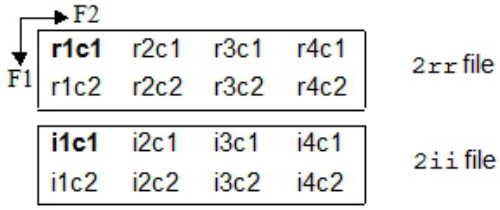



Below, the F1 input data are simply redisplayed in vertical order, with the first complex input point in bold.

### Input F1 processing




### Output F1 processing




### FnMODE ≠ QF

**xfb** performs hypercomplex (four-quadrant) processing. Both in F2 and F1, the data are acquired phase sensitive. In the example below, the following parameters settings are used:

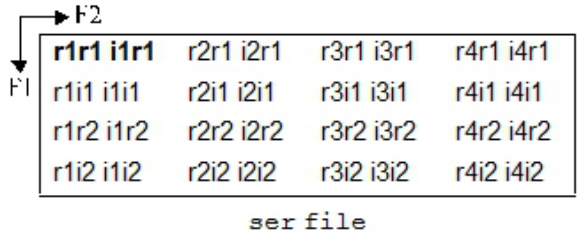
In F2: TD = 8, SI is 4

In F1: TD = 4, SI = 2

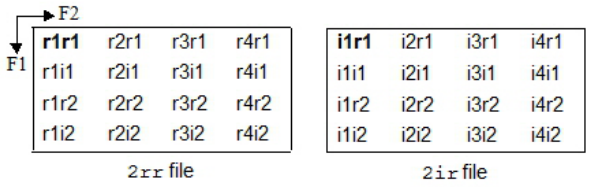
Furthermore, the following notation is used for individual data points:

- **rnrm** : point *n* of FID *m*. This point is real in F2 and F1
- **inrm** : point *n* of FID *m*. This point is imaginary in F2 and real in F1
- **rnim**: point *n* of FID *m*. This point is real in F2 and imaginary in F1
- **inim** : point *n* of FID *m*. This point is imaginary in F2 and F1

### Input F2 processing (raw data)

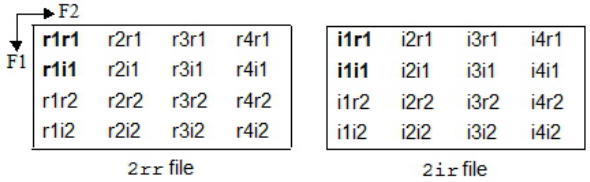



For F2 processing, **r1r1 i1r1** is the first hypercomplex input data point, r2r1 i2r1 the second etc. **Output F2 processing = Input F1 processing**

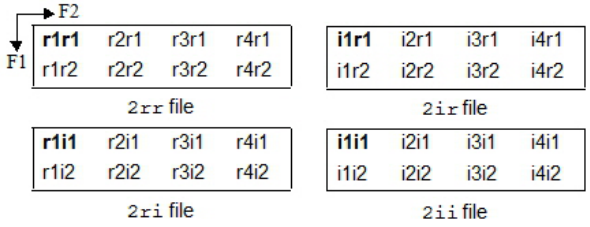



Below, the F1 input data are simply redisplayed, with the first F1 complex input points in bold.

### Input F1 processing




### Output F1 processing




### FnMODE = Echo-Antiecho

**xfb** performs hypercomplex (four-quadrant) processing. Both in F2 and F1, the data are acquired phase sensitive. In the example below, the following parameters settings are used:

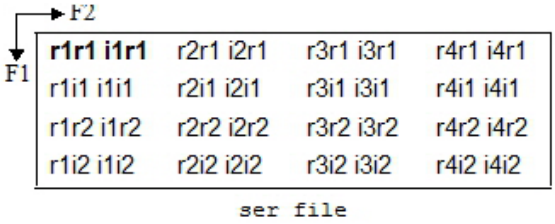
In F2: TD = 8, SI is 4

In F1: TD = 4, SI = 2

Furthermore, the following notation is used for individual data points:

- **rnrm** : point *n* of FID *m*. This point is real in F2 and F1
- **inrm** : point *n* of FID *m*. This point is imaginary in F2 and real in F1
- **rnim**: point *n* of FID *m*. This point is real in F2 and imaginary in F1
- **inim** : point *n* of FID *m*. This point is imaginary in F2 and F1

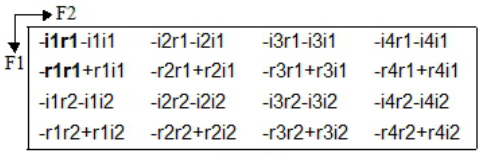
### Input F2 processing (raw data)



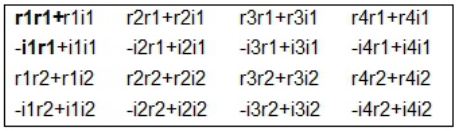

ser file

For F2 processing, **r1r1 i1r1** is the first hyper complex input data point, r2r1 i2r1 the second etc.

### Output F2 processing = Input F1 processing



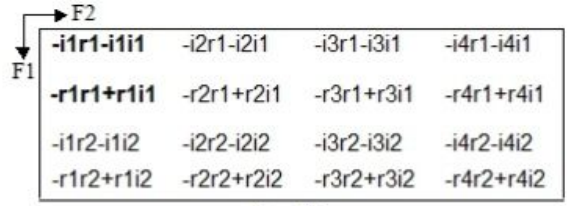

2rr file



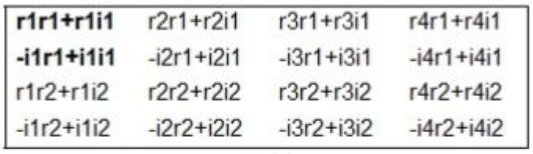

2ir file

Below, the F1 input data are simply redisplayed, with the first F1 complex input points in bold.

### Input F1 processing

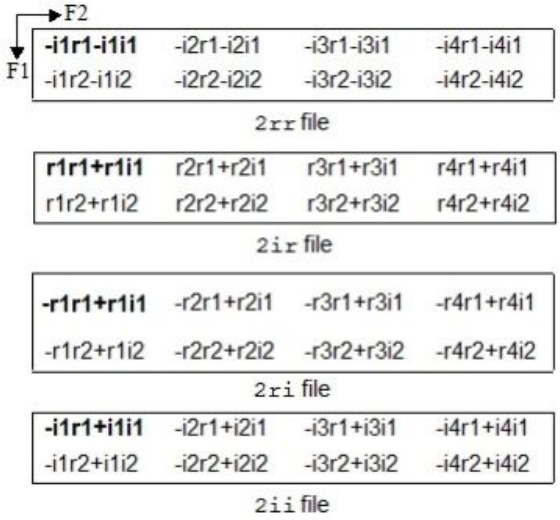



2rr file




2ir file

### Output F1 processing




Note that:

- For FnMODE ≠ QF, zero filling once in F1 is done when SI = TD. For FnMODE = QF, zero filling once in F1 is done when SI = 2*TD.
- FnMODE = QF is normally used on magnitude or power data. For this purpose, the F1 processing parameter PH_mod must be set to MC or PS, respectively. Note that in these cases, no imaginary data are stored after F1 processing.
- FnMODE = Echo-Antiecho is equivalent to FnMODE = States, except that two consecutive FIDs (rows of the 2D raw data) are linearly combined according to the following rules:
  - re0 = -im1 - im0
  - im0 = re1 + re0
  - re1 = re1 - re0
  - im1 = im1 - im0
- xfb n does not store imaginary data after F1 processing.

### 2D PROCESSING OF 3D DATA

**xfb** can also be used to process one 2D plane of a 3D spectrum. This can be a plane in the F3-F2 or in the F3-F1 direction. The output 2D data are stored in a separate *procno*. When the current dataset is a 3D, **xfb** will prompt you for the plane axis direction, the plane number, the output *procno* and, if applicable, for the permission to overwrite existing data. Alternatively, you can enter this information as arguments on the command line, for example:

**xfb s23 17 2 y**

Will read the F3-F2 plane number 17 and store it under procno 2, overwriting possibly existing data. Furthermore, you can use the **nodisp** argument to prevent opening/displaying the destination dataset, e.g.:

**xfb s23 17 2 y nodisp**

For 2D processing of 3D echo-antiecho (EA) data the option **eao** is available. This option ensures EA calculation when:

- the 3D raw data are EA in either F2 or F1 (the acquisition status parameter FnMODE = Echo-Antiecho in F2 or F1, respectively)
- the processed plane does not include the EA direction

For example, to process F2-F3 plane 17 of a 3D dataset which is EA in F1, enter:

**xfb eao s23 17 2 y**

If you omit the **eao** option, the plane is still processed but no EA calculation is done. Using the **eao** option allows to determine the correct phase values for EA data or compare the processed plane with a plane extracted from a 3D processed data. Note that if the processed plane includes the EA direction, or if the 3D data are not EA in any direction, the option **eao** has no effect.

When executed on a dataset with 3D raw data but 2D processed data (usually a result of a previous 2D processing command on that 3D dataset), **xfb** takes one argument:

**xfb <plane>**

Process the specified plane and store it under the current *procno*.

**xfb same**

Process the same plane as the previous processing command and store it under the current *procno*. The **same** option is automatically used by the AU program macro XFB. When used on a regular 2D dataset (i.e. with 2D raw data), it has no effect.

## INPUT PARAMETERS

### F2 and F1 parameters

Set from the **ftf** dialog box, with **edp** or by typing **bc_mod**, **bcfw** etc.

BC_mod - FID baseline correction mode

BCFW - filter width for BC_mod = sfil or qfil

COROFFS - correction offset for BC_mod = spol/qpol or sfil/qfil

ME_mod - FID linear prediction mode

NCOEF - number of linear prediction coefficients

LPBIN - number of points for linear prediction

TDoff - number of raw data points predicted for ME_mod = LPb*

WDW - FID window multiplication mode

LB - Lorentzian broadening factor for WDW = em or gm

GB - Gaussian broadening factor for WDW = gm, sinc or qsinc

SSB - Sine bell shift for WDW = sine, qsine, sinc or qsinc

TM1, TM2 - limits of the trapezoidal window for WDW = trap

PH_mod - phase correction mode

PHC0 - zero order phase correction value for PH_mod = pk

PHC1 - first order phase correction value for PH_mod = pk

SI - size of the processed data

STSR - strip start: first output point of strip transform

STSI - strip size: number of output points of strip transform

TDeff - number of raw data points to be used for processing

TDoff - first point of the FID used for processing (default 0)

FCOR - first (FID) data point multiplication factor (0.0-2.0, default 0.5)

REVERSE - flag indicating to reverse the spectrum

XDIM - submatrix size (only used for the command **xfb xdim**)

Set by the acquisition, can be viewed with **dpa** or by typing **s td** :

TD - time domain; number of raw data points

### F2 parameters

Set from the **ftf** dialog box, with **edp** or by typing **pknl** :

PKNL - group delay compensation (Avance) or filter correction (A*X)
Set by the acquisition, can be viewed with **dpa** or by typing **s aq_mod**.:
AQ_mod - acquisition mode (determines the Fourier transform mode)
BYTORDA - byteorder or the raw data
NC - normalization constant

**F1 parameters**
Set by the acquisition, can be viewed with **dpa** or by typing **s fnmode** :
FnMODE - F1 Acquisition transform mode
Set by the user with **edp** or by typing **mc2** :
MC2 - FT mode in F1 (only used if F1-FnMODE = undefined)

**OUTPUT PARAMETERS**

**F2 and F1 parameters**
Can be viewed with **dpp** or by typing **s si**, **s tdeff** etc.:
SI - size of the processed data
TDeff - number of raw data points that were used for processing
FTSIZE - Fourier transform size
STSR - strip start: first output point of strip transform
STSI - strip size: number of output points of strip transform
XDIM - submatrix size
FT_mod - Fourier transform mode

**F2 parameters**
Can be viewed with **dpp** or by typing **s ymax_p**, **s ymin_p** etc.:
YMAX_p - maximum intensity of the processed data
YMIN_p - minimum intensity of the processed data
S_DEV - standard deviation of the processed data
NC_proc - intensity scaling factor
BYTORDP - byte order of the processed data

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*
*ser* - raw data (input if *2rr* does not exit or is Fourier transformed)
*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*
*2rr* - real processed 2D data (input if it exists but is not Fourier transformed)
*proc* - F2 processing parameters
*proc2* - F1 processing parameters
*acqus* - F2 acquisition status parameters
*acqu2s* - F1 acquisition status parameters

Note that if *2rr* is input, then *2ir* and *2ri* can also be input, depending on the processing status of the data.

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

**For FnMODE □ QF:**

*2rr* - real processed 2D data

*2ir* - second quadrant imaginary processed data

*2ri* - third quadrant imaginary processed data

*2ii* - fourth quadrant imaginary processed data

**For FnMODE = QF:**

*2rr* - real processed 2D data

*2ii* - second quadrant imaginary processed data

**For all values of FnMODE:**

*procs* - F2 processing status parameters

*proc2s* - F1 processing status parameters

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

XFB

If you want to use XFB with an option, you can do that with XCMD, e.g.

XCMD("xfb raw")

**SEE ALSO**

*xf1 [*▸ 131], *xf2 [*▸ 136], *xfbm, xf2m, xf1m [*▸ 133], *xfbp, xf2p, xf1p [*▸ 150], *xfbps, xf2ps, xf1ps [*▸ 135], *xtrf, xtrf2 [*▸ 154]

## 4.27 xfbp, xf2p, xf1p

**NAME**

xfbp - Phase correction in F2 and F1 direction (2D)

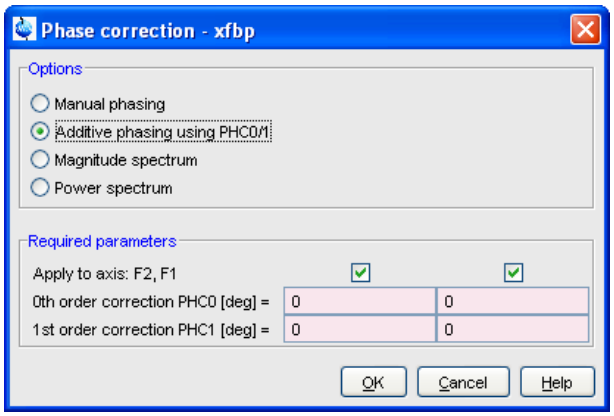
xf2p - Phase correction in F2 (2D)

xf1p - Phase correction in F1 (2D)

ph - Open phase correction dialog box (1D,2D)

**DESCRIPTION**

2D phase correction can be started from the command line or from the phase correction dialog box. The latter is opened with the command **ph**:




This dialog box offers several options, each of which selects a certain command for execution.

**Additive phasing using PHC0/1 (F2 and F1)**

This option selects the command **xfbp** for execution. It performs a zero and first order 2D phase correction in the F2 and F1 direction. **xfbp** works like the 1D command **pk**. This means it does not calculate the phase values, it simply applies the current values of PHC0 and PHC1.

**Additive phasing using PHC0/1 (F2)**

This option selects the command **xf2p** for execution. It works like **xfbp**, except that it only corrects the phase in the F2 direction.

**Additive phasing using PHC0/1 (F1)**

This option selects the command **xf1p** for execution. It works like **xfbp**, except that it only corrects the phase in the F1 direction.

**xf*p** are only useful when the PHC0 and PHC1 values are known. If they are not, you can perform 2D interactive phase correction. To do that, select the option *Manual Phasing* in the **ph** dialog box or click in the toolbar. The interactive phase correction procedure is described in the TopSpin Users Guide.

The phase values can also be determined by the 1D interactive phase correction of a row or column. To do that, read a row (**rsr**) and/or column (**rsc**) and click in the toolbar (see TopSpin Users Guide). Alternatively, you can phase correct a row or column with **apk** and view the calculated phase values with **dpp**. Then you can go back to the 2D dataset, set the determined phase values with **edp** and run **xfbp** to apply them.

**xfbp** uses but does not change the processing parameters PHC0 and PHC1 (**edp**). It does, however, change the corresponding processing status parameters (**dpp**), by adding the applied phase values.

The **ph** command can be used on 1D or 2D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

## INPUT PARAMETERS

Set from the **ph** dialog box, with **edp** or by typing **phc0**, **phc1**:

PHC0 - zero order phase correction value (frequency independent)

PHC1 - first order phase correction value (frequency dependent)

## OUTPUT PARAMETERS

Can be viewed with **dpp** or by typing **s phc0**, **s phc1**:

PHC0 - zero order phase correction value (frequency independent)

PHC1 - first order phase correction value (frequency dependent)

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *ir*, *2ri*, *2ii* - processed 2D data

*procs* - F2 processing status parameters

*proc2s* - F1 processing status parameters

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *ir*, *2ri*, *2ii* - processed 2D data

*procs* - F2 processing status parameters

*proc2s* - F1 processing status parameters

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

XFBP

XF2P

XF1P

### SEE ALSO

*xfb, ftf* [▷ 139], *xf2* [▷ 136], *xf1* [▷ 131], *xtrf, xtrf2* [▷ 154], *xtrfp, xtrfp2* [▷ 157]

## 4.28 xht2, xht1

### NAME

xht2 - Hilbert transform in F2 (2D)

xht1 - Hilbert transform in F1 (2D)

### DESCRIPTION

The command **xht2** performs a Hilbert transform of 2D data in the F2 direction.

The command **xht1** performs a Hilbert transform of 2D data in the F1 direction.

Hilbert transform creates imaginary data from the real data. Imaginary data are required for phase correction. They are normally created during Fourier transform with **xfb**, **xf2** or **xf1**. If, however, the imaginary data were not stored (**xfb n**) or have been deleted (**deli**), you can (re)create them with **xht2** or **xht1**.

Note that Hilbert Transform is only useful when the real data have been created from zero filled raw data, with SI ≥ TD.

Hilbert transform can also be used if the imaginary data exist but do not match the real data. This is the case when the latter have been manipulated after Fourier transform, for example by **abs1**, **abs2**, **sub***, **sym** or third party software.

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

*2ir* - second quadrant imaginary data (if existing, input of **xht1**)

*2ri* - third quadrant imaginary data (if existing, input of **xht2**)

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

*2ir* - second quadrant imaginary data (output of **xht2**, created from *2rr*)

*2ri* - third quadrant imaginary data (output of **xht1**, created from *2rr*)

*2ii* - fourth quadrant imaginary data

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

XHT2

XHT1

### SEE ALSO

*xfb, ftf [▶ 139]*, *xf2 [▶ 136]*, *xf1 [▶ 131]*

## 4.29 xif2, xif1

### NAME

xif2 - Inverse Fourier transform in F2 (2D)

xif1 - Inverse Fourier transform in F1 (2D)

### DESCRIPTION

The command **xif2** performs an inverse Fourier transform in the F2 direction. This means frequency domain data (spectrum) are transformed into time domain data (FID).

**xif1** performs an inverse Fourier transform in the F1 direction.

Note that after xif2 or xif1 (or both), the data are still stored as processed data, i.e. the raw data are not overwritten. You can, however, create pseudo-raw data with the command genser which creates a new dataset.

Inverse Fourier transform can also be done with the commands **xtrfp**, **xtrfp2** and **xtrfp1**. To do that:

1. Type **dpp** and check the status FT_mod.
2. Type **edp** to set the processing parameters; set BC_mod, WDW, ME_mod and PH_mod to *no* and FT_mod to the inverse equivalent of the status FT_mod.
3. Perform **xtrfp**, **xtrfp2** or **xtrfp1**.

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *ir*, *2ri*, *2ii* - processed 2D data

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *ir*, *2ri*, *2ii* - processed 2D data

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

XIF2

XIF1

### SEE ALSO

*genser* [▷ 108], *xtrfp, xtrfp2* [▷ 157]

## 4.30 xtrf, xtrf2

### NAME

xtrf - Custom processing of raw data in F2 and F1 (2D)

xtrf2 - Custom processing of raw data in F2 (2D)

### DESCRIPTION

The command **xtrf** performs customized processing of the raw data in both the F2 and F1 direction. It processes data according to the processing parameters BC_mod, WDW, ME_mod, FT_mod and PH_mod. **xtrf** works like **xfb**, except for the following differences:

1. The Fourier transform is performed according to the processing parameter FT_mod, whereas the acquisition status parameter AQ_mod is ignored. This, for example, allows to process the data without Fourier transform (FT_mod = no). Furthermore, you can choose a Fourier transform mode different from the one that would be evaluated from the acquisition mode. This feature is not used very often because the Fourier transform as evaluated from the acquisition mode is usually the correct one. If, however, you want to manipulate the acquisition mode of the raw data, you can Fourier transform the data with one FT_mod, inverse Fourier transform them with a different FT_mod. Then you can use **genser** to create pseudo-raw data with a different acquisition mode than the original raw data. The table below shows a list of values of FT_mod.
2. A baseline correction is performed according to BC_mod. This parameter can take the value *no*, *single*, *quad*, *spol*, *qpol*, *sfil* or *qfil.* **xtrf** evaluates BC_mod for the baseline correction mode (e.g. quad, qpol or qfil) and for the detection mode (e.g. single or quad, spol or qpol, sfil or qfil). Note that **xfb** evaluates the acquisition status parameter AQ_mod for the detection mode. More details on BC_mod can be found in chapter *List of processing parameters* [▷ 20].
3. When all parameters mentioned above are set to *no*, no processing is done but the raw data are still stored as processed data and displayed on the screen. This means the raw data are converted to submatrix format (files *2rr*, *2ir*, *2ri* and *2ii*) and scaled according to the vertical resolution. The intensity scaling factor is stored in the processing status parameter NC_proc and can be viewed with **dpp**. The size of these processed data and the number of raw data points which are used are determined by the parameters SI, TDeff and TDoff, as described for the command **xfb**. For example, if 0 < TDeff < TD, the processed data are truncated. This allows to create pseudo-raw data with a smaller size than the original raw data (see also **genser**).

| FT_mod | Fourier transform mode |
|---|---|
| no | no Fourier transform |
| fsr | forward, single channel, real |
| fqr | forward, quadrature, real |
| fsc | forward, single channel, complex |
| fqc | forward, quadrature, complex |
| isr | inverse, single channel, real |

| FT_mod | Fourier transform mode |
|---|---|
| iqr | inverse, quadrature, real |
| isc | inverse, single channel, complex |
| iqc | inverse, quadrature, complex |

The F1 Fourier transform mode and data storage mode depends on the F1 acquisition mode (see INPUT PARAMETERS below and the description of **xfb**).

**xtrf2** works like **xtrf**, except that it only works in the F2 direction.

**xtrf** and **xtrf2** take the same options as **xfb**.

**xtrf** can be used to do a combination of forward and backward prediction.

Run **xtrf** with ME_mod = LPfc and **xtrfp** (or **xfb**) with ME_mod = LPbc.

**INPUT PARAMETERS**

**F2 and F1 direction**

Set by the user with **edp** or by typing **si**, **bc_mod**, **bcfw** etc.:

SI - size of the processed data

TDeff - number of raw data points to be used for processing

TDoff - first point of the FID used for processing (default 0)

FCOR - first (FID) data point multiplication factor (0.0-2.0, default 0.5)

BC_mod - FID baseline correction mode

BCFW - filter width for BC_mod = sfil or qfil

COROFFS - correction offset for BC_mod = spol/qpol or sfil/qfil

ME_mod - FID linear prediction mode

NCOEF - number of linear prediction coefficients

LPBIN - number of points for linear prediction

TDoff - number of raw data points predicted for ME_mod = LPb*

WDW - FID window multiplication mode

LB - Lorentzian broadening factor for WDW = em or gm

GB - Gaussian broadening factor for WDW = gm, sinc or qsinc

SSB - Sine bell shift for WDW = sine, qsine, sinc or qsinc

TM1, TM2 - limits of the trapezoidal window for WDW = trap

FT_mod - Fourier transform mode

STSR - strip start: first output point of strip transform

STSI - strip size: number of output points of strip transform

REVERSE - flag indicating to reverse the spectrum

PKNL - group delay compensation (Avance) or filter correction (A*X)

PH_mod - phase correction mode

PHC0 - zero order phase correction value for PH_mod = pk

PHC1 - first order phase correction value for PH_mod = pk

Set by the acquisition, can be viewed with **dpa** or by typing **s td** :

TD - time domain; number of raw data points

**F2 direction**

Set by the acquisition, can be viewed with **dpa** or by typing **s bytorda**:

BYTORDA - byteorder or the raw data
NC - normalization constant
**F1 direction**
Set by the acquisition, can be viewed with **dpa** or by typing **s fnmode**:
FnMODE - Acquisition mode

**OUTPUT PARAMETERS**

**F2 and F1 parameters**
Can be viewed with **dpp** or by typing **s si** etc.:
SI - size of the processed data
TDeff - number of raw data points that were used for processing
STSR - strip start: first output point of strip transform
STSI - strip size: number of output points of strip transform
XDIM - submatrix size
**F2 parameters**
Can be viewed with **dpp** or by typing **s ymax_p**, **s ymin_p** etc.:
YMAX_p - maximum intensity of the processed data
YMIN_p - minimum intensity of the processed data
S_DEV - standard deviation of the processed data
NC_proc - intensity scaling factor
BYTORDP - byte order of the processed data

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*
*ser* - raw data
*acqus* - F2 acquisition status parameters
*acqu2s* - F1 acquisition status parameters
*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*
*proc* - F2 processing parameters
*proc2* - F1 processing parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*
*2rr*, *2ir*, *2ri*, *2ii* - processed 2D data
*procs* - processing status parameters
*proc2s* - processing status parameters
*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

XTRF
XTRF2

**SEE ALSO**

*xtrfp, xtrfp2 [▸ 157]*, *xfb, ftf [▸ 139]*, *xf2 [▸ 136]*, *xf1 [▸ 131]*

## 4.31 xtrfp, xtrfp2, xtrfp1

### NAME

xtrfp - Custom processing of processed data in F2 and F1 (2D)

xtrfp2 - Custom processing of processed data in F2 (2D)

xtrfp1 - Custom processing of processed data in F1 (2D)

### DESCRIPTION

The command **xtrfp** performs customized processing of processed data both the F2 and F1 direction. It works like **xtrf**, except that it only works on processed data. If processed data do not exist, an error message is displayed. If processed data do exist, they are further processed according to the parameters BC_mod, WDW, ME_mod, FT_mod and PH_mod as described for **xtrf**.

**xtrfp2** works like **xtrfp**, except that it only works in the F2 direction.

**xtrfp1** works like **xtrfp**, except that it only works in the F1 direction.

The **xtrfp*** commands can, for example, be used to perform multiple additive baseline corrections. This can be necessary if the raw data contain multiple frequency baseline distortions. You cannot do this with **xfb** or **xtrf** because these commands always work on the raw data, i.e. they are not additive.

**xtrfp**, **xtrfp2** and **xtrfp1** can also be used for inverse Fourier transform. To do that:

1. Type **dpp** to check the status FT_mod
2. Type **edp** to set the processing parameters; set BC_mod, WDW, ME_mod and PH_mod to *no* and FT_mod to the inverse equivalent of the status FT_mod
3. Perform **xtrfp**, **xtrfp2** or **xtrfp1**

As an alternative way to perform an inverse Fourier transform use the commands **xif2** and **xif1**.

### INPUT PARAMETERS

**F2 and F1 parameters**

Set by the user with **edp** or by typing **bc_mod**, **bcfw** etc.:

BC_mod - FID baseline correction mode

BCFW - filter width for BC_mod = sfil or qfil

COROFFS - correction offset for BC_mod = spol/qpol or sfil/qfil

ME_mod - FID linear prediction mode

NCOEF - number of linear prediction coefficients

LPBIN - number of points for linear prediction

TDoff - number of raw data points predicted for ME_mod = LPb*

WDW - FID window multiplication mode

LB - Lorentzian broadening factor for WDW = em or gm

GB - Gaussian broadening factor for WDW = gm, sinc or qsinc

SSB - Sine bell shift for WDW = sine, qsine, sinc or qsinc

TM1, TM2 - limits of the trapezoidal window for WDW = trap

FT_mod - Fourier transform mode

PH_mod - phase correction mode

PHC0 - zero order phase correction value for PH_mod = pk

PHC1 - first order phase correction value for PH_mod = pk

FCOR - first (FID) data point multiplication factor (0.0-2.0, default 0.5)
REVERSE - flag indicating to reverse the spectrum
Set by a previous processing command, e.g. **xtrf**, can be viewed with **dpp** :
SI - size of the processed data
STSR - strip start: first output point of strip transform
STSI - strip size: number of output points of strip transform
TDeff - number of raw data points to be used for processing
TDoff - first point of the FID used for processing (default 0)

**F1 parameters**

Set by a previous processing command, e.g. **xtrf**, can be viewed with **dpp** :
MC2 - Fourier transform mode

## OUTPUT PARAMETERS

**F2 parameters**

Can be viewed with **dpp** or by typing **s ymax_p**, **s ymin_p** etc.:
YMAX_p - maximum intensity of the processed data
YMIN_p - minimum intensity of the processed data
S_DEV - standard deviation of the processed data
NC_proc - intensity scaling factor
BYTORDP - byte order of the processed data

## INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*
*2rr*, *2ir*, *2ri*, *2ii* - processed 2D data
*proc* - F2 processing parameters
*proc2* - F1 processing parameters

## OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*
*2rr*, *2ir*, *2ri*, *2ii* - processed 2D data
*procs* - F2 processing status parameters
*proc2s* - F1 processing status parameters
*auditp.txt* - processing audit trail

## USAGE IN AU PROGRAMS

XTRFP
XTRFP2
XTRFP1

## SEE ALSO

*xtrf, xtrf2 [* 154], *xfb, ftf [* 139], *xf2 [* 136], *xf1 [* 131]

## 4.32 zert2, zert1, zert

### NAME

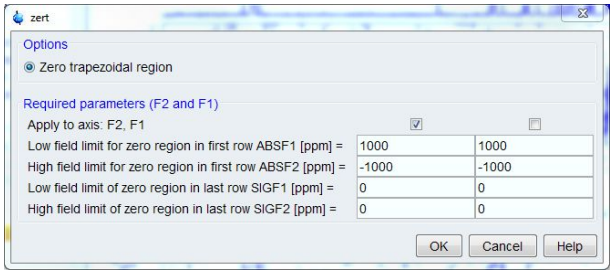
zert2 - Zero a trapezoidal region of each row (2D)

zert1 - Zero a trapezoidal region of each column (2D)

zert - Open zero region dialog box (2D)

### DESCRIPTION

The zero region commands can be started from the command line or from the zero region dialog box. The latter is opened with the command **zert**.




This dialog box offers only one option which can be used in the F2 or F1 direction.

**Zero trapezoidal region in F2**

This option selects the command **zert2** for execution. The trapezoidal region to be zeroed is defined as follows:

- Only the rows between F1-ABSF2 and F1-ABSF1 are zeroed
- The part (region) of each row which is zeroed shifts from row to row. The first row is zeroed between F2-ABSF2 and F2-ABSF1. The last row is zeroed between F2-SIGF2 and F2-SIGF1. For intermediate rows, the low field limit is an interpolation of F2-ABSF2 and F2-SIGF2 and the high field limit is an interpolation of F2-ABSF1 and F2-SIGF1.

**zert2** works exactly like **abst2**, except that the data points are zeroed instead of baseline corrected.

**Zero trapezoidal region in F1**

This option selects the command **zert1** for execution. The trapezoidal region to be zeroed is defined as follows:

- Only the columns between F2-ABSF2 and F2-ABSF1 are zeroed
- The part (region) of each column which is zeroed shifts from column to column. The first column is zeroed between F1-ABSF2 and F1-ABSF1. The last column is zeroed between F1-SIGF2 and F1-SIGF1. For intermediate columns, the low field limit is an interpolation of F1-ABSF2 and F1-SIGF2 and the high field limit is an interpolation of F1-ABSF1 and F1-SIGF1.

**zert1** works exactly like **abst1**, except that the data points are zeroed instead of baseline corrected.

### INPUT PARAMETERS

Set from the **zert** dialog box, with **edp** or by typing **absf1**, **absf2** etc.:

ABSF1 - low field limit of the zero region in the first row

ABSF2 - high field limit of the zero region in the first row

SIGF1 - low field limit of the zero region in the last row

SIGF2 - high field limit of the zero region in the last row

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

*proc2* - F1 processing parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

*proc2s* - F1 processing status parameters

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

ZERT2

ZERT1

**SEE ALSO**

*abs2, abst2 commanda [▷ 95]*, *abs1, abst1 commanda [▷ 97]*

# 5 3D Processing Commands

This chapter describes all TopSpin 3D processing commands. They only work on 3D data and store their output in processed data files. 3D raw data are never overwritten.

We will often refer to the three directions of a 3D dataset as the F3, F2 and F1 direction. F3 is always the acquisition direction. For processed data, F2 and F1 are always the second and third direction, respectively. For raw data, this order can be the same or reversed as expressed by the acquisition status parameter AQSEQ. 3D processing commands which work on raw data automatically determine their storage order from AQSEQ.

The name of a 3D processing command expresses the direction in which it works, e.g. **tf3** works in F3, **tf2** in F2 and **tf1** in the F1 direction. The command **r12** reads an F1-F2 plane, **r13** reads an F1-F3 plane etc.

For each command, the relevant input and output parameters are mentioned.

Furthermore, the relevant input and output files and their location are mentioned. Although file handling is completely transparent, it is sometimes useful to know which files are involved and where they reside. For example, if you have permission problems or if you want to process or interpret your data with third party software.

## 5.1 ft3d

### NAME

ft3d - Process data, including FT, in the F3, F2 and F1 direction (3D)

### DESCRIPTION

The command **ft3d** processes a 3D dataset in all three directions F3, F2 and F1. It is equivalent to the command sequence **tf3**-**tf2**-**tf1** or **tf3**-**tf1**-**tf2** (see below).

**ft3d** performs a Fourier transform which transforms time domain data (FID) into frequency domain data (spectrum). Depending on the processing parameters BC_mod, WDW, ME_mod and PH_mod, it also performs baseline correction, window multiplication, linear prediction and spectrum phase correction.

**ft3d** executes the following processing steps:

1. Baseline correction

   The time domain data are baseline corrected according to BC_mod. This parameter takes the value *no*, *single*, *quad*, *spol*, *qpol sfil* or *qfil*.

2. Linear prediction

   Linear prediction is done according to ME_mod. This parameter takes the value *no*, *LPfr*, *LPfc*, *LPbr*, *LPbc, LPmifr* or *LPmifc.* Usually, ME_mod = no, which means no prediction is done. Forward prediction (*LPfr*, *LPfc, LPmifr* or *LPmifc*) can, for example, be used to extend truncated FIDs. Backward prediction (*LPbr* or *LPbc*) is usually only done in F3, e.g. improve the initial data points of the FID. Linear prediction is only performed if NCOEF > 0. Furthermore, the parameters LPBIN and, for backward prediction, TDoff are evaluated.

3. Window multiplication

   The time domain data are multiplied with a window function according to WDW. This parameter takes the value *em*, *gm*, *sine*, *qsine*, *trap*, *user*, *sinc*, *qsinc*, *traf* or *trafs*.

4. Fourier transform

   The time domain data are Fourier transformed in F3 according to the acquisition status parameter AQ_mod (see AQ_mod table below).

In F2 and F1, they are Fourier transformed according to the acquisition status parameter FnMODE (if FnMODE = undefined, ft3d evaluates the processing parameter MC2).

The Fourier transform mode is stored in the processing status parameter FT_mod. Note that **ft3d** does not evaluate the processing parameter FT_mod!

5. Phase correction

   The frequency domain data are phase corrected according to PH_mod. This parameter takes the value *no*, *pk*, *mc* or *ps*. For PH_mod = pk, **ft3d** applies the values of PHC0 and PHC1. This is only useful if the phase values are known. You can determine them by typing **xfb** on the 3D data to process a 23 or 13 plane, do a phase correction on the resulting the 2D dataset and store the phase values to 3D.

| status AQ_mod | Fourier transform mode | status FT_mod |
|---|---|---|
| qf | forward, single, real | fsr |
| qsim | forward, quad, complex | fqc |
| qseq | forward, quad, real | fqr |
| DQD | forward, quad, complex | fqc |
| | | |
| FnMODE | Fourier transform mode | status FT_mod |
| undefined | according to MC2 | |
| QF | forward, quad, real | fqc |
| QSEQ | forward, quad, real | fqr |
| TPPI | forward, single, real | fsr |
| States | forward, quad, complex | fqc |
| States-TPPI | forward, single, complex | fsc |
| Echo-AntiEcho | forward, quad, complex | fqc |

The size of the processed data is determined by the processing parameter SI; SI real and SI imaginary points are created. A typical value for SI is TD/2 in which case, all raw data points are used and no zero filling is done. In fact, several parameters control the number of input and output data points, for example:

- SI > TD/2: the raw data are zero filled before the Fourier transform
- SI < TD/2: only the first 2*SI raw data points are used
- 0 < TDeff < TD: only the first TDeff raw data points are used
- 0 < TDoff < TD: the first TDoff raw data points are cut off and TDoff zeroes are appended at the end
- TDoff < 0: -TDoff zeroes are prepended at the beginning. Note that:
  - for SI < (TD-TDoff)/2 raw data are cut off at the end
  - for DIGMOD=digital, the zeroes would be prepended to the group delay which does not make sense. You can avoid that by converting the raw data with **convdta** before you process them.
- 0 < STSR < SI: only the processed data between STSR and STSR+STSI are stored (if STSI = 0, STSR is ignored and SI points are stored)
- 0 < STSI < SI: only the processed data between STSR and STSR+STSI are stored.

Note that only in the first case the processed data contain the total information of the raw data. In all other cases, information is lost. Before you run **ft3d**, you must set the processing parameter SI in all three directions F3, F2 and F1.

**ft3d** evaluates the acquisition status parameter AQSEQ, which defines the storage order of the raw data. Raw data can be stored in the order 3-2-1 or 3-1-2. Processed data, however, are always stored in the order 3-2-1. For AQSEQ=321, **ft3d** is equivalent to the command sequence **tf3**-**tf2**-**tf1**. For AQSEQ=312, it is equivalent to **tf3**-**tf1**-**tf2**. Note, however, that for magnitude or power data, the processing order is independent of AQSEQ. **ft3d** then behave as follows:

- for F1-PH_mod = mc / ps, **tf3**-**tf2**-**tf1** is executed
- for F2-PH_mod = mc / ps, **tf3**-**tf1**-**tf2** is executed

Note that PH_mod = mc/ps is only allowed in either F2 or F1, not in both and also not in F3.

**ft3d** evaluates the processing parameter FCOR. The first point of the FIDs is multiplied with the value of FCOR which lies between 0.0 and 2.0. For digitally filtered Avance data, FCOR has no effect in F3 because the first point is part of the group delay and, as such, is zero. In that case, it only plays a role in the F2 and F1 direction. However, on A*X data or Avance data measured with DIGMOD = analog, there is no group delay and FCOR also plays a role in F3.

**ft3d** evaluates the processing parameter PKNL. On A*X spectrometers, PKNL = true causes a non linear 5th order phase correction of the raw data. This corrects possible errors caused by non linear behaviour of the analog filters. On Avance spectrometers, PKNL must always be set to TRUE. For digitally filtered data, it causes **ft3d** to handle the group delay of the FID. For analog data it has no effect.

**ft3d** evaluates the processing parameter REVERSE. If REVERSE = TRUE, the spectrum will be reversed, i.e. the first data point becomes the last and the last data point becomes the first.

**ft3d** can be used with the following command line arguments:

**n**

**ft3d** will not store the imaginary data. Imaginary data are only needed for phase correction in last processed direction. If the phase values are already known and PHC0 and PHC1 have been set accordingly, **ft3d** will perform phase correction and there is no need to store the imaginary data. This will save processing time and disk space. If you still need to do a phase correction after **ft3d**, you can create imaginary data from the real data with a Hilbert transform (see **tht1**). Note that if the **n** option is omitted, imaginary data are only stored in the last processed direction.

**21** or **12**

**ft3d 21** is equivalent to the command sequence **tf3**-**tf2**-**tf1**, whereas **ft3d 12** is equivalent to **tf3**-**tf1**-**tf2**.

**xdim**

3D spectra are stored in the so-called subcube format. The size of the subcubes is calculated by **ft3d** and depends on the size of the spectrum and the available memory. The option **xdim** allows to use predefined subcube sizes. It causes **ft3d** to interpret the F3, F2 and F1 processing parameter XDIM which can be set by entering **xdim** on the command line. Note that XDIM = 0, is evaluated as XDIM = SI. The actually used subcube sizes, whether predefined or calculated, are stored as the F3, F2 and F1 processing status parameter XDIM and can be viewed with **dpp**. Predefining subcube sizes is, for example, used to read the processed data with third party software which can not interpret the processing status parameter XDIM.

**big/little**

**ft3d** stores the data in the data storage order of the computer it runs on, e.g. little endian on Windows PCs. Note that TopSpin's predecessor XWIN-NMR on SGI UNIX workstations stores data in big endian.The storage order is stored in the processing status parameter BYTORDP (type **s bytordp**). If, however, you want to read the processed data with third party software which can not interpret this parameter, you can use the **big/little** option to predefine the storage order.

**p<du>**

The option **p** allows to store the processed data on a different top level data directory, typically a different disk. The rest of the data directory path is the same as that of the raw data. If the specified top level directory does not exist, it will be created.

Normally, **ft3d** stores the entire spectral region as determined by the spectral width. However, you can do a so-called strip transform which means that only a certain region of the spectrum is stored. This can be done by setting the parameters STSR and STSI which represent the strip start and strip size, respectively. They both can take a value between 0 and SI. The values which are actually used can be a little different. STSI is always rounded to the next higher multiple of 16. Furthermore, when the data are stored in subcube format (see below), STSI is rounded to the next multiple of the subcube size. Type **dpp** to check this; if XDIM is smaller than SI, then the data are stored in subcube format and STSI is a multiple of XDIM.

**ft3d** stores the data in subcube format. It automatically calculates the subcube sizes such that one row (F3) of subcubes fits in the available memory. Furthermore, one column (F2) and one tube (F1) of subcubes must fit in the available memory. The calculated subcube sizes are stored in the processing status parameter XDIM (type **dpp**). The alignment of the data points subcube format is the extension of the alignment in a 2D dataset as it is shown in *Figure 4.2 [▷ 143]*. The storage handling is completely transparent to the user and is only of interest when the data are interpreted by third party software.

## INPUT PARAMETERS

### F3, F2 and F1 parameters

Set by the acquisition, can be viewed with **dpa** or **s td**:

TD - time domain; number of raw data points

Set by the user with **edp** or by typing **si**, **stsr** etc.:

SI - size of the processed data

STSR - strip start: first output point of strip transform

STSI - number of output points of strip transform

TDeff - number of raw data points to be used for processing

TDoff - first point of the FID used for processing (default 0)

BC_mod - FID baseline correction mode

BCFW - filter width for BC_mod = sfil or qfil

COROFFS - correction offset for BC_mod = spol/qpol or sfil/qfil

ME_mod - FID linear prediction mode

NCOEF - number of linear prediction coefficients

LPBIN - number of points for linear prediction

TDoff - number of raw data points predicted for ME_mod = LPb*

WDW - FID window multiplication mode

LB - Lorentzian broadening factor for WDW = em or gm

GB - Gaussian broadening factor for WDW = gm, sinc or qsinc

SSB - Sine bell shift for WDW = sine, qsine, sinc or qsinc

TM1, TM2 - limits of the trapezoidal window for WDW = trap

PH_mod - phase correction mode

PHC0 - zero order phase correction value for PH_mod = pk

PHC1 - first order phase correction value for PH_mod = pk

FCOR - first (FID) data point multiplication factor (0.0-2.0, default 0.5)

REVERSE - flag indicating to reverse the spectrum

**F3 parameters**

Set by the acquisition, can be viewed with **dpa** or **s aq_mod** etc.:
AQ_mod - acquisition mode (determines the status FT_mod)
AQSEQ - acquisition sequence (3-2-1 or 3-1-2)
BYTORDA - byteorder or the raw data
NC - normalization constant

**F2 and F1 parameters**

Set by the acquisition, can be viewed with **dpa** or by typing **s fnmode**:
FnMODE - Fourier transform mode

**OUTPUT PARAMETERS**

**F3, F2 and F1 parameters**

Can be viewed with **dpp** or by typing **s si**, **s stsr** etc.:
SI - size of the processed data
STSR - strip start: first output point of strip transform
STSI - strip size: number of output points of strip transform
FTSIZE - Fourier transform size
TDeff - number of raw data points that were used for processing
TDoff - first point of the FID used for processing (default 0)
XDIM - subcube size
FT_mod - Fourier transform mode

**F3 parameters**

Can be viewed with **dpp** or by typing **s ymax_p** etc.:
YMAX_p - maximum intensity of the processed data
YMIN_p - minimum intensity of the processed data
S_DEV - standard deviation of the processed data
NC_proc - intensity scaling factor
BYTORDP - byte order of the processed data

**F2 and F1 parameters**

Can be viewed with **dpp** or by typing **s mc2**:
MC2 - Fourier transform mode

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*
*ser* - raw data
*acqus* - F3 acquisition status parameters
*acqu2s* - F2 acquisition status parameters
*acqu3s* - F1 acquisition status parameters
*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*
*proc* - F3 processing parameters

*proc2* - F2 processing parameters

*proc3* - F1 processing parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3rrr* - real processed 3D data

*3rri* - real/imaginary processed data (for AQSEQ =321, FnMODE ≠ QF)

*3rir* - real/imaginary processed data (for AQSEQ =312, FnMODE ≠ QF)

*3iii* - imaginary processed data (for FnMODE = QF)

*procs* - F3 processing status parameters

*proc2s* - F2 processing status parameters

*proc3s* - F1 processing status parameters

*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

FT3D

**SEE ALSO**

*tf3* [▷ 179], *tf2* [▷ 176], *tf1* [▷ 173]

## 5.2 projplp, projpln, sumpl

**NAME**

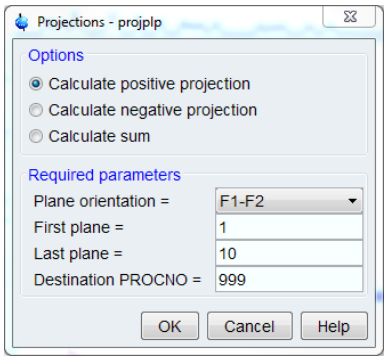
projplp - Calculate positive projection (nD)

projpln - Calculate negative projection (nD)

sumpl - Calculate sum projection (nD)

**DESCRIPTION**

The commands **projplp**, **projpln** and **sumpl** calculate the 2D positive, negative and sum projection, respectively. When entered without arguments, they all open the same dialog:




Here you can select the desired command in the **Options** section and specify the plane orientation, first and last row/column and output PROCNO in the Parameter section.

The parameters can also be specified as arguments. Up to 5 arguments can be used:

**<plane orientation>**

*23, 13, 12 (3D data)*

*34, 24, 14, 23, 13, 12, 43, ..., 21 (4D data)*

**<first plane>**

The plane included in the calculation

**<last plane>**

The last plane included in the calculation

**<dest. procno>**

The *procno* where the 2D output data are stored

**n**

Prevents the destination dataset from being displayed/activated (optional)

Here is an example:

**projplp 13 10 128 998 n**

Calculates the positive F1-F3 projection of the planes 10 to 128 along F2 and stores it under PROCNO 998.

Instead of specifying the first and last plane, you can also use the argument **all** for all cubes. For example:

**projplp 23 all 10**

Calculates the positive F2-F3 projection of all planes along F1 and stores it under PROCNO 10.

**projplp**, **projpln** and **sumpl** work on data of dimension ≥3D. On 4D and 5D data, the dialog shown in the figure above does not appear. Instead, the arguments are prompted for one at a time, if they are not specified on the command line.

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3rrr* - real processed 3D data

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

### SEE ALSO

*rpl* [▶ 199], *wpl* [▶ 205], *rser2d* [▶ 171]

## 5.3 r12, r13, r23, slice

### NAME

r12 - Read F1-F2 plane from 3D data and store as 2D data

r13 - Read F1-F3 plane from 3D data and store as 2D data

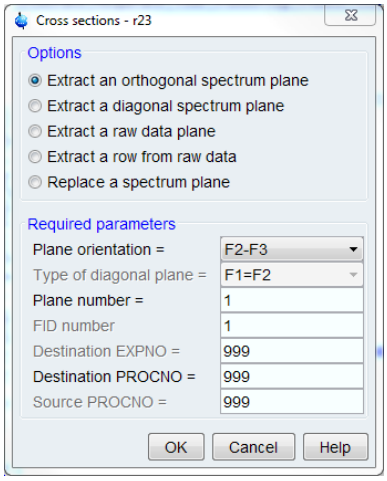
r23 - Read F2-F3 plane from 3D data and store as 2D data

slice - Open the read plane dialog box (2D, 3D)

### DESCRIPTION

The commands **r12**, **r13** and **r23** read a plane from 3D processed data and store it as a 2D data set.

When entered without arguments, they open the dialog box shown:




This dialog box offers several options, each of which selects a certain command for execution. Furthermore, you must specify three parameters:

- *Plane orientation*: F1-F2, F1-F3 or F2-F3. This parameter determines which of the commands **r12**, **r13** or **r23** is executed.
- *Plane number*: The maximum plane number is the SI value in the direction orthogonal to the plane orientation.
- *Destination procno*: The *procno* where the output 2D dataset is stored.

For each option described below, a table shows how the processing state of the output 2D data relates to the processing state of the input 3D data. This table can be interpreted as follows:

- *FID*: Data have not been Fourier transformed (time domain data)
- *Real*:- Data have been Fourier transformed but imaginary data do not exist
- *real+imag*: Data have been Fourier transformed and imaginary data exist

Depending on the processing state, an extracted plane can be further processed with 2D processing commands like **xf2**, **xf1**, **xf2p** etc.

### Extract an orthogonal spectrum plane in F1-F2

This option selects the command **r12** for execution. It reads an F1-F2 plane from a 3D data set and stores it as a 2D data set:

| 3D data processed with | 3D input data | | | 2D output data | |
|---|---|---|---|---|---|
| | F3 | F2 | F1 | F2 | F1 |
| tf3 | real+imag | FID | FID | FID | FID |
| tf3, tf2 | real | real+imag | FID | real+imag | FID |
| tf3, tf2, tf1 | real | real | real+imag | real | real+imag |
| tf3, tf1, tf2 | real | real+imag | real | real+imag | real |
| **r12** input/output data | | | | | |

### Extract an orthogonal spectrum plane in F1-F3

This option selects the command **r13** for execution. It reads an F1-F3 plane from a 3D data set and stores it as a 2D data set:

| 3D data processed with | 3D input data | | | 2D output data | |
|---|---|---|---|---|---|
| | F3 | F2 | F1 | F2 | F1 |
| tf3 | real+imag | FID | FID | real+imag | FID |
| tf3, tf2 | real | real+imag | FID | real | FID |
| tf3, tf2, tf1 | real | real | real+imag | real | real+imag |
| tf3, tf1, tf2 | real | real+imag | real | real | real |
| **r13** input/output data | | | | | |

**Extract an orthogonal spectrum plane in F2-F3**

This option selects the command **r23** for execution. It reads an F2-F3 plane from a 3D data set and stores it as a 2D data set:

| 3D data processed with | 3D input data | | | 2D output data | |
|---|---|---|---|---|---|
| | F3 | F2 | F1 | F2 | F1 |
| tf3 | real+imag | FID | FID | real+imag | FID |
| tf3, tf2 | real | real+imag | FID | real | real+imag |
| tf3, tf2, tf1 | real | real | real+imag | real | real |
| tf3, tf1, tf2 | real | real+imag | real | real | real+imag |
| **r23** input/output data | | | | | |

The parameters required by **r12**, **r13** and **r23** can also be entered as arguments on the command line. In that case, the command is executed without opening the dialog box. For example, **r12 10 999** reads an F1-F2 plane number 10 and stores it in *procno* 999. Note that the Plane orientation is not specified as an argument but part of the command name.

The commands **r12**, **r13** and **r23** are equivalent to the commands **rpl 12**, **rpl 13** and **rpl 23**, respectively (see the description of **rpl**).

## INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3rrr*, *3irr*, *3rir*, *3rri*, *3iii* - processed 3D data

## OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *2ir*, *2ri*, *2ii* - processed 2D data

*auditp.txt* - processing audit trail

## USAGE IN AU PROGRAMS

R12(plane, procno). For example R12(64, 1)

R13(plane, procno). For example R13(64, 1)

R23(plane, procno). For example R23(64, 1)

## SEE ALSO

*r12, r13, r23, slice* [▸ 167], *r12d, r13d, r23d* [▸ 170], , *rpl* [▸ 199], *wpl* [▸ 205]

## 5.4 r12d, r13d, r23d

### NAME

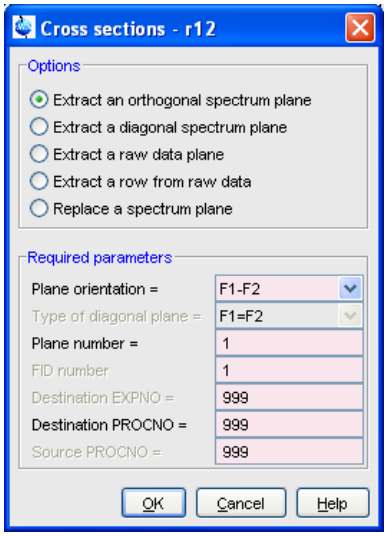
r12d - Read diagonal F1=F2 plane and store as 2D data (3D)

r13d - Read diagonal F1=F3 plane and store as 2D data (3D)

r23d - Read diagonal F2=F3 plane and store as 2D data (3D)

### DESCRIPTION

Read plane commands can be started from the command line or from the read plane dialog box. The latter is opened with the command **slice**.




This dialog box offers several options, each of which selects a certain command for execution.

### Extract a diagonal spectrum plane in F1-F2

This option selects the command **r12d** for execution. It reads the diagonal F1=F2 plane from a 3D data set and stores it as a 2D data set.

### Extract a diagonal spectrum plane in F1-F3

This option selects the command **r13d** for execution. It reads the diagonal F1=F3 plane from a 3D data set and stores it as a 2D data set.

### Extract a diagonal spectrum plane in F2-F3

This option selects the command **r23d** for execution. It reads the diagonal F2=F3 plane from a 3D data set and stores it as a 2D data set.

For each option, you must specify the destination *procno*.

**r12d**, **r13d** and **r23d** only store the real data.

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3rrr* - real processed 3D data

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

### SEE ALSO

*r12, r13* [▶ 167], *rpl* [▶ 199], *wpl* [▶ 205]

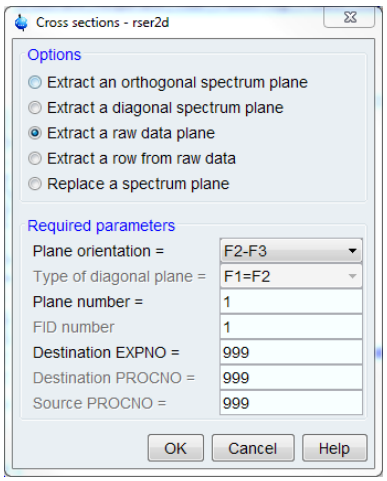
## 5.5 rser2d

### NAME

rser2d - Read plane from raw 3D data and store as a 2D (3D).

### DESCRIPTION

The command **rser2d** reads a plane from 3D raw data (a series of FIDs) and stores it as a pseudo raw 2D data set. When entered without arguments, it opens the following dialog box:




Here you can specify three required parameters:

- *Plane orientation*: F1-F3 or F2-F3 (must contain acquisition (F3) direction)
- *Plane number*: the maximum plane number is the TD value in the direction orthogonal to the plane orientation
- *Destination EXPNO*: the *expno* where the output 2D dataset is stored

The parameters can also be entered as arguments on the command line. In that case, the command is executed without opening the dialog box. For example, **rser2d s23 10 999** reads an F3-F2 plane number 10 and stores it in *expno* 999

In contrast to **rser**, **rser2d** can only be entered on the source dataset, not on the destination dataset.

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*ser* - 3D raw data

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*ser* - 2D pseudo raw data

*audita.txt* - acquisition audit trail

*<dir>/data/<user>/nmr/<name>/<expno2>/pdata/1/*

*used_from* - data path of the source 3D data and the plane number

### USAGE IN AU PROGRAMS

RSER2D (direction, plane, expno)

### SEE ALSO

*wser* [▷ 127], *wserp* [▷ 128], *rpl* [▷ 199], *wpl* [▷ 205]

## 5.6 tabs3, tabs2, tabs1

### NAME

tabs3 - Automatic baseline correction in F3 (3D)

tabs2 - Automatic baseline correction in F2 (3D)

tabs1 - Automatic baseline correction in F1 (3D)

### DESCRIPTION

**tabs3** performs an automatic baseline correction in the F3 direction, by subtracting a polynomial. The degree of the polynomial is determined by the F3 parameter ABSG which has a value between 0 and 5, with a default of 5. **tabs3** works like **absf** in 1D and **abs2** in 2D. This means that it only corrects a certain spectral region which is determined by the parameters ABSF1 and ABSF2.

**tabs2** works like **tabs3**, except that corrects data in the F2 direction using the F2 parameters ABSG, ABSF2 and ABSF1.

**tabs1** works like **tabs3**, except that corrects data in the F1 direction using the F1 parameters ABSG, ABSF2 and ABSF1.

### INPUT PARAMETERS

#### F3 parameters

Set by the user with **edp** or by typing **absg**:

ABSG - degree of the polynomial to be subtracted (0 to 5, default of 5)

#### F3, F2 and F1 parameters

Set by the user with **edp** or by typing **absf1**, **absf2**:

ABSF1- low field limit of the correction region

ABSF2 - high field limit of the correction region

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3rrr* - real processed 3D data

*proc* - F3 processing parameters

*proc2* - F2 processing parameters

*proc3* - F1 processing parameters

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3rrr* - real processed 3D data

*procs* - F3 processing status parameters

*proc2s* - F2 processing status parameters

*proc3s* - F1 processing status parameters

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

TABS3

TABS2

TABS1

### SEE ALSO

*abs, absf, absd, bas* [ 42], *abs1, abst1, absd1, absot1, bas* [ 97], *abs2, abst2, absd2, absot2* [ 95]

## 5.7 tf1

### NAME

tf1 - Process data, including FT, in F2 (3D)

### DESCRIPTION

The command **tf1** processes a 3D dataset in the F1 direction. This involves a Fourier transform which transforms time domain data (FID) into frequency domain data (spectrum). Depending on the processing parameters BC_mod, WDW, ME_mod and PH_mod, **tf1** also performs baseline correction, window multiplication, linear prediction and spectrum phase correction.

The processing steps done by **tf1** can be described as follows:

**tf1** only works on data which have already been processed with **tf3** and possibly with **tf2**. It performs the following processing steps:

1. Baseline correction of the F1 time domain data
2. Each tube is baseline corrected according to BC_mod. This parameter takes the value *no*, *single*, *quad*, *spol*, *qpol sfil* or *qfil*. More details on BC_mod can be found in chapter *List of processing parameters* [ 20].
3. Linear prediction of the F1 time domain data
4. Linear prediction is done according to ME_mod. This parameter takes the value *no*, *LPfr*, *LPfc*, *LPbr*, *LPbc, LPmifr, LPmifc.* Usually, ME_mod = no, which means no prediction is done. Forward prediction in F1 (*LPfr*, *LPfc*, *LPmifr* or *LPmifc*) can, for example, be used to extend truncated FIDs. Backward prediction (*LPbr* or *LPbc*) is not used very often in F1. Linear prediction is only performed for NCOEF > 0. Furthermore, LPBIN and, for backward prediction, TDoff play a role (see these parameters in chapter *List of processing parameters* [ 20]).
5. Window multiplication of the F1 time domain data
6. Each tube is multiplied with a window function according to WDW. This parameter takes the value *em*, *gm*, *sine*, *qsine*, *trap*, *user*, *sinc*, *qsinc*, *traf* or *trafs*. More details on WDW can be found in chapter *List of processing parameters* [ 20].

7. Fourier transform of the F1 time domain data. Each tube is Fourier transformed according to the F1 processing status parameter MC2. **tf1** does not evaluate the processing parameter FT_mod! Instead, it evaluates the F1 processing status parameter MC2, which was set by **tf3** to the value of the F1 acquisition status parameter FnMODE (if FnMODE = undefined, tf3 sets processing status MC2 to processing MC2). **tf1** stores the corresponding Fourier transform mode as the processing status parameter FT_mod (type **dpp**).
8. Phase correction of the F1 frequency domain data.
9. Each column is phase corrected according to PH_mod. This parameter takes the value *no*, *pk*, *mc* or *ps*. For PH_mod = pk, **tf1** applies the values of PHC0 and PHC1. This is only useful if the phase values are known. You can determine them by typing **xfb** on the 3D data to process a 13 or 12 plane, do a phase correction on the resulting the 2D dataset and store the phase values to 3D. More details on PH_mod can be found in chapter *List of processing parameters* [▷ 20].

| F1 MC2 | Fourier transform mode | status FT_mod |
|---|---|---|
| QF | forward, quad, real | fqc |
| QSEQ | forward, quad, real | fqr |
| TPPI | forward, single, real | fsr |
| States | forward, quad, complex | fqc |
| States-TPPI | forward, single, complex | fsc |
| Echo-AntiEcho | forward, quad, complex | fqc |

The F1 processing parameter SI determines the size of the processed data in the F1 direction. This must, however, be set before **tf3** is done and cannot be changed after **tf3**. See **tf3** for the role of TD, TDeff and TDoff.

**tf1** can do a strip transform according to the F1 parameters STSR and STSI (see **tf3**).

**tf1** evaluates the F1 parameter FCOR. The first point of the FIDs is multiplied with the value of FCOR which is a value between 0.0 and 2.0. As such, FCOR allows to control the DC offset of the spectrum.

**tf1** evaluates the F1 parameter REVERSE. If REVERSE=TRUE, the spectrum will be reversed in F1, i.e. the first data point becomes the last and the last data point becomes the first.

**tf1** evaluates the F1 status parameter MC2. For MC2 ≠ QF, **tf1** uses the file *3rrr* as input and the files *3rrr* and *3rri* as output. For MC2 = QF, **tf1** uses the files *3rrr* and *3iii* as input and output. The role of MC2 is described in detail for the 2D processing command **xfb**.

## INPUT PARAMETERS

### F1 parameters

Set by the user with **edp** or by typing **bc_mod**, **bcfw** etc.:

BC_mod - FID baseline correction mode

BCFW - filter width for BC_mod = sfil or qfil

COROFFS - correction offset for BC_mod = spol/qpol or sfil/qfil

ME_mod - FID linear prediction mode

NCOEF - number of linear prediction coefficients

LPBIN - number of points for linear prediction

TDoff - number of raw data points predicted for ME_mod = LPb*

WDW - FID window multiplication mode

LB - Lorentzian broadening factor for WDW = em or gm
GB - Gaussian broadening factor for WDW = gm, sinc or qsinc
SSB - Sine bell shift for WDW = sine, qsine, sinc or qsinc
TM1, TM2 - limits of the trapezoidal window for WDW = trap
PH_mod - phase correction mode
PHC0 - zero order phase correction value for PH_mod = pk
PHC1 - first order phase correction value for PH_mod = pk
FCOR - first (FID) data point multiplication factor (0.0-2.0, default 0.5)
REVERSE - flag indicating to reverse the spectrum

**F3, F2 and F1 parameters**

Set by **tf3**, can be viewed with **dpp** or by typing **s si**, **s stsi** etc.:
SI - size of the processed data
STSR - strip start: first output point of strip transform
STSI - strip size: number of output points of strip transform
TDeff - number of raw data points to be used for processing
TDoff - first point of the FID used for processing (default 0)

**F1 parameters**

Set by the **tf3**, can be viewed with **dpp** or by typing **s mc2** :
MC2 - Fourier transform mode

**OUTPUT PARAMETERS**

**F1 parameters**

can be viewed with **dpp** or by typing **s ft_mod** :
FT_mod - Fourier transform mode
FTSIZE - Fourier transform size

**F3 parameters**

Can be viewed with **dpp** or by typing **s ymax_p** etc.:
YMAX_p - maximum intensity of the processed data
YMIN_p - minimum intensity of the processed data
S_DEV - standard deviation of the processed data
NC_proc - intensity scaling factor

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*
*acqu3s* - F1 acquisition status parameters
*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*
*3rrr* - processed 3D data (Fourier transformed in F1)
*3iii* - real/imaginary processed data (if MC2 = QF)
*proc3* - F1 processing parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3rrr* - real processed 3D data

*3rir* - real/imaginary data (if MC2 ≠ QF)

*3iii* - real/imaginary processed data (if MC2 = QF)

*proc3s* - F1 processing status parameters

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

TF1(store_imag)

Where *store_image* can be *y* or *n*

### SEE ALSO

*tf3* [▸ 179], *tf2* [▸ 176], *ft3d* [▸ 161]

## 5.8 tf2

### NAME

tf2 - Process data, including FT, in F2 (3D)

### DESCRIPTION

The command **tf2** processes a 3D dataset in the F2 direction. This involves a Fourier transform which transforms time domain data (FID) into frequency domain data (spectrum). Depending on the processing parameters BC_mod, WDW, ME_mod and PH_mod, **tf2** also performs baseline correction, window multiplication, linear prediction and spectrum phase correction.

The processing steps done by **tf2** can be described as follows:

**tf2** only works on data which have already been processed with **tf3**. It performs the following processing steps in the F2 direction:

1. Baseline correction of the F2 time domain data
2. Each column is baseline corrected according to BC_mod. This parameter takes the value *no*, *single*, *quad*, *spol*, *qpol sfil* or *qfil*. More details on BC_mod can be found in chapter *List of processing parameters* [▸ 20].
3. Linear prediction of the F2 time domain data
4. Linear prediction is done according to ME_mod. This parameter takes the value *no*, *LPfr*, *LPfc*, *LPbr*, *LPbc, LPmifr or LPmifc.* Usually, ME_mod = no, which means no prediction is done. Forward prediction in F2 (*LPfr*, *LPfc*, *LPmifr* or *LPmifc*) can, for example, be used to extend truncated FIDs. Backward prediction (*LPbr* or *LPbc*) is not used very often in F2. Linear prediction is only performed for NCOEF > 0. Furthermore, LPBIN and, for backward prediction, TDoff play a role (see these parameters in chapter *List of processing parameters* [▸ 20]).
5. Window multiplication of the F2 time domain data
6. Each column is multiplied with a window function according to WDW. This parameter takes the value *em*, *gm*, *sine*, *qsine*, *trap*, *user*, *sinc*, *qsinc*, *traf* or *trafs*. More details on WDW can be found in chapter *List of processing parameters* [▸ 20].
7. Fourier transform of the F2 time domain data
8. **tf2** Fourier transforms each column according to the F2 processing status parameter MC2 and stores the corresponding Fourier transform mode in the processing status parameter FT_mod (see table below). The status MC2 has been set by the **tf3** command

to the value of the F2 acquisition status parameter FnMODE (if FnMODE = undefined, tf3 sets processing status MC2 to processing MC2). Note that **tf2** does not evaluate the processing parameter FT_mod!

9. Phase correction of the F2 frequency domain data.
10. Each column is phase corrected according to PH_mod. This parameter takes the value *no*, *pk*, *mc* or *ps*. For PH_mod = pk, **tf2** applies the values of PHC0 and PHC1. This is only useful if the phase values are known. You can determine them by typing **xfb** on the 3D data to process a 23 or 12 plane, do a phase correction on the resulting the 2D dataset and store the phase values to 3D. More details on PH_mod can be found in chapter *List of processing parameters [▷ 20]*.

| F2 status MC2 | Fourier transform mode | status FT_mod |
|---|---|---|
| QF | forward, quad, real | fqc |
| QSEQ | forward, quad, real | fqr |
| TPPI | forward, single, real | fsr |
| States | forward, quad, complex | fqc |
| States-TPPI | forward, single, complex | fsc |
| Echo-AntiEcho | forward, quad, complex | fqc |

The F2 processing parameter SI determines the size of the processed data in the F2 direction. This must, however, be set before **tf3** is done and cannot be changed after **tf3**. See **tf3** for the role of TD, TDeff and TDoff.

**tf2** can do a strip transform according to the F2 parameters STSR and STSI (see **tf3**).

**tf2** evaluates the F2 parameter FCOR. The first point of the FIDs is multiplied with the value of FCOR which is a value between 0.0 and 2.0. As such, FCOR allows to control the DC offset of the spectrum.

**tf2** evaluates the F2 parameter REVERSE. If REVERSE = TRUE, the spectrum will be reversed in F2, i.e. the first data point becomes the last and the last data point becomes the first.

**tf2** evaluates the F2 status parameter MC2. For MC2 ≠ QF, **tf2** uses the file *3rrr* as input and the files *3rrr* and *3rir* as output. For MC2 = QF, **tf2** uses the files *3rrr* and *3iii* as input and output. The role of MC2 is described in detail for the 2D processing command **xfb**.

## INPUT PARAMETERS

### F2 parameters

Set by the user with **edp** or by typing **bc_mod**, **bcfw** etc.:

BC_mod - FID baseline correction mode

BCFW - filter width for BC_mod = sfil or qfil

COROFFS - correction offset for BC_mod = spol/qpol or sfil/qfil

ME_mod - FID linear prediction mode

NCOEF - number of linear prediction coefficients

LPBIN - number of points for linear prediction

TDoff - number of raw data points predicted for ME_mod = LPb*

WDW - FID window multiplication mode

LB - Lorentzian broadening factor for WDW = em or gm

GB - Gaussian broadening factor for WDW = gm, sinc or qsinc

SSB - Sine bell shift for WDW = sine, qsine, sinc or qsinc

TM1, TM2 - limits of the trapezoidal window for WDW = trap
PH_mod - phase correction mode
PHC0 - zero order phase correction value for PH_mod = pk
PHC1 - first order phase correction value for PH_mod = pk
FCOR - first (FID) data point multiplication factor (0.0-2.0, default 0.5)
REVERSE - flag indicating to reverse the spectrum

**F3, F2 and F1 parameters**

Set by **tf3**, can be viewed with **dpp** or by typing **s si**, **s stsi** etc.:
SI - size of the processed data
STSR - strip start: first output point of strip transform
STSI - strip size: number of output points of strip transform
TDeff - number of raw data points to be used for processing
TDoff - first point of the FID used for processing (default 0)

**F2 parameters**

Set by the **tf3**, can be viewed with **dpp** or by typing **s mc2** :
MC2 - Fourier transform mode

**F1 parameters**

Set by the acquisition, can be viewed with **dpa** or by typing **s td** etc.:
TD - time domain; number of raw data points

**OUTPUT PARAMETERS**

**F2 parameters**

Can be viewed with **dpp** or by typing **s ft_mod** :
FT_mod - Fourier transform mode
FTSIZE - Fourier transform size

**F3 parameters**

Can be viewed with **dpp** or by typing **s ymax_p**, **s ymin_p** etc.:
YMAX_p - maximum intensity of the processed data
YMIN_p - minimum intensity of the processed data
S_DEV - standard deviation of the processed data
NC_proc - intensity scaling factor

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*
*acqu2s* - F2 acquisition status parameters
*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*
*3rrr* - processed 3D data (Fourier transformed in F3)
*3iii* - real/imaginary processed data (if MC2 = QF)
*proc2* - F2 processing parameters
*procs*, *proc2s*, *proc3s* - F3, F2, F1 processing status parameters

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3rrr* - real processed 3D data

*3rir* - real/imaginary data (if MC2 ≠ QF)

*3iii* - real/imaginary processed data (if MC2 = QF)

*procs* - F3 processing status parameters

*proc2s* - F2 processing status parameters

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

TF2(store_imag)

where *store_image* can be *y* or *n*

### SEE ALSO

*tf3* [▷ 179], *tf1* [▷ 173], *ft3d* [▷ 161]

## 5.9 tf3

### NAME

tf3 - Process data, including FT, in F3 (3D)

### DESCRIPTION

The command **tf3** processes a 3D dataset in the F3 direction. F3 is the first direction of a 3D dataset, i.e. the acquisition direction. **tf3** always performs a Fourier transform which transforms time domain data (FID) into frequency domain data (spectrum). Depending on the processing parameters BC_mod, WDW, ME_mod and PH_mod, it also performs baseline correction, window multiplication, linear prediction and spectrum phase correction.

The processing steps done by **tf3** can be described as follows:

1. Baseline correction of the F3 time domain data
2. Each row is baseline corrected according to BC_mod. This parameter takes the value *no*, *single*, *quad*, *spol*, *qpol sfil* or *qfil*. More details on BC_mod can be found in chapter *List of processing parameters* [▷ 20].
3. Linear prediction of the F3 time domain data
4. Linear prediction is done according to ME_mod. This parameter takes the value *no*, *LPfr*, *LPfc*, *LPbr*, *LPbc, LPmifr* or *LPmifc.* Usually, ME_mod = no, which means no prediction is done. Forward prediction (*LPfr*, *LPfc, LPmifr* or *LPmifc*) can, for example, be used to extend truncated FIDs. Backward prediction (*LPbr* or *LPbc*) can be used to improve the initial data points of the FID. Linear prediction is only performed if NCOEF > 0. Furthermore, the parameters LPBIN and, for backward prediction, TDoff play a role (see these parameters in chapter *List of processing parameters* [▷ 20]).
5. Window multiplication of the F3 time domain data
6. Each row is multiplied with a window function according to WDW. This parameter takes the value *em*, *gm*, *sine*, *qsine*, *trap*, *user*, *sinc*, *qsinc*, *traf* or *trafs*. More details on WDW can be found in chapter *List of processing parameters* [▷ 20].
7. Fourier transform of the F3 time domain data
8. Each row is Fourier transformed according to the acquisition status parameter AQ_mod as shown in the table below. **tf3** does not evaluate the processing parameter FT_mod! However, it stores the Fourier transform mode in the processing status parameter FT_mod.

9. Phase correction of the F3 frequency domain data
10. Each row is phase corrected according to PH_mod. This parameter takes the value *no*, *pk*, *mc* or *ps*. For PH_mod = pk, **tf3** applies the values of PHC0 and PHC1. This is only useful if the phase values are known. You can determine them by typing **xfb** on the 3D data to process a 23 or 13 plane, do a phase correction on the resulting the 2D dataset and store the phase values to 3D. More details on PH_mod can be found in chapter *List of processing parameters [▷ 20]*.

| AQ_mod | Fourier transform mode | status FT_mod |
|---|---|---|
| qf | forward, single, real | fsr |
| qsim | forward, quad, complex | fqc |
| qseq | forward, quad, real | fqr |
| DQD | forward, quad, complex | fqc |

The size of the processed data is determined by the processing parameter SI; SI real and SI imaginary points are created. A typical value for SI is TD/2 in which case, all raw data points are used and no zero filling is done. In fact, several parameters control the number of input and output data points, for example:

- SI > TD/2: the raw data are zero filled before the Fourier transform
- SI < TD/2: only the first 2*SI raw data points are used
- 0 < TDeff < TD: only the first TDeff raw data points are used
- 0 < TDoff < TD: the first TDoff raw data points are cut off and TDoff zeroes are appended at the end
- TDoff < 0: -TDoff zeroes are prepended at the beginning. Note that:
  - for SI < (TD-TDoff)/2 raw data are cut off at the end
  - for DIGMOD=digital, the zeroes would be prepended to the group delay which does not make sense. You can avoid that by converting the raw data with **convdta** before you process them.
- 0 < STSR < SI: only the processed data between STSR and STSR+STSI are stored (if STSI = 0, STSR is ignored and SI points are stored)
- 0 < STSI < SI: only the processed data between STSR and STSR+STSI are stored.

Note that only in the first case the processed data contain the total information of the raw data. In all other cases, information is lost.

Before you run **tf3**, you must set the processing parameter SI in all three directions F3, F2 and F1. The command**tf2** does not evaluate the F2 processing parameter SI, it evaluates the processing status parameter SI as it was set by **tf3**.

**tf3** evaluates the acquisition status parameter AQSEQ. This parameter defines the storage order of the raw data 3-2-1 or 3-1-2. For processed data, F2 and F1 are always the second and third direction, respectively. For raw data, this order can be the same or reversed as expressed by AQSEQ.

**tf3** evaluates the processing parameter FCOR. The first point of the FIDs is multiplied with the value of FCOR which lies between 0.0 and 2.0. For digitally filtered Avance data, FCOR has no effect in F3 because the first point is part of the group delay and, as such, is zero. In that case, it only plays a role in the F2 and F1 direction (see **tf2** and **tf1**). However, on A*X data or Avance data measured with DIGMOD = analog, there is no group delay and FCOR also plays a role in F3.

**tf3** evaluates the processing parameter PKNL. On A*X spectrometers, PKNL = true causes a non linear 5th order phase correction of the raw data. This corrects possible errors caused by non linear behaviour of the analog filters. On Avance spectrometers, PKNL must always be set to TRUE. For digitally filtered data, it causes **tf3** to handle the group delay of the FID. For analog data it has no effect.

**tf3** evaluates the processing parameter REVERSE. If REVERSE = TRUE, the spectrum will be reversed in F3, i.e. the first data point becomes the last and the last data point becomes the first.

**tf3** can be used with the following command line options:

**n**

**tf3** will not store the imaginary data. Imaginary data are only needed for phase correction. If the phase values are already known and PHC0 and PHC1 have been set accordingly, **tf3** will perform phase correction and there is no need to store the imaginary data. This will save processing time and disk space. If you still need to do a phase correction after **tf3**, you can create imaginary data from the real data with a Hilbert transform (see **tht3**).

**xdim**

3D spectra are stored in the so-called subcube format. The size of the subcubes is calculated by **tf3** and depends on the size of the spectrum and the available memory. The option **xdim** allows to use predefined subcube sizes. It causes **tf3** to interpret the F3, F2 and F1 processing parameter XDIM which can be set with the command **xdim**. The actually used subcube sizes, whether predefined or calculated, are stored as the F3, F2 and F1 processing status parameter XDIM and can be viewed with **dpp**. Predefining subcube sizes is, for example, used to read the processed data with third party software which cannot interpret the processing status parameter XDIM.

**big/little**

**tf3** stores the data in the data storage order of the computer it runs on, e.g. little endian on Windows PCs. Note that TopSpin's predecessor XWIN-NMR on SGI UNIX workstations stores data in big endian.The storage order is stored in the processing status parameter BYTORDP (type **s bytordp**). If, however, you want to read the processed data with third party software which can not interpret this parameter, you can use the **big/little** option to predefine the storage order.

Normally, **tf3** stores the entire spectral region as determined by the spectral width. However, you can do a so-called strip transform which means that only a certain region of the spectrum is stored. This can be done by setting the parameters STSR and STSI which represent the strip start and strip size, respectively. They both can take a value between 0 and SI. The values which are actually used can be a little different. STSI is always rounded to the next higher multiple of 16. Furthermore, when the data are stored in subcube format (see below), STSI is rounded to the next multiple of the subcube size. Type **dpp** to check this; if XDIM is smaller than SI, then the data are stored in subcube format and STSI is a multiple of XDIM.

**tf3** stores the data in subcube format. It automatically calculates the subcube sizes such that one row (F3) of subcubes fits in the available memory. Furthermore, one column (F2) and one tube (F1) of subcubes must fit in the available memory. The calculated subcube sizes are stored in the processing status parameter XDIM (type **dpp**). The alignment of the data points for sequential and subcube format is the extension of the alignment in a 2D dataset as it is shown in *Figure 4.1 [▷ 143]* and *Figure 4.2 [▷ 143]*. The storage handling is completely transparent to the user and is only of interest when the data are interpreted by third party software.

## INPUT PARAMETERS

### F3, F2 and F1 parameters

Set by the user with **edp** or by typing **si**, **stsr** etc.:

SI - size of the processed data

STSR - strip start: first output point of strip transform
STSI - number of output points of strip transform
TDeff - number of raw data points to be used for processing
TDoff - first point of the FID used for processing (default 0)

**F3 parameters**

Set by the user with **edp** or by typing **bc_mod**, **bcfw** etc.:
BC_mod - FID baseline correction mode
BCFW - filter width for BC_mod = sfil or qfil
COROFFS - correction offset for BC_mod = spol/qpol or sfil/qfil
ME_mod - FID linear prediction mode
NCOEF - number of linear prediction coefficients
LPBIN - number of points for linear prediction
TDoff - number of raw data points predicted for ME_mod = LPb*
WDW - FID window multiplication mode
LB - Lorentzian broadening factor for WDW = em or gm
GB - Gaussian broadening factor for WDW = gm, sinc or qsinc
SSB - Sine bell shift for WDW = sine, qsine, sinc or qsinc
TM1, TM2 - limits of the trapezoidal window for WDW = trap
PH_mod - phase correction mode
PHC0 - zero order phase correction value for PH_mod = pk
PHC1 - first order phase correction value for PH_mod = pk
FCOR - first (FID) data point multiplication factor (0.0-2.0, default 0.5)
REVERSE - flag indicating to reverse the spectrum
PKNL - group delay compensation (Avance) or filter correction (A*X)
Set by the acquisition, can be viewed with **dpa** or **s aq_mod** etc.:
AQ_mod - acquisition mode (determines the status FT_mod)
AQSEQ - acquisition sequence (3-2-1 or 3-1-2)
TD - time domain; number of raw data points
BYTORDA - byteorder or the raw data
NC - normalization constant

**F2 and F1 parameters**

Set by the acquisition, can be viewed with **dpa** or by typing **s fnmode** etc.:
FnMODE - Fourier transform mode

**OUTPUT PARAMETERS**

**F3, F2 and F1**

Can be viewed with **dpp** or by typing **s si**, **s stsi** etc.:
SI - size of the processed data
STSR - strip start: first output point of strip transform
STSI - strip size: number of output points of strip transform
TDeff - number of raw data points that were used for processing

TDoff - first point of the FID used for processing (default 0)
XDIM - subcube size

**F3 parameters**

Can be viewed with **dpp** or by typing **s si**, **s tdeff** etc.:
FTSIZE - Fourier transform size
FT_mod - Fourier transform mode
YMAX_p - maximum intensity of the processed data
YMIN_p - minimum intensity of the processed data
S_DEV - standard deviation of the processed data
NC_proc - intensity scaling factor
BYTORDP - byte order of the processed data

**F2 and F1 parameters**

Can be viewed with **dpp** or by typing **s mc2** etc.:
MC2 - Fourier transform mode

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*
*ser* - raw data
*acqus* - F3 acquisition status parameters
*acqu2s* - F2 acquisition status parameters
*acqu3s* - F1 acquisition status parameters
*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*
*proc* - F3 processing parameters
*proc2* - F2 processing parameters
*proc3* - F1 processing parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*
*3rrr* - real processed 3D data
*3irr* - real/imaginary processed data (for FnMODE ≠ QF)
*3iii* - real/imaginary processed data (for FnMODE = QF)
*procs* - F3 processing status parameters
*proc2s* - F2 processing status parameters
*proc3s* - F1 processing status parameters
*auditp.txt* - processing audit trail

**USAGE IN AU PROGRAMS**

TF3(store_imag, partition)
Where *store_image* can be *y* or *n* and *partition* is the top level data directory

**SEE ALSO**

*tf2 [▸ 176]*, *tf1 [▸ 173]*, *ft3d [▸ 161]*

## 5.10 tf3p, tf2p, tf1p

### NAME

tf3p - Phase correction in F3 (3D)

tf2p - Phase correction in F2 (3D)

tf1p - Phase correction in F1 (3D)

### DESCRIPTION

**tf3p** performs a phase correction in the F3 direction applying the values of PHC0 and PHC1. These values must first be determined, for example on a 2D plane. You can do that by typing **xfb** on the 3D data to process a 23 or 13 plane, do a phase correction on the resulting the 2D dataset and store the phase values to 3D.

**tf2p** works like **tf3p**, except that it works in the F2 direction applying the F2 parameters PHC0 and PHC1. These can be determined on a 2D plane extracted with **r23** or **r12**.

**tf1p** works like **tf3p**, except that it works in the F1 direction applying the F1 parameters PHC0 and PHC1. These can be determined on a 2D plane extracted with **r13** or **r12**.

**tf3p** can only be done:

- directly after **tf3** (not after **tf2** or **tf1**)
- if the F3 imaginary data exist

Note that the command **tf3 n** does not store the imaginary data. You can, however, create them data from the real data with a Hilbert transform (command **tht3**).

Phase correction is already done as a part of the commands **tf3**, **tf2** and **tf1**, if PH_mod = pk and PHC0 and PHC1 are set.

### INPUT PARAMETERS

Set by the user with **edp** or by typing **phc0**, **phc1** etc.

PHC0 - zero order phase correction value (frequency independent)

PHC1 - first order phase correction value (frequency dependent)

### OUTPUT PARAMETERS

Can be viewed with **dpp** or by typing **s phc0**, **s phc1** etc.:

PHC0 - zero order phase correction value (frequency independent)

PHC1 - first order phase correction value (frequency dependent)

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3rrr* - real processed 3D data

*3irr* - F3 imaginary processed data (input of **tf3p**)

*3rir* - F2 imaginary processed data (input of **tf2p**)

*3rri* - F1-imaginary processed data (input of **tf1p**)

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3rrr* - real processed 3D data

*3irr* - F3 imaginary processed data (output of **tf3p**)

*3rir* - F2 imaginary processed data (output of **tf2p**)

*3rri* - F1-imaginary processed data (output of **tf1p**)

*auditp.txt* - processing audit trail

### USAGE IN AU PROGRAMS

TF3P(store_imag), where *store_image* can be *y* or *n*

TF2P(store_imag), where *store_image* can be *y* or *n*

TF1P(store_imag), where *store_image* can be *y* or *n*

### SEE ALSO

*tf3* [▶ 179], *tf2* [▶ 176], *tf1* [▶ 173], *pk* [▶ 71], *xfbp, xf2p, xf1p* [▶ 150]

## 5.11 tht3, tht2, tht1

### NAME

tht3 - Hilbert transform in F3 (3D)

tht2 - Hilbert transform in F2 (3D)

tht1 - Hilbert transform in F1 (3D)

### DESCRIPTION

**tht3** performs a Hilbert transform in the F3 direction creating imaginary data from the real data. The resulting imaginary data can then be used for phase correction with **tf3p**.

**tht2** performs a Hilbert transform in the F2 direction creating imaginary data from the real data. The resulting imaginary data can then be used for phase correction with **tf2p**.

**tht1** performs a Hilbert transform in the F1 direction creating imaginary data from the real data. The resulting imaginary data can then be used for phase correction with **tf1p**.

Note that Hilbert Transform is only useful when the real data have been created from zero filled raw data, with SI ≥ TD.

Normally, the imaginary data are created during Fourier transform. If, however, the imaginary data are missing or do not match the real data and you want to do a phase correction, you can (re)create them with Hilbert transform. Imaginary data do not match the real data if the latter have been manipulated after the Fourier transform, for example by baseline correction or third party software.

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3rrr* - real processed 3D data

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3irr* - F3 imaginary processed data (output of **tht3**)

*3rir* - F2 imaginary processed data (output of **tht2**)

*3rri* - F1-imaginary processed data (output of **tht1**)

*auditp.txt* - processing audit trail

### SEE ALSO

*tf3* [▶ 179], *tf2* [▶ 176], *tf1* [▶ 173]

# 6 nD Processing Commands

TopSpin offers nD processing. Datasets up to 5D have been tested by Bruker. nD data can be displayed by reading cubes, planes or traces.

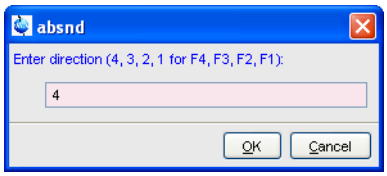
## 6.1 absnd

### NAME

absnd - nD automatic baseline correction

### DESCRIPTION

The command **absnd** performs an automatic baseline correction of data of dimension ≥3D. It takes one argument, the direction to be corrected. If no argument is specified on the command line, it is requested:




**absnd** subtracts a polynomial, the degree of which is determined by the parameter ABSG, which has a value between 0 and 5, with a default of 5. It only corrects a certain spectral region which is determined by the parameters ABSF1 and ABSF2.

**absnd** actually processes 2D planes of an nD data set, performing a series of **abs2** or **abs1** commands. On 3D data, the commands **absnd 3**, **absnd 2** and **absnd 1** are equivalent to **tabs3**, **tabs2** and **tabs1**, respectively.

### INPUT PARAMETERS

**Acquisition direction:**

Set by the user with **edp** or by typing **absg**.:

ABSG - degree of the polynomial to be subtracted (0 to 5, default of 5)

**All directions:**

Set by the user with **edp** or by typing **absf1**, **absf2**:

ABSF1- low field limit of the correction region

ABSF2 - high field limit of the correction region

### INPUT FILES

**For 4D data:**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*4rrrr* - processed 4D data

*proc* - F4 processing parameters

*proc2* - F3 processing parameters

*proc3* - F2 processing parameters

*proc4* - F1 processing parameters

For 3D data, the input data file is *3rrr* whereas the *proc4* does not exist. For data of dimension n where n ≥ 5, input data files are named *nr* and *ni*, e.g. *5r*, *5i*, *6r*, *6i* etc.

**OUTPUT FILES**

**For 4D data:**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*4rrrr* - processed 4D data

*procs* - F4 processing status parameters

*proc2s* - F3 processing status parameters

*proc3s* - F2 processing status parameters

*proc4s* - F1 processing status parameters

For 3D data, the output data file is *3rrr* whereas *proc4s* does not exist. For data of dimension n where n ≥ 5, output data files are named *nr* and *ni*, e.g. *5r*, *5i*, *6r*, *6i* etc.

**SEE ALSO**

*abs1, abst1, absd1, absot1, bas [▷ 97]*, *abs2, abst2, absd2, absot2 [▷ 95]*, *tabs3, tabs2, tabs1 [▷ 172]*

## 6.2 ftnd

**NAME**

ftnd - nD processing including Fourier transform (≥ 3D)

**DESCRIPTION**

The command **ftnd** processes nD data performing fid baseline correction, linear prediction, window multiplication, Fourier transform and phase correction. The command automatically recognizes the data dimensionality and handles data of dimension ≥3D. In TopSpin, **ftnd** has been tested by Bruker on 3D, 4D, 5D and 6Ddata. Note that 3D data can also be processed with the conventional commands **tf3**, **tf2**, **tf1** and **ft3d**.

As an example, **ftnd** is described here for a 4D dataset. It takes the following three arguments:

**<direction>**

the direction(s) to be processed. Allowed values are:

0 : all directions, in the order defined by AQSEQ

4321, 4312, 4231, 4213, 4132, 4123 : all directions in specified order

4, 3, 2, or 1 : F4, F3, F2 or F1, respectively.

**<procno>**

Output procno of the processed data. Optional argument, normally unused. In special cases, however, the data cannot be processed in-place, and must be stored in a different procno. **ftnd** will then prompt you for an output procno.

**dlp**

Delayed linear prediction. Optional argument, only applicable when all directions are processed. This argument ensures that when linear prediction is performed in a certain direction, all other directions are already Fourier transformed (see below).

If the arguments are not specified on the command line, **ftnd** will normally only prompt you for the direction. The output procno is only prompted for if inplace operation is not possible.

Here are some example of specifying arguments on the command line:

**ftnd 0**

Process the data in all directions in the order defined by the acquisition status parameter AQSEQ

**ftnd 4**

Process data in direction F4

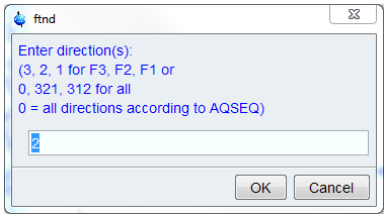
**ftnd 4312 999**

Process the data in all directions, in the order F4-F3-F1-F2 and store the result in procno 999

**ftnd 0 dlp**

Process the data in all directions, in the order defined by AQSEQ, performing delayed linear prediction according to ME_MOD and NCOEF.

Missing arguments are prompted for, except for the **dlp** argument. Note that for the first argument, the direction, only the allowed directions are displayed and the next logical direction is suggested. The figure below shows the dialog opened by **ftnd** on a 4D dataset that has already been processed in F4 and F3.




**Extract 1D, 2D or 3D data from 4D, 5D,... processed data.**

To view the result of 4D processing, open the dataset (*procno*) where the processed data are stored and read a 3D-cube, 2D-plane or 1D trace. This can be done with the commands **rcb**, **rpl** and **rtr**, respectively. These commands automatically switch to the destination dataset showing the 3D, 2D or 1D dataset, respectively (see the description of these commands for more information). Furthermore, you can extract positive, negative or sum cube projections with the commands **projcbp**, **projcbn** and **sumcb**, respectively. Similarly, you can extract plane projections with the commands **projpIp**, **projpIn** and **sumpl**, respectively.

Instead of processing the entire 4D dataset and reading a certain plane or trace, you can also process single 2D-planes or 1D fids of the 4D raw data. To process a plane, just enter **xfb**, **xf2** or **xtrf** and specify the requested plane axis orientation, plane number and output *procno*. To process a trace, just enter a 1D processing command like **ft** or **trf** and specify the requested fid number and output *procno*. Obviously, 1D/2D processing commands can also be used to further process or reprocess traces/planes or processed 4D data. For example:

1. Open a 4D dataset
2. **ftnd 4** - Perform 4D processing in the F4 direction
3. **rpl 34 1 999** - Read F3-F4 plane 1 and store it in *procno* 999. Note that the plane is stored as a F2-processed 2D dataset.
4. **xf1** - Perform 2D processing in the F1-direction.

**Processing the four directions in separate steps**

Normally, **ftnd** with the argument **0** or one of the arguments 4321, 4312, .. etc. to process all directions. In some cases, you may want to process the different directions in individual steps and perform the sequence **ftnd 4**, **ftnd 3**, .. etc. The first direction to be processed must be

F4, the other three directions can be processed in any order. Note that every order in which the data are processed in F3, F2 an and F1 gives the same result, unless linear prediction is done (ME_mod and NCOEF ≠ 0)

### Delayed linear prediction

Linear prediction is a valuable method for improving the resolution of nD data with small TD values and often truncated FIDs. The effect of linear prediction in one direction can, however, be distorted by modulations introduced by other, untransformed, directions. The **dlp** argument allows to perform linear prediction in a certain direction while all other directions have already been Fourier transformed. Let's take an example to see how this works. Suppose you have a 4D dataset with acquisition order 4321 (parameter AQSEQ), which you want to processed in all 4 directions including Window Multiplication (WM) and Fourier transform (FT). Furthermore, you want to increase the resolution with linear prediction (LP) in the third (F2) and fourth (F1) direction. As such, you have set the parameters WDW, and, in F2 and F1, ME_mod and NCOEF, to appropriate values. If you use the command **ftnd 0** the following happens:

- Processing in F4 (WM - FT)
- Processing in F3 (WM - FT
- Processing in F2 (LP - WM - FT
- Processing in F1 (LP - WM - FT

So when linear prediction is done in F2, data have not been Fourier transformed yet in F1, which can cause distortions.

If, however, you use the command **ftnd 0 dlp** for delayed linear prediction, the following happens:

- Processing in F4 (WM - FT)
- Processing in F3 (WM - FT)
- Processing in F2 (FT)
- Processing in F1 (LP - WM - FT)
- Processing in F2 (IFT - inverse Fourier transform, including Hilbert Transform to create temporary imaginary data)
- Processing in F2 (LP - WM - FT)

Now when linear prediction is done in F2, the data are Fourier transformed in F1 (and all other directions). For the F1 direction, linear prediction does not have to be delayed because F1 is the last direction being processed. Note that **ftnd** also performs fid baseline correction and spectrum phase correction if the parameters BC_mod and PH_mod, respectively, are set.

Delayed linear prediction can also be performed in two steps. The command:

**ftnd 0 dlp** (with F2-ME_mod ≠ 0 and NCOEF ≠0)

is equivalent with the command sequence:

- **ftnd 0** (with F2-ME_mod = 0) and WDW = 0
- **lpnd 2** (with F2-ME_mod ≠ 0, NCOEF ≠ 0 and WDW ≠ 0)

### In-place operation

Normally, **ftnd** can perform an in-place operation, which means the processed data are stored in the current *procno*. In special cases, however, in-place operation is not possible and the processed data must be stored in a different *procno*. **ftnd** will prompt the user for the output *procno*. When processing is finished, the display will automatically change to the destination PROCNO.

Whether or not in-place operation is possible depends on the direction being processed and the zero-filling conditions. In-place operation is done:

- In the first direction: always
- In the second direction: always as long as all directions are processed with one command, e.g. with **ftnd 0**.
- In the third, fourth etc. directions: if at least single zero filling (SI ≥ TD and (STSI = 0 or STSI ≥ TD)).

Note that if a *procno* is specified on the command line, it is used, i.e. the processed data of the last two directions are stored there.

Restrictions nD processing

The command **ftnd** has the following restrictions:

- Raw and processed data have the same dimensionality, i.e. the values of the status parameters PARMODE and PPARMOD must be the same. Note that 2D processing commands like **xfb** also work on datasets with different raw and processed data dimensionality, e.g. 3D raw and 2D processed data.
- If dimension > 3 and the acquisition mode (acquisition status parameter FnMODE) is QF in one direction, it must be QF in all directions. In other words, you cannot process mixed single detection/hypercomplex data for dimension > 3.
- For data of dimension ≥ 5D, only the natural acquisition order (AQSEQ = 0) is supported.
- Simultaneous echo-antiecho not supported; the acquisition status parameter FnMODE must not be echo-antiecho in more than 1 direction.

Note that the values of parameters which use a predefined list are stored as integers. The first value of the list is always stored as 0, the second value as 1 etc. The table below shows the values of the parameter PH_mod as an example:

| Parameter value | Integer stored in the proc(s) file |
|---|---|
| no | 0 |
| pk | 1 |
| mc | 2 |
| ps | 3 |

## INPUT PARAMETERS

### F4, F3, F2 and F1 parameters

Set by the user with **edp** or by typing **si**, **stsr** etc.:

SI - size of the processed data

STSR - strip start: first output point of strip transform

STSI - number of output points of strip transform

TDeff - number of raw data points to be used for processing

TDoff - first point of the FID used for processing (default 0)

FCOR - first (FID) data point multiplication factor (0.0-2.0, default 0.5)

REVERSE - flag indicating to reverse the spectrum

BC_mod - FID baseline correction mode

BCFW - filter width for BC_mod = sfil or qfil

COROFFS - correction offset for BC_mod = spol/qpol or sfil/qfil

ME_mod - FID linear prediction mode

NCOEF - number of linear prediction coefficients

LPBIN - number of points for linear prediction

TDoff - number of raw data points predicted for ME_mod = LPb*
WDW - FID window multiplication mode
LB - Lorentzian broadening factor for WDW = em or gm
GB - Gaussian broadening factor for WDW = gm, sinc or qsinc
SSB - Sine bell shift for WDW = sine, qsine, sinc or qsinc
TM1, TM2 - limits of the trapezoidal window for WDW = trap
PH_mod - phase correction mode
PHC0 - zero order phase correction value for PH_mod = pk
PHC1 - first order phase correction value for PH_mod = pk
Set by the acquisition, can be viewed with **dpa** or **s aq_mod** etc.:
TD - time domain; number of raw data points

**F4 parameters**

Set by the user with **edp** or by typing **aqorder**, **pknl** etc.:
AQORDER - Acquisition order
PKNL - group delay compensation (Avance) or filter correction (A*X)
Set by the acquisition, can be viewed with **dpa** or **s aq_mod** etc.:
AQ_mod - acquisition mode (determines the status FT_mod)
AQSEQ - acquisition sequence (3-2-1 or 3-1-2)
BYTORDA - byteorder or the raw data
NC - normalization constant

**F3, F2 and F1 parameters**

Set by the acquisition, can be viewed with **dpa** or by typing **s fnmode** etc.:
FnMODE - Fourier transform mode

**OUTPUT PARAMETERS**

**F4, F3, F2 and F1**

Can be viewed with **dpp** or by typing **s si**, **s stsi** etc.:
SI - size of the processed data
STSR - strip start: first output point of strip transform
STSI - strip size: number of output points of strip transform
TDeff - number of raw data points that were used for processing
TDoff - first point of the FID used for processing (default 0)
XDIM - subcube size
FT_mod - Fourier transform mode
FTSIZE - Fourier transform size

**F4 parameters**

Can be viewed with **dpp** or by typing **s si**, **s tdeff** etc.:
AQORDER - Acquisition order
YMAX_p - maximum intensity of the processed data
YMIN_p - minimum intensity of the processed data
S_DEV - standard deviation of the processed data

NC_proc - intensity scaling factor

BYTORDP - byte order of the processed data

**F3, F2 and F1 parameters**

Can be viewed with **dpp** or by typing **s mc2** etc.:

MC2 - Fourier transform mode

**INPUT FILES**

**For 4D data:**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*ser* - raw data

*acqus* - F4 acquisition status parameters

*acqu2s* - F3 acquisition status parameters

*acqu3s* - F2 acquisition status parameters

*acqu4s* - F1 acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*proc* - F4 processing parameters

*proc2* - F3 processing parameters

*proc3* - F2 processing parameters

*proc4* - F1 processing parameters

For 3D data *proc4s* does not exist. For data of dimension n where n ≥ 5 the additional files proc5,...,etc. exist.

**OUTPUT FILES**

**For 4D data:**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*4rrrr* - processed 4D data

*procs* - F4 processing status parameters

*proc2s* - F3 processing status parameters

*proc3s* - F2 processing status parameters

*proc4s* - F1 processing status parameters

For 3D data, the output data file is *3rrr* whereas *proc4s* does not exist. For data of dimension n where n ≥ 5, processed data files are named *nr* and *ni*, e.g. *5r*, *5i*, *6r*, *6i* etc. and the additional files proc5s,..., etc. exist.

**SEE ALSO**

*absnd* [▶ 186], *lpnd* [▶ 192], *pknd* [▶ 195], *projcbp, projcbn, sumcb* [▶ 196], *projplp, projpln, sumpl* [▶ 166]

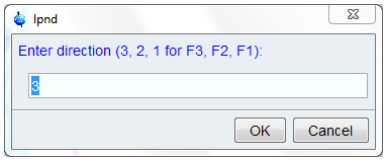
## 6.3 lpnd

**NAME**

lpnd - nD linear prediction

## DESCRIPTION

The command **lpnd** performs a linear prediction of data with dimension ≥3D. It takes one argument, the direction to be processed. If no argument is specified on the command line, it is requested:




**lpnd** works on data that have already been Fourier transformed in the specified direction, e.g. with **ftnd**. Since linear prediction is normally performed on a unfiltered FID, the data should first be processed with **ftnd** with WDW = no, and then with **lpnd** while WDW is set to the desired window function.

**lpnd** performs the following steps in the specified direction:

1. Inverse Fourier transform (if imaginary data do not exist, they are automatically created with Hilbert transform).
2. Regular processing including:
   - Linear prediction according to ME_mod, NCOEF
   - Window multiplication according to WDW
   - Fourier transform

Linear prediction is a valuable method for improving the resolution of nD data with small TD values and often truncated FIDs. The effect of linear prediction in one direction can, however, be distorted by modulations introduced by other, untransformed, directions. Therefore, it is a good idea to first process the data in all directions and then perform **lpnd**. This entire procedure, including the correct window handling, is automatically performed by the command **ftnd dlp** (delayed linear prediction). However, if you want both backward and forward prediction, the latter must be done with **lpnd**. In this case, you have to perform the following steps:

1. Backward prediction with **ftnd** while ME_mod=LPbr or LPbc and WDW=no.
2. Forward prediction with **lpnd** while ME_mod=LPfr or LPfc and WDW set to the desired window function.

For more information, see the description of **ftnd**.

## INPUT AND OUTPUT PARAMETERS

See **ftnd**

## INPUT FILES

**For 4D data:**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*4rrrr* - processed 4D data

*proc* - F4 processing parameters

*proc2* - F3 processing parameters

*proc3* - F2 processing parameters

*proc4* - F1 processing parameters

For 3D data, the input data file is *3rrr* whereas the *proc4* does not exist. For data of dimension n where n ≥ 5, input data files are named *nr* and *ni*, e.g. *5r*, *5i*, *6r*, *6i* etc.

### OUTPUT FILES

**For 4D data:**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*4rrrr* - processed 4D data

*procs* - F4 processing status parameters

*proc2s* - F3 processing status parameters

*proc3s* - F2 processing status parameters

*proc4s* - F1 processing status parameters

For 3D data, the output data file is *3rrr* whereas *proc4s* does not exist. For data of dimension n where n ≥ 5, output data files are named *nr* and *ni*, e.g. *5r*, *5i*, *6r*, *6i* etc.

### SEE ALSO

*ftnd* [▶ 187]

## 6.4 mcnd

### NAME

mcnd - magnitude calculation on nD data

### DESCRIPTION

The command **mcnd** calculates the magnitude spectrum of a nD dataset. The intensity i is replaced by its absolute value according to the formula:

$$ABS(i) = \sqrt{(R(i)^2 + I(i)^2)}$$

where R and I are the real and imaginary part of the spectrum respectively. The imaginary part of nD QF datasets is kept in a separate file

in 3iii for 3D data

in 4iii for 4D data

in 5i for 5D data

in 6i for 6D data

when processing the last direction of a nD QF dataset. PH_mod in this direction is usually set to mc. This leads to a magnitude calculation after Fourier transform and the file holding imaginary data is removed.

With the **mcnd** command the magnitude calculation can be done in a separate processing step, especially if PH_mod in the last processing direction was not set to mc or ps.

If the command **mcnd** is applied to hypercomplex nD datasets imaginary data are calculated internally by a Hilbert transform.

### INPUT FILES

*3rrrr*, *3iii* - for 3D data

*4rrrr*, *4iiii* - for 4D data

*5r*, *5i* - for 5D data

*6r*, *6i* - for 6D data

### OUTPUT FILES

*auditp.txt*

*3rrr* - for 3D data

*4rrrr* - for 4D data

*5r* - for 5r data

*6r* - for 6D data

**SEE ALSO**

*mc* [▸ 68], *ps* [▸ 74], *apk, apks, apkm, apkf, ph* [▸ 51], *trf, trfp* [▸ 89]

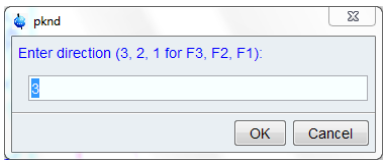
## 6.5 pknd

**NAME**

pknd - nD phase correction

**DESCRIPTION**

The command **pknd** performs a phase correction of data of dimension ≥3D, applying the values of PHC0 and PHC1. It takes one argument, the direction to be corrected. If no argument is specified on the command line, it is requested:




Before you execute **pknd**, the phase values must first be determined, for example on a 2D plane. You can do that by typing **xfb** on the nD data to process a plane, do a phase correction on the resulting the 2D dataset and store the phase values in the nD dataset.

Note that phase correction normally requires the existence of imaginary data. Usually, however these do not exist for data of dimension ≥ 4. Therefore, **pknd** automatically creates temporary imaginary data using Hilbert transform. Actually the command processes 2D planes of an nD dataset, performing a series of **xht2** - **xf2p** or **xht1** - **xf1** commands.

On 3D data, the commands **pknd 3**, **pknd 2** and **pknd 1** are equivalent to **tf3p**, **tf2p** and **tf1p**, respectively.

**INPUT PARAMETERS**

Set by the user with **edp** or by typing **phc0**, **phc1** etc.

PHC0 - zero order phase correction value (frequency independent)

PHC1 - first order phase correction value (frequency dependent)

**OUTPUT PARAMETERS**

Can be viewed with **dpp** or by typing **s phc0**, **s phc1** etc.:

PHC0 - zero order phase correction value (frequency independent)

PHC1 - first order phase correction value (frequency dependent)

**INPUT FILES**

**For 4D data:**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*4rrrr* - processed 4D data

*proc* - F4 processing parameters

*proc2* - F3 processing parameters

*proc3* - F2 processing parameters

*proc4* - F1 processing parameters

For 3D data, the input data file is *3rrr* whereas the *proc4* does not exist. For data of dimension n where n ≥ 5, input data files are named *nr* and *ni*, e.g. *5r*, *5i*, *6r*, *6i* etc.

### OUTPUT FILES

**For 4D data:**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*4rrrr* - processed 4D data

*procs* - F4 processing status parameters

*proc2s* - F3 processing status parameters

*proc3s* - F2 processing status parameters

*proc4s* - F1 processing status parameters

For 3D data, the output data file is *3rrr* whereas *proc4s* does not exist. For data of dimension n where n ≥ 5, output data files are named *nr* and *ni*, e.g. *5r*, *5i*, *6r*, *6i* etc.

### SEE ALSO

*ftnd* [▶ 187], *tf3p, tf2p, tf1p* [▶ 184], *xfbp, xf2p, xf1p* [▶ 150], *xht2, xht1* [▶ 152]

## 6.6 projcbp, projcbn, sumcb

### NAME

projcbp - Calculate positive 3D projection

projcbn - Calculate negative 3D projection

sumcb - Calculate sum 3D projection

### DESCRIPTION

The commands **projcbp**, **projcbn** and **sumcb** calculate the positive, negative and sum 3D projection, respectively, from a dataset of dimension ≥ 4.

They require take up to 5 arguments:

- **<cube orientation>** *: 234, 134, 124, ..., 432, 321 etc.*
- **<first cube>** : the first cube included in the calculation
- **<last cube>** : the last cube included in the calculation
- **<dest. procno>** : the *procno* where the 3D output data are stored
- **xdim :** sets the subcube sizes according to XDIM (optional)
- **n :** prevents the destination dataset from being displayed/activated (optional)

Here is an example of the usage of a 3D projection command:

**projcbp 234 1 32 999 n**

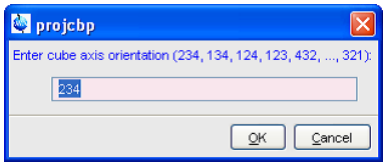
Calculates the positive F2-F3-F4 3D projection of cube 1 to 32 along the F1 direction, stores it under PROCNO 999 but does not change the display to the output data.

Instead of specifying the first and last cube, you can also use the argument **all** for all cubes. For example:

**projcbp 234 all 10**

Calculates the positive F2-F3-F4 3D projection of all cubes along F1 and stores it under PROCNO 10.

Missing arguments (except for the optional ones) will be prompted for. For example, entering **projcbp** without any arguments will display the dialog:




Note the following aspects:

- The maximum first and last cube is determined by the size of the data in the direction not included cube orientation; i.e. the direction along which the projection is calculated.
- XDIM is a processing parameter which must be set in each direction included cube orientation when with the **xdim** argument is used.
- The numerical arguments must be specified in the above order, whereas the arguments **all**, **xdim** and **n** can be specified at any position.

### INPUT FILES

For a 4D dataset:

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*4rrrr* - real processed 4D data

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3rrr* - real processed 3D data

*procs* - F3 processing status parameters

*proc2s* - F2 processing status parameters

*proc3s* - F1 processing status parameters

*auditp.txt* - processing audit trail

### SEE ALSO

*projplp, projpln, sumpl* [▶ 166]

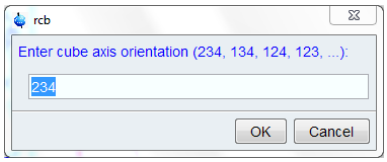
## 6.7 rcb

### NAME

rcb - Read cube from data ≥ 4D and store as 3D data

### DESCRIPTION

The command **rcb** reads a cube from processed data of dimension ≥ 4. It stores the extracted cube in a different *procno* as a 3D dataset.




**rcb** takes up to five arguments:

**<cube axis orientation>** *: 234, 134, 124, ..., 432, 321 etc.*

The digits refer to the F4, F3, F2 and F1 axes of the 4D data. Note that the order of the three digits is relevant:

- The first digit is the 4D axis that corresponds to the 3D-F1 axis.
- The second digit is the 4D axis that corresponds to the 3D-F2 axis.
- The last digit is the 4D axis that corresponds to the 3D-F3-axis.

This means that for values like 234, 134, 124 etc. the axis order or the 3D cube and the 4D dataset are the same. For values like 432, 423, 143 etc., they are different.

**<cube number>** : 1 - SI

SI is the 4D size in the direction orthogonal to the cube orientation

**<procno>** :

Destination 3D *procno* (source 4D *procno* if **rcb** is entered on the destination 3D dataset)

**xdim :** optional argument

Sets the subcube sizes according to the processing parameter XDIM in the respective directions. This parameter must be set in the source 4D dataset before **rcb** is executed.

**n :** optional argument

Prevents the destination dataset from being displayed/activated

Arguments which are not specified on the command line will be prompted for, except for **xdim** and **n** argument.

**rcb** can be entered on the source 4D dataset or, if this already exists, on the destination 3D dataset. The number of required arguments is different (see below).

### rcb entered on a source 4D dataset

In this case, **rcb** prompts the user for three arguments. Alternatively, these can be entered on the command line.

Here are some examples:

**rcb**

Prompt the user for the *cube axis orientation*, the *cube number* and *destination 3D procno* and read the cube accordingly.

**rcb 234 10 999**

Read F2-F3-F4 cube 10 and store it in *procno* 999.

**rcb 324 10 999**

Read F2-F3-F4 plane 10 and store it in *procno* 999, exchanging the F2 and F3 axes

**rcb 124 64 101 xdim**

Read F1-F2-F4 plane 64 with subcube sizes according to the respective XDIM values and store it in *procno* 101.

**rcb 124 64**

Read F1-F2-F4 plane 64, prompt the user for the destination *procno*

**rcb 214 1 10 n**

Read an F1-F2-F4 plane number 1 and store it in *procno* 10, exchanging the F2 and F1 axes. Do not display/activate the destination dataset.

**rcb entered on a destination 3D dataset**

This is typically done on a 3D dataset which is a cube extracted by a previous **rcb** command, which was entered on the source 4D dataset. In that case, **rcb** requires only one argument; the *cube number.* By default, the same cube *axis orientation* and *source 4D dataset (procno)* are used as with the previous **rcb** command (as defined in the *used_from* file of the 3D dataset). You can, however, use two or three arguments to specify a different cube *axis orientation* and/or *4D source procno*. On a regular 3D dataset (not a plane from a 3D), **rcb** requires three arguments.

Here are some examples of **rcb** executed on a 3D dataset, where the 3D dataset is a cube from a 4D dataset:

**rcb**

Prompt the user for the *cube number*. Use the *cube axis orientation* and *source 4D procno* as defined in the current 3D dataset.

**rcb 11**

Read cube 11. Use the *cube axis orientation* and *Source 4D procno* as defined in current 3D dataset.

**rcb 123 11**

Read F1-F2-F3 plane 11. Use the *source 4D procno* as defined in current 3D dataset.

**rcb 123 11 2**

Read F1-F2-F3 plane 11 from the 4D dataset under *procno* 2

As described above, the **rcb** argument *cube axis orientation* determines whether the axes are exchanged. Axes exchange is sometimes required to match nuclei when you compare a 4D cube with a 3D dataset.

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*4rrrr*, *4iiii* - processed 4D data

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3rrr*, *3iii* - processed 3D data

*auditp.txt* - processing audit trail

*used_from* - data path of the source 4D data and the cube axis orientation

**SEE ALSO**

*rpl* [▶ 199], *wpl* [▶ 205], *rtr* [▶ 202], *wtr* [▶ 207]

## 6.8 rpl

**NAME**

rpl - Read plane from data ≥ 3D and store as 2D data

**DESCRIPTION**

The command **rpl** reads a plane from processed data with dimension ≥ 3D and stores it as a 2D dataset in a different *procno*.

**rpl** takes up to five arguments. As an example we take a plane read from a 3D dataset:

**<plane axis orientation>** *: 23, 13, 12, 32, 31 or 21*

The digits refer to the F3, F2 and F1 axes of the 3D data. Note that the order of the two digits is relevant:

- the first digit is the 3D axis that corresponds to the 2D-F1 axis
- the last digit is the 3D axis that corresponds to the 2D-F2-axis

This means that for the values 21, 31 and 32, the axes are exchanged, storing rows as columns and vice versa (see below).

**<plane number>** : 1 - SI

SI is the 3D size in the direction orthogonal to the plane orientation

**<procno>** :

Destination 2D *procno* (source 3D *procno* if **rpl** is entered on the destination 2D dataset)

**<inmem>** : optional argument for usage in AU programs only

Improves performance by data caching. Caution: nD data must not be modified by any command other than **wpl** between two consecutive **rpl inmem** or **wpl inmem** commands.

**n :** optional argument

Prevents the destination dataset from being displayed/activated

Obligatory arguments which are not specified on the command line will be prompted for.

**rpl** can be entered on the source 3D dataset or, if it already exists, on the destination 2D dataset. The number of required arguments is different (see below).

**rpl entered on a source 3D dataset**

In this case, **rpl** prompts the user for three arguments. Alternatively, these can be entered on the command line.

Here are some examples:

**rpl**

Prompt the user for the *plane axis orientation*, the *plane number* and *source 3D procno* and read the plane accordingly.

**rpl 23 10 999**

Read F2-F3 plane 10 and store it in *procno* 999.

**rpl 32 10 999**

Read F2-F3 plane 10 and store it in *procno* 999, exchanging the F2 and F3 axes.

**rpl 12 64 101**

Read F1-F2 plane 64 and store it in *procno* 101.

**rpl 12 64**

Read F1-F2 plane 64, prompt the user for the *destination procno*

**rpl 31 1 10 n**

Read an F1-F3 plane number 1 and store it in *procno* 10, exchanging the F1 and F3 axes. Do not display/activate the destination dataset.

**rpl entered on a destination 2D dataset**

This is typically done on a 2D dataset which is a plane extracted by a previous **rpl** command, which was entered on the source 3D dataset. In that case, **rpl** requires only one argument; the *plane number*. By default, the same p*lane axis orientation* and *source 3D dataset (procno)* are used as with the previous **rpl** command (as defined in the *used_from* file of the

2D dataset). You can, however, use two or three arguments to specify a different p*lane axis orientation* and/or *3D source procno*. On a regular 2D dataset (not a plane from a 3D), **rpl** requires three arguments.

Here are some examples of **rpl** executed on a 2D dataset, where the 2D dataset is a plane from a 3D dataset:

**rpl**

Prompt the user for the plane number, use the *plane axis orientation* and *source 3D procno* as defined in the current 2D dataset and read the plane accordingly.

**rpl 11**

Read plane 11. Use the *plane axis orientation* and *source 3D procno* as defined in current 2D dataset.

**rpl 31 11**

Read F1-F3 plane 11, exchanging the F1 and F3 axes. Use the *source 3D procno* as defined in current 2D dataset.

**rpl 13 11 2**

Read F1-F3 plane 11 from the 3D dataset under *procno* 2

As described above, the **rpl** argument *plane axis orientation* determines whether the axes are exchanged. This is sometimes required to match nuclei when you compare a 3D plane with a 2D dataset. Example: you have a 3D NOESYHSQC (F3-1H, F2-13C, F1-1H) and want to compare an F2-F1 plane with a 2D HSQC (F2-1H, F1-13C). Now compare the following actions:

**rpl 12**: The plane is stored as a 2D dataset with F2-13C, F1-1H which cannot be directly compared with the a HSQC.

**rpl 21**: The plane is stored as a 2D dataset with F2-1H, F1-13C which can be directly compared with the a HSQC.

In special cases, **rpl** results in a 2D dataset which is not Fourier transformed in F2. This occurs, for example, if you run **rpl 12** on a 3D dataset which has only been transformed in F3. **rpl** unshuffles the output data, storing the odd and even points in separate data files (*2rr* and *2ir*). As a result the size in F2 (parameter SI) is only half the size of the corresponding direction in the 3D dataset. If, for some reason, you want keep the same size, you can use **rpl** with the option **keepsize**. The output data are then zero filled once in F2. Here is an example:

**rpl 12 1 10 keepsize**

Note that a plane read with **keepsize** cannot be written back to the source dataset because the sizes do not match.

The behaviour of the command **rpl** is similar to the commands **rsr** and **rsc**, in the sense that it can be entered from the source and destination dataset.

On a data with dimension > 3, **rpl** works the same as on a 3D dataset, except that there are more plane axis orientations. For example on 4D dataset, possible orientations are *34*, *24*, *14*, *23*, *13*, *12*, *43*, *42*, *41*, *32*, *31* and *21*.

For an example if the **inmem** option, see the AU program **ift3d**.

## INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3rrr*, *3irr*, *3rir*, *3rri*, *3iii* - processed data (**rpl** on 3D data)

*4rrrr*, *4iiii* - processed data (**rpl** on 4D data)

## OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *2ir*, *2ri*, *2ii* - processed 2D data

*auditp.txt* - processing audit trail

*used_from* - data path of the source 3D data and the plane number

**SEE ALSO**

*wpl* [ 205], *rtr* [ 202], *wtr* [ 207], *rcb* [ 197], *wser* [ 127], *wserp* [ 128], *rser2d* [ 171]

## 6.9 rtr

**NAME**

rtr - Read trace from data ≥ 2D and store as 1D data

**DESCRIPTION**

The command **rtr** reads a trace from processed data with dimension ≥ 2D and stores it as a 1D dataset.

**rtr** takes up to four arguments. As an example we take a trace read from a 3D dataset:

**<axis orientation>** : 1, 2 or 3

The digit refers to the F3, F2 and F1 axis of the 3D data.

**<trace number>** : 1 - MAX

Where MAX is the product of the SI value in the directions orthogonal to the trace orientation.

<*procno*> :

Destination 1D *procno* (source 3D *procno* if **rtr** is entered on the destination 1D dataset)

**n :** optional argument.

Prevents the destination dataset from being displayed/activated

Obligatory arguments that are not specified on the command line will be prompted for.

**rtr** can be entered on the source 3D dataset or, if this already exists, on the destination 1D dataset. The number of required arguments is different (see below).

**rtr entered on a source 3D dataset**

In this case, **rtr** prompts the user for three arguments. Alternatively, these can be entered on the command line.

**rtr**

Prompt the user for the *axis orientation*, *trace number* and *destination procno* and read the trace accordingly.

**rtr 3 10 999**

Read F3 trace 10 and store it in *procno* 999.

**rtr 1 1 10 n**

Read F1 trace 1 and store it in *procno* 10. Do not display/activate the destination dataset.

**rtr entered on a destination 1D dataset**

This is typically done on a 1D dataset which is a trace extracted by a previous **rtr** command, which was entered on the source 3D dataset. In that case, **rtr** requires only one argument; the *trace number.* By default, the same *axis orientation* and *source 3D dataset (procno)* are used as with the previous **rtr** command (as defined in the *used_from* file of the 1D dataset). You can, however, use two or three arguments to specify a different *axis orientation* and/or *3D source procno*. On a regular 1D dataset (not a trace from a 3D), **rtr** requires three arguments.

Here are some examples of **rtr** executed on a 1D dataset which is a trace from a 3D dataset:

**rtr**

Prompt the user for the *trace number*, use the *axis orientation* and *source 3D procno* as defined in the current 1D dataset and read the trace accordingly.

**rtr 11**

Read trace 11. Use the *axis orientation* and *source 3D procno* as defined in current 1D dataset.

**rtr 3 11 2**

Read F3 trace 11 from the 3D dataset under *procno* 2

Note that on 2D data the command **rtr** works like **rsr** and **rsc**, except that the trace direction can be freely chosen. Furthermore, **rtr** always stores the 1D output data in a different *procno* of the same dataset whereas **rsr** and **rsc** can store data in the dataset ~TEMP.

On 4D or higher dimensional datasets, **rtr** works the same as on a 3D dataset, except that there are more axis orientations.

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *2ir*, *2ri*, *2ii* - processed data (**rtr** on 2D data)

*3rrr*, *3irr*, *3rir*, *3rri*, *3iii* - processed data (**rtr** on 3D data)

*4rrrr*, *4iiii* - processed data (**rtr** on 4D data)

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data

*auditp.txt* - processing audit trail

*used_from* - data path of the source data and the trace number

### SEE ALSO

*wtr* [▷ 207], *rpl* [▷ 199], *wpl* [▷ 205], *rcb* [▷ 197], *wser* [▷ 127], *wserp* [▷ 128]

## 6.10 wcb

### NAME

wcb - Write 3D data to a cube of data ≥ 4D

### DESCRIPTION

The command **wcb** replaces a cube of processed data with dimension ≥ 4D with a 3D processed dataset. It is usually, but not necessarily, used to write back a cube that was extracted with **rcb**. This cube can be modified and/or written back to a different cube number.

**wcb** takes up to three arguments. As an example, we take a cube written to a 4D dataset:

<cube axis orientation> : 234, 134, 124, 123, 432, ..., 321 etc.

The digits refer to the F4, F3, F2 and F1 axes of the 4D data. Note that the order of the three digits is relevant:

the first digit is the 4D axis that corresponds to the 3D-F1 axis

the last digit is the 4D axis that corresponds to the 3D-F3-axis

<cube number> : 1 - SI

SI is the 4D size in the direction orthogonal to the cube axis orientation

<procno>

destination 4D procno (source 4D procno if **wcb** is entered on the destination 3D dataset).

**wcb** can be entered on the 3D source dataset or on the destination 4D dataset. The number of required arguments is different (see below).

### wcb entered on the source 3D dataset

In this case, **wcb** prompts the user for two arguments only, the cube number and the 4D destination procno. The cube axis orientation is taken from the 3D dataset (used_from file). The two arguments can also be specified on the command line. If, however, you specify three arguments, the plane axis orientation is taken from the first argument rather than from the 3D dataset.

Examples:

**wcb**

prompt the user for the cube number and destination 4D procno, take the cube axis orientation from the current 3D dataset and write the cube accordingly.

**wcb 11 1**

write the current 3D data to cube 11 of the 4D dataset in procno 1. Take the cube axis orientation from the current 2D dataset.

**wcb 432 11 2**

write the current 4D data to F2-F3-F4 cube number 11 of the 4D data in procno 2, exchanging the F2 and F4 axes.

Note that if the source 3D dataset does not contain a *used_from* file, for example because it is not an extracted plane, **wcb** will prompt the user for the cube axis orientation.

### Entering wcb on the destination 4D dataset

In this case, **wcb** prompts the user for three arguments. Alternatively, these can be entered on the command line.

Examples:

**wcb 234 10 999**

Write the 3D data in procno 999 to F2-F3-F4 cube 10 of the current 4D data.

**wcb 234 32 101**

Write the 3D data in procno 101, to the F2-F3-F4 cube 32 of the current 4D data

**wcb 234**

Prompt the user for the procno of the source 3D dataset and the cube number. Write the 3D dataset to the specified F2-F3-F4 cube accordingly.

### Entering wcb on a 5D dataset

On a data with dimension > 4, **wcb** works the same as on a 4D dataset, except that there are more cube axis orientations.

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3rrr, 3iii* - processed 3D data

*used_from* - data path of the source 4D data and the cube number

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*4rrrr*, - processed data (**wcb** on 4D data)

*5r*, *5i* - processed data (**wcb** on 5D data)

*auditp.txt* - processing audit trail

**SEE ALSO**

*rcb* [▷ 197], *rpl* [▷ 199], *rtr* [▷ 202], *wtr* [▷ 207], *rser* [▷ 118], *wser* [▷ 127], *wserp* [▷ 128]

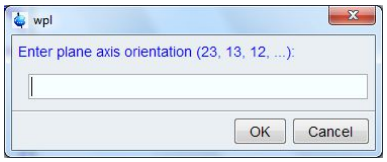
## 6.11 wpl

**NAME**

wpl - Write 2D data to a plane of data ≥ 3D

**DESCRIPTION**

The command **wpl** replaces a plane of processed data with dimension ≥ 3D with a 2D processed data set. It is usually, but not necessarily, used to write back a plane that was extracted with **rpl**. This plane can be modified and/or written back to a different plane number.




**wpl** takes up to four arguments. As an example we take a plane written to a 3D data set:

**<plane axis orientation>** : 12, 13, 23, 21, 31 or 32

The digits refer to the F3, F2 and F1 axes of the 3D data. Note that the order of the two digits is relevant:

- the first digit is the 3D axis that corresponds to the 2D-F1 axis
- the last digit is the 3D axis that corresponds to the 2D-F2-axis

This means that for the values 21, 31 and 32, the axes are exchanged, i.e. rows are stored as columns and vice versa (see below).

**<plane number>** : 1 - SI

SI is the 3D size in the direction orthogonal to the plane axis orientation

**<procno>**

Destination 3D *procno* (source 3D *procno* if **wpl** is entered on the destination 2D data set)

**<inmem>** : optional argument for usage in AU programs only

Improves performance by data caching. Caution: nD data must not be modified by any command other than **wpl** between two consecutive **rpl inmem** or **wpl inmem** commands.

**n**

Do not write imaginary data. Only the real data plane is written to the real destination data. This option prevents **wpl** to abort when nD destination data exist, but 2D source data do not. Caution: this options makes the nD imaginary data inconsistent.

**wpl** can be entered on the 2D source dataset or on the destination 3D data set. The number of required arguments is different (see below).

**wpl entered on the source 2D data set**

In this case, **wpl** prompts the user for two arguments only, the *plane number* and the *3D destination procno*. The *plane axis orientation* is taken from the 2D data set (*used_from* file). The two arguments can also be specified on the command line. If, however, you specify three arguments, the *plane axis orientation* is taken from the first argument rather than from the 2D data set.

Examples:

**wpl**

Prompt the user for the *plane number* and *destination 3D procno*, take the *plane axis orientation* from the current 2D data set and write the plane accordingly.

**wpl 11 1**

Write the current 2D data to plane 11 of the 3D dataset in *procno* 1. Take the *plane axis orientation* from the current 2D data set.

**wpl 31 11 2**

Write the current 2D data to F1-F3 plane number 11 of the 3D data in *procno* 2, exchanging the F1 and F3 axes.

Note that if the source 2D data set does not contain a *used_from* file, for example because it is not an extracted plane, **wpl** will prompt the user for the *plane axis orientation*.

**Entering wpl on the destination 3D dataset**

In this case, **wpl** prompts the user for three arguments. Alternatively, these can be entered on the command line.

Examples:

**wpl 23 10 999**

Write the 2D data in *procno* 999 to F2-F3 plane 10 of the current 3D data.

**wpl 12 32 101**

Write the 2D data in *procno* 101, to the F1-F2 plane 32 of the current 3D data

**wpl 12**

Prompt the user for the *procno* of the source 2D dataset and the plane number. Write the 2D dataset to the specified F1-F2 plane accordingly.

**Entering wpl on a 4D dataset**

On a data with dimension > 3, **wpl** works the same as on a 3D data set, except that there are more plane axis orientations. For example on 4D data set, possible orientations are *12, 13, 14, 23, 24, 34, 21, 31, 32, 41, 42* and *43.*

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *2ir*, *2ri*, *2ii* - processed 2D data

*used_from* - data path of the source 3D data and the plane number

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3rrr*, *3irr*, *3rir*, *3rri*, *3iii* - processed data (**wpl** on 3D data)

*4rrrr*, *4iiii* - processed data (**wpl** on 4D data)

*auditp.txt* - processing audit trail

### SEE ALSO

*rpl* [▷ 199], *rtr* [▷ 202], *wtr* [▷ 207], *rcb* [▷ 197], *wser* [▷ 127], *wserp* [▷ 128]

## 6.12 wtr

### NAME

wtr - Write 1D data to a trace of data ≥ 2D

### DESCRIPTION

The command **wtr** replaces a trace of processed data with dimension ≥ 2D with a 1D processed dataset. It is usually, but not necessarily, used to write back a trace that was extracted with **rtr**. This trace can be modified and/or written back to a different trace number.

**wtr** takes up to three arguments. As an example we take a trace written to a 3D dataset:

**<axis orientation>** : 1, 2 or 3

The digit refer to the F3, F2 and F1 axes of the 3D data.

**<trace number>** : 1 - MAX

Where MAX is the product of the SI value in the directions orthogonal to the trace orientation

**<procno>**

Destination 3D *procno* (source 1D *procno* if **wtr** is entered on the destination 3D dataset)

**wtr** can be entered on the 1D source dataset or on the destination 3D dataset. The number of required arguments is different (see below).

#### wtr entered on the source 1D dataset

In this case, **wtr** prompts the user for two arguments only, the *trace number* and the *1D destination procno*. The *axis orientation* is taken from the 3D dataset (*used_from* file). The two arguments can also be specified on the command line. If, however, you specify three arguments, the *axis orientation* is taken from the first argument rather than from the 3D dataset.

Examples:

**wtr**

Prompt the user for the *trace number* and *destination 3D procno*, take the *axis orientation* from the current 1D dataset and write the trace accordingly.

**wtr 11 1**

Write the current 1D data to trace 11 of the 3D dataset in *procno* 1. Take the *axis orientation* from the current 1D dataset.

**wtr 3 11 2**

Write the current 1D data to F3 trace number 11 of the 3D data in *procno* 2.

Note that if the source 1D dataset does not contain a *used_from* file, for example because it is not an extracted trace, **wtr** will prompt the user for the *axis orientation*.

#### Entering wtr on the destination 3D dataset

In this case, **wtr** prompts the user for three arguments. Alternatively, these can be entered on the command line.

Examples:

**wtr 2 10 999**

Write the 1D data in *procno* 999 to F2 trace 10 of the current 3D data.

**wtr 1 32 101**

Write the 1D data in *procno* 101, to the F1 trace 32 of the current 3D data.

**wtr 1**

Prompt the user for the trace number and the *procno* of the source 1D dataset. Write the 1D dataset to the specified F1 trace accordingly.

### Entering wtr on a 4D dataset

On a data with dimension > 3, **wtr** works the same as on a 3D dataset, except that there are more axis orientations. For example on 4D dataset, possible orientations are *1, 2, 3* and *4*.

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data

*used_from* - data path of the source nD data and the trace number

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr*, *2ir*, *2ri*, *2ii* - processed data (**wtr** on 2D data)

*3rrr*, *3irr*, *3rir*, *3rri*, *3iii* - processed data (**wtr** on 3D data)

*4rrrr*, *4iiii* - processed data (**wtr** on 4D data)

*auditp.txt* - processing audit trail

### SEE ALSO

*rtr* [▸ 202], *rpl* [▸ 199], *wpl* [▸ 205], *rcb* [▸ 197], *wser* [▸ 127], *wserp* [▸ 128]

# 7 Analysis Commands

This chapter describes TopSpin analysis commands for 1D, 2D and 3D data. Although they do not really process (manipulate) the data, they are part of the processing part of TopSpin. Some of them merely interpret the data and display their output, i.e. they do not change the dataset in any way. Others change parameters (like **sref** and **sino**) or create new files (like **edti** and **pps**). None of them, however, change the processed data.

## 7.1 autocalib

### NAME

autocalib – automatic calibration (2D)

### DESCRIPTION

The command **autocalib** align 2D and 1D datasets relative to a reference (the first dataset given in the call). As a requirement, the reference has to be a 2D dataset.

### OUTPUT PARAMETERS

As a consequence of the shifting in the alignment the following parameter will be adapt (except for the reference):

SR – Spectrum reference frequency

### USAGE

autocalib F1 F2 “<path_reference>” “<path_data1>” “<path_data2>” ….

F1 / F2 – determine the direction for the alignment

<path_reference> - the first given dataset is the reference as a default (has to be 2D)

<path> - all paths have to be given in the following absolute format:

*<path-to-data>*\<*expno*>*pdata*\<*procno*>

### SEE ALSO

The interactive usage in the TopSpin User Manual – 2D Calibration in Multiple Display.

## 7.2 daisy

### NAME

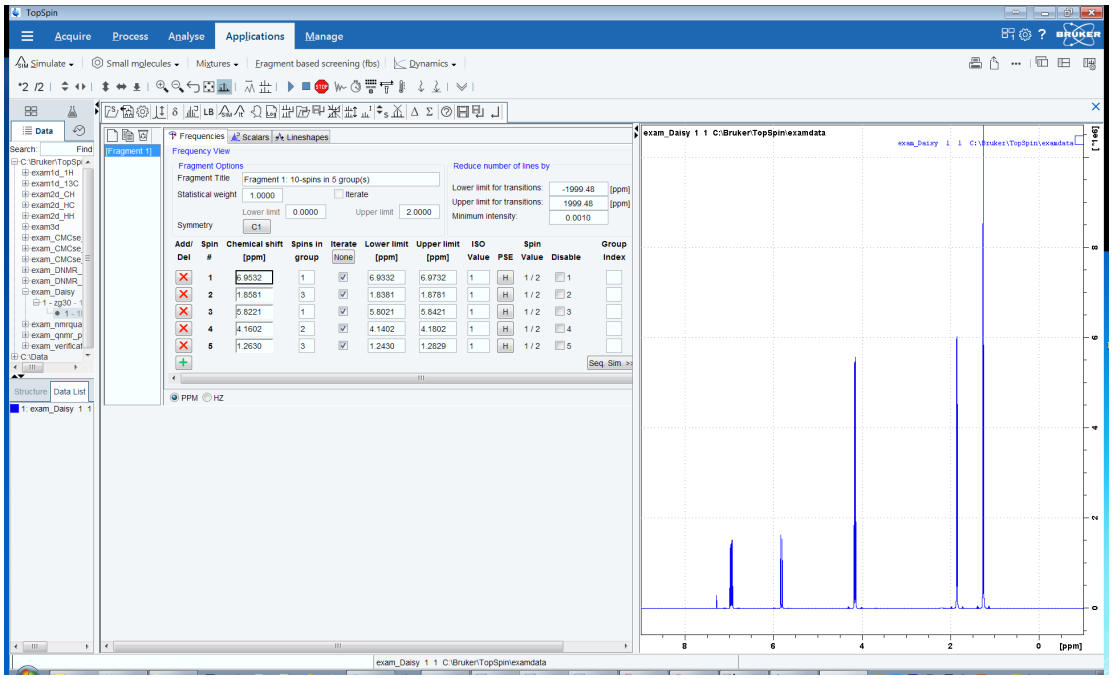
daisy - 1D simulation program

### DESCRIPTION

TopSpin offers the Daisy package for simulating spectra based on chemical shifts and coupling constants. Daisy supports the following input data:

- TopSpin multiplet analysis package
- Windaisy
- HAM
- ACD
- Perch

Daisy can be started as follows: Click **Applications | Simulate | Simulate/Iterate 1D Spectrum** [**daisy**]




For more information on **daisy**:

Click **Help | Manuals | Analysis and Simulation | Daisy**

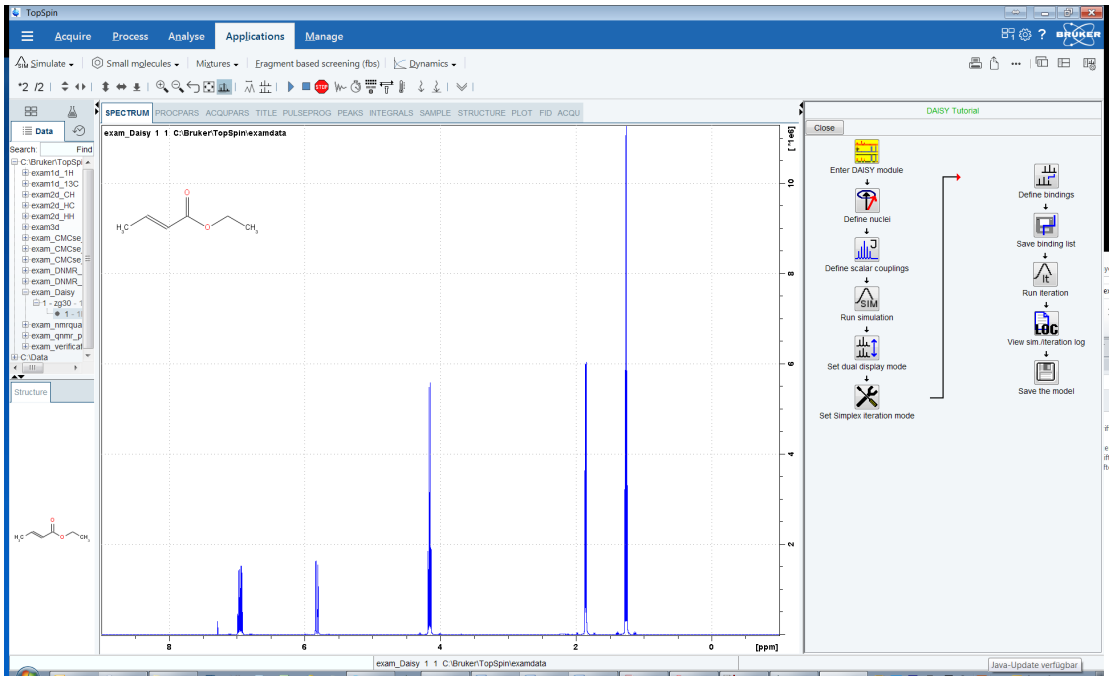
## 7.3 daisyguide

### NAME

daisyguide - Daisy tutorial

### DESCRIPTION

The command **daisyguide** opens the Daisy tutorial:




This guides you through the Daisy program.

Note that this can also be started with the command daisy.

For more information on **daisyguide**:

Click **Help | Manuals | Analysis and Simulation | Daisy**

### SEE ALSO

*daisy [▶ 209]*

## 7.4 dcon2d, dcon

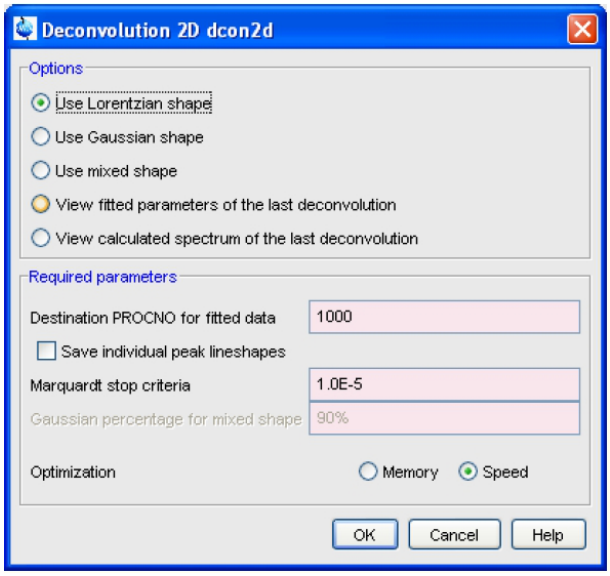
### NAME

dcon2d - Gaussian, Lorentzian or mixed deconvolution (2D)

dcon - Open deconvolution dialog box (1D,2D)

### DESCRIPTION

The command **dcon2d** performs deconvolution, fitting a Gaussian, Lorentzian or mixed function to the peaks in the displayed region. Before you start this command, you must select the desired region and perform peak picking (command **pp**). Then enter the command **dcon** or **dcon2d** to open the dialog box.




This offers several options, each of which selects a certain command for execution.

**Use Lorentzian shape**

This option deconvolves the spectrum by fitting a Lorentzian function to the peaks. It is typically used for overlapping peaks with a Lorentzian lineshape to determine the ratio of each individual peak.

**Use Gaussian shape**

This option deconvolves the spectrum by fitting a Gaussian function to the peaks. It is typically used for overlapping peaks with a Gaussian lineshape to determine the ratio of each individual peak.

**Use mixed shape**

This option deconvolves the spectrum by fitting a mixed Lorentzian/Gaussian function to the peaks. It requires the parameter **Gaussian percentage for mixed shape** to be set. A mixed shape deconvolution is typically used for spectra which cannot be approximated by a pure Lorentzian or a pure Gaussian lineshape.

**View fitted parameters of the last deconvolution**

This option shows the fitted parameters and peaks of the last performed deconvolution on the current dataset.

**View calculated spectrum of the last deconvolution**

This option shows the graphical result of the last deconvolution; the original and the deconvolved spectrum in multi-display mode.

The result of deconvolution is:

- The quality of the fit expressed by the minimized chi-square value.
- A list of peaks within the selected region, and for each peak its frequency, width, intensity and integral. This list is displayed on the screen.
- The fitted line shape, which is shown together with the original spectrum in multi-display mode.

Note that the deconvolution can be optimized for memory usage or speed. Furthermore, you can check the option *Save individual peak line shapes* to store the deconvolution result for each peak in a separate procno. All resulting procnos are shown superimposed in multi-display mode. As such, each deconvolved peak can be separately scaled and shifted.

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

*peaklist.xml* - peak list

*proc* - processing parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/1000/*

*2rr* - deconvolved processed 2D data (first individual peak)

*dcon2dpeaks.txt* - deconvolution parameters and peaks

*procs* - processing status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/1001/*

*2rr* - deconvolved processed 2D data (second individual peak)

*dcon2dpeaks.txt* - deconvolution parameters and peaks

*procs* - processing status parameters

etc.

**SEE ALSO**

*gdcon, ldcon, mdcon, ppp, dconpl, dcon* [▶ 215], *mldcon* [▶ 224]

## 7.5 dosy2d

**NAME**

dosy2d - Process DOSY dataset (2D)

**DESCRIPTION**

The command **dosy2d** processes a 2D DOSY dataset.

DOSY is a special representation of diffusion measurements. Instead of generating just numbers using the T1/T2 fitting package (i.e. diffusion coefficients and error values), the DOSY processing gives pseudo 2D data, where the F1 axis displays diffusion constants rather than NMR frequencies.

For more information on **dosy** :

click **Help | Manuals | Acquisition Application Manuals | Dosy**

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*difflist* - list of gradient amplitudes in Gauss/cm

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - 2D data processed in F2 only

*dosy* - DOSY processing parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - 2D processed data

*auditp.txt* - processing audit trail

**SEE ALSO**

*eddosy* [▷ 289], *dosy3d* [▷ 214]

## 7.6 dosy3d

**NAME**

dosy3d - Process DOSY dataset (3D)

**DESCRIPTION**

The command **dosy3d** processes a 3D DOSY dataset.

DOSY is a special representation of diffusion measurements. Instead of generating just numbers using the T1/T2 fitting package (i.e. diffusion coefficients and error values), the DOSY processing gives pseudo 3D data where the F2 or F1 axis displays diffusion constants rather than NMR frequencies.

For more information on **dosy3d** :

Click **Help | Manuals | Acquisition Application Manuals | Dosy**

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*difflist* - list of gradient amplitudes in Gauss/cm

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3rrr* - 3D data which are processed in F3 and F2 or in F3 and F1

*dosy* - DOSY processing parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3rrr* - 3D processed data

*auditp.txt* - processing audit trail

**SEE ALSO**

*eddosy* [▷ 289], *dosy2d* [▷ 213]

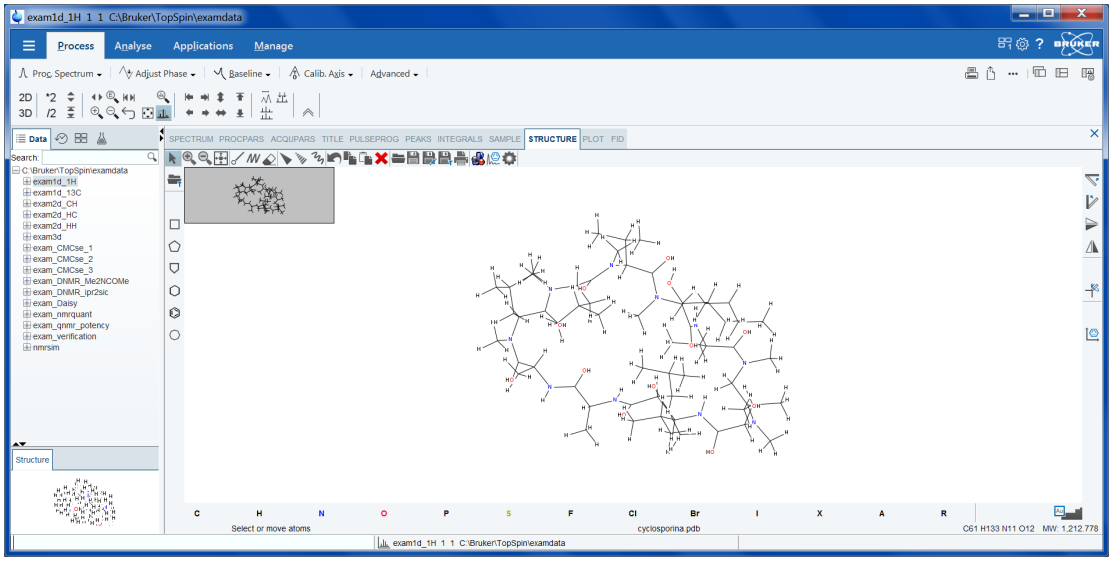
## 7.7 edstruc

**NAME**

edstruc - Open the 2D Molecule Structure Editor

### DESCRIPTION

The command **edstruc** opens the 2D Molecule Structure Editor. Entering this command is equivalent to clicking the **Structure** tab in the 2D data window and clicking the button **2D Editor**.




A full description of the 2D Structure Editor package can be found under:

**Help | Manuals | Analysis and Simulation | 2D Structure Editor**

### SEE ALSO

*jmol [▶ 220]*

## 7.8 gdcon, ldcon, mdcon, ppp, dconpl, dcon

### NAME

gdcon - Gaussian deconvolution (1D)

ldcon - Lorentzian deconvolution (1D)

mdcon - Mixed Gaussian/Lorentzian deconvolution (1D)

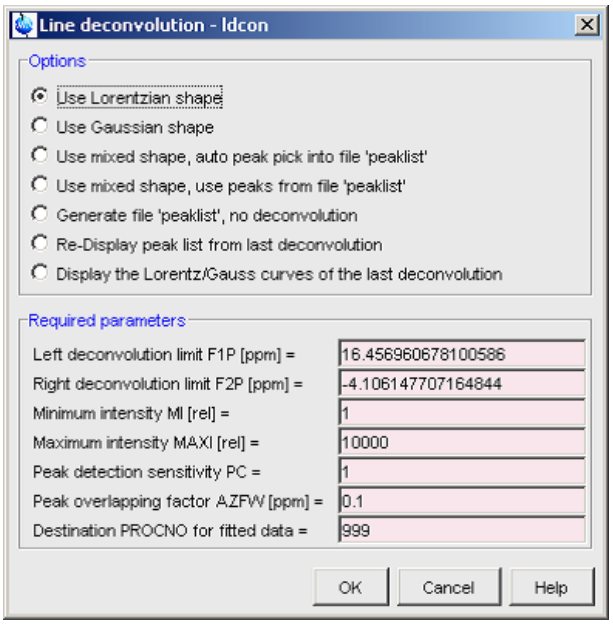
ppp - Generate peak list for deconvolution (1D)

dconpl - Show result of last deconvolution (1D)

dcon - Open deconvolution dialog box (1D,2D)

### DESCRIPTION

Deconvolution commands can be entered on the command line or started from the deconvolution dialog box, which is opened with the command **dcon**.




This offers several options, each of which selects a certain command for execution.

**Use Lorentzian shape**

This option selects the command **ldcon** for execution. It deconvolves the spectrum fitting a Lorentzian function to the peaks. It is typically used for overlapping peaks with a Lorentzian line shape to determine the ratio of each individual peak.

**Use Gaussian shape**

This option selects the command **gdcon** for execution. It deconvolves the spectrum by fitting a Gaussian function to the peaks. It is typically used for overlapping peaks with a Gaussian line shape to determine the ratio of each individual peak.

**Use mixed shape, auto peak pick into file *peaklist***

This option selects the command **mdcon auto** for execution. It first picks the peaks for deconvolution and stores them in the *peaklist* file. Then it deconvolves the spectrum by fitting a mixed Lorentzian/Gaussian function to these peaks. This command is typically used to deconvolve spectra which cannot be approximated by a pure Lorentzian or a pure Gaussian lineshape.

**Use mixed shape, use peaks from file *peaklist***

This option selects the command **mdcon** for execution. It works like **mdcon auto**, except that it uses an existing *peaklist* file. This file must have been created:

- by executing **mdcon auto**
- by executing **ppp**
- by executing **pps** and exporting the peak table (Peaks tab in data window) to the file *peaklist*.

**Generate peak list, no deconvolution**

This option selects the command **ppp** for execution. It picks the peaks for deconvolution and stores the result in the file *peaklist*. **ppp** is implicitly executed by **mdcon auto**.

### Re-Display peak list from last deconvolution

This option selects the command **dconpl** for execution. It shows the peak list (file *dconpeaks.txt*) which was created with the last deconvolution on the current dataset.

### Display the Lorentz/Gauss curves of the last deconvolution

This option selects the command **dconpl v** for execution. It shows the individually fitted peaks and their sum.

The deconvolution commands only work on the displayed region, as expressed by the parameters F1P and F2P. Furthermore, they select peaks according to the peak picking parameters MI, MAXI and PC. They also evaluate the parameter AZFW, which determines the minimum distance between two peaks for them to be fitted independently. Peaks which are less than AZFW ppm apart, are considered to be overlapping. As a rule of the thumb, set AZFW to ten times the width at half height of the signal.

The result of deconvolution is:

- the quality of the fit expressed by the minimized chi-square value
- a list of peaks within the plot region, and for each peak its frequency, width, intensity and area. This list is displayed on the screen.
- the fitted lineshape which is shown together with the original spectrum in multi-display mode.
- individually fitted peaks and their sum, as shown by **dconpl v**

All deconvolution commands can be started from the command line. In this case, they use the current values of the required parameters.

### Tailor Mixed Shape Deconvolution

Use peak list created by regular peak picking

Mixed deconvolution creates and uses its own peaklist. You can, however, force it: use the peaklist created with regular peak picking with the command **convertpeaklist**. To do that:

1. Perform peak picking, e.g. with **pps**.
2. Enter **convertpeaklist peaklist**
3. Enter **mdcon**.

### Select fit parameters for each individual peaks

The deconvolution fit parameters can be enabled/disabled for each individual peak. To do that:

Edit the file *peaklist* in the PROCNO directory of the dataset. At the end of a peak entry, you can specify three flags for the three parameters to be optimized; chemical shift, half width and amplitude:

0 = optimize this parameter

1 = do not optimize this parameter

Here is an example of a peaklist:

H

#frequency half width %gauss/100.

```
3304.390 4.52 0.0 0 0 0
3289.368 2.26 0.0 1 1 1
3262.410 7.91 0.0 0 1 0
3216.022 4.52 0.0 0 0 1
```

Signal 1: All 3 Parameters are optimized (default)

Signal 2: All three Parameters are not optimized

Signal 3:chemical shift and amplitude are optimized, half width is not
Signal 4: chemical shift and half width are optimized, amplitude is not

**INPUT PARAMETERS**

Set from the **dcon** dialog box, with **edp** or by typing **azfw**, **f1p** etc.:
AZFW - minimum distance in ppm for peaks to be fitted independently
F1P - low field (left) limit of the deconvolution region (= plot region)
F2P - high field (right) limit of the deconvolution region (= plot region)
MI - minimum relative intensity (cm) for peak picking
MAXI - maximum relative intensity (cm) for peak picking
PC - peak picking sensitivity

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*
*1r* - real processed 1D data
*dconpeaks.txt* - peak list (input of **dconpl)**
**peaklist** - peak list (input of **mdcon)**
**proc** - processing parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*
*1r* - real processed 1D data
*dconpeaks.txt* - peak list (output of **ldcon**, **gdcon**, **mdcon)**
**peaklist** - peak list (output of **ppp** and **mdcon auto)**
**procs** - processing status parameters

**USAGE IN AU PROGRAMS**

LDCON
GDCON
MDCON
PPP

**USAGE IN AU PROGRAMS**

dcon2d
For further information about deconvolution please look up the User Manual.

**SEE ALSO**

*dcon2d, dcon [▸ 211], mldcon [▸ 224]*

## 7.9 int2d, int3d, int

**NAME**

int2d - Calculate integrals (2D)
int3d - Calculate integrals (2D)
int - Open integral dialog box (1D, 2D, 3D)

## DESCRIPTION

The command **int2d** calculates 2D integrals. It opens the following dialog box:


Here you can set the minimum threshold for integration. You can enter:

- Enter the relative intensity: value between 0.0 and 1.0
- Enter the absolute intensity: value between 0.0 and YMax_p (processing status parameter).
- Click **Set to**... and choose from one of the following options:
  - *lowest contour level* - value of the lowest contour level (see **edlev**)
  - *value stored in MI* - value of the processing parameter MI (see **edp**)
  - *most recent MI used* - value used by last **int2d** command on any dataset

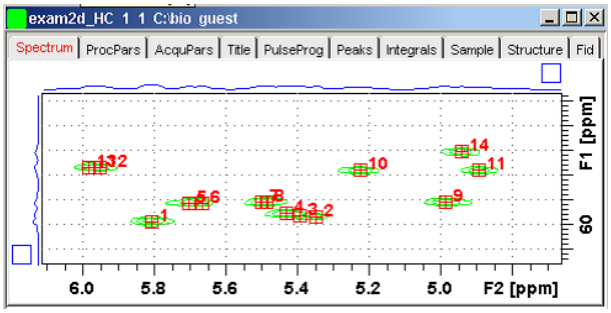
If you enter a relative value, the absolute value is automatically adjusted and vice versa. Setting the *most recent MI used* allows to compare integral value, e.g. of the NOE peak of a series of 2D spectra. Obviously, this only makes sense for spectra that are measured and processing under similar conditions.

The calculated integrals will be marked in the data field and can be listed by clicking the **Integrals** tab.

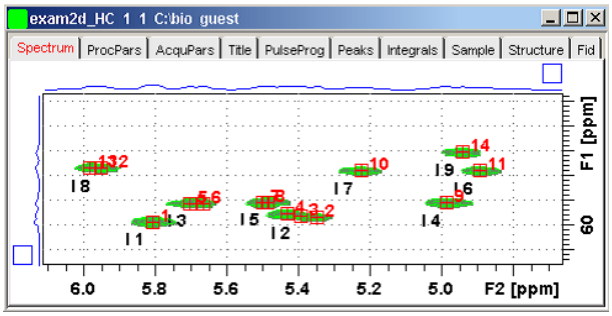
**int3d** is the same as **int2d**, except that it works on 3D data.

The **int** command can be used on 1D, 2D or 3D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

The following figure shows a region of peaks after peak picking.




The next figure shows the same region after 2D integration. Here you can see the integral labels and areas. The area color can be set in the user preferences (command **set**) as *Color of 3rd 1D spectrum*.




**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data (input of **int2d**)

*3rrr* - real processed 3D data (input of **int3d**)

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*integ_points.txt* - data points of integral regions

*integrals.txt* - peaks, integral regions and integral values

**SEE ALSO**

*li, lipp, lippf, int [▸ 221]*

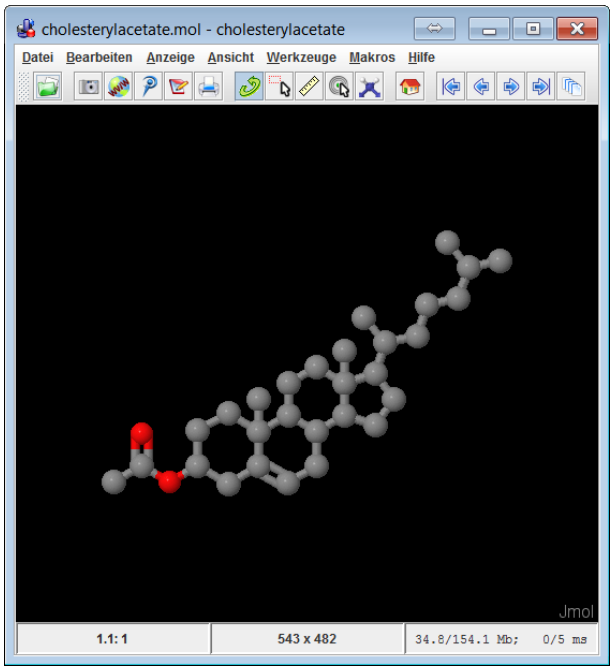
## 7.10 jmol

**NAME**

jmol - Open the Jmol molecule structure viewer

**DESCRIPTION**

The command **jmol** opens the *Jmol* molecule structure editor.




A description of the Jmol Molecule Viewer can be found under the Jmol *Help* menu, submenu *User Guide*.

### INPUT PARAMETERS

Set by the user with **eda** or by typing **chemstr**:

CHEMSTR - molecule structure filename

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*<name>* - molecule structure file

*acqu* - TopSpin acquisition parameters

*<tshome>/classes/prop/StructureSamples/** - molecule structure files

### SEE ALSO

*edstruc* [▷ 214]

## 7.11 li, lipp, lippf

### NAME

li - List integrals (1D)

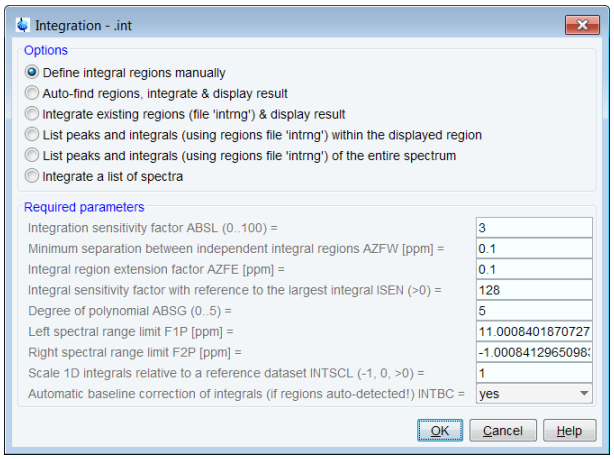
lipp - List integrals and peaks within F1P-F2P (1D)

lippf - List integrals and peaks of the full spectrum (1D)

int - Open integral dialog box (1D, 2D, 3D)

### DESCRIPTION

Integral commands can be started from the command line or from the integration dialog box.




The latter is opened with the command **int**.

This dialog box has several options, each of which selects a certain command for execution.

### Auto-find regions, integrate & display results

This option executes the command sequence **abs** - **li**. The command **abs** determines the integral regions creating the *intrng* file. The command **li** calculates the integral value for each integral region and shows the result in on the screen.

### Integrate existing regions and display results

This option executes the command **li**. This command calculates the integral value for each integral region and shows the result in on the screen.

### List peaks and integrals within the displayed region

This option executes the command **lipp**. It works like **li**, except that it also performs peak picking and shows a list of integral regions and peaks within the region F1P - F2P.

### List peaks and integrals of the entire spectrum

This option executes the command **lippf**. It works like **lipp**, except that it only determines the integrals and peaks over the entire spectrum.

The **li*** commands evaluates the parameter INTSCL if the regions have been determined interactively. For INTSCL ≠ –1, the current dataset is defined as reference dataset for integral scaling. For INTSCL = –1, the integrals of the current dataset are scaled relative to the reference dataset. As such, you can compare the areas of peaks in a series of experiments. Furthermore, the parameter INTBC is evaluated. For INTBC = yes, an automatic baseline correction (slope and bias) of the integrals is performed. This, however, is only done when the integral regions were determined with **abs**, not if they were determined interactively.

The **int** command can be used on 1D, 2D or 3D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

### INPUT PARAMETERS

Set with **edp**, from the **int** dialog box or by typing **intscl**, **intbc** etc.:

INTSCL - scale 1D integrals relative to a reference data set

INTBC - automatic baseline correction of integrals created by **abs**

F1P - low field (left) limit of the plot region in ppm (input for **lipp**)

F2P - high field (right) limit of the plot region in ppm (input for **lipp**)

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r* - real processed 1D data

*intrng* - 1D integral regions (created by **abs** or interactive integration)

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*integrals.txt* - ascii file containing the output of **li**

*integrals_lipp.txt* - ascii file containing the output of **lipp** *integrals_lippf.txt* - ascii file containing the output of **lippf**

### USAGE IN AU PROGRAMS

LI

LIPP

LIPPF

### SEE ALSO

*int2d, int3d, int* [▷ 218]

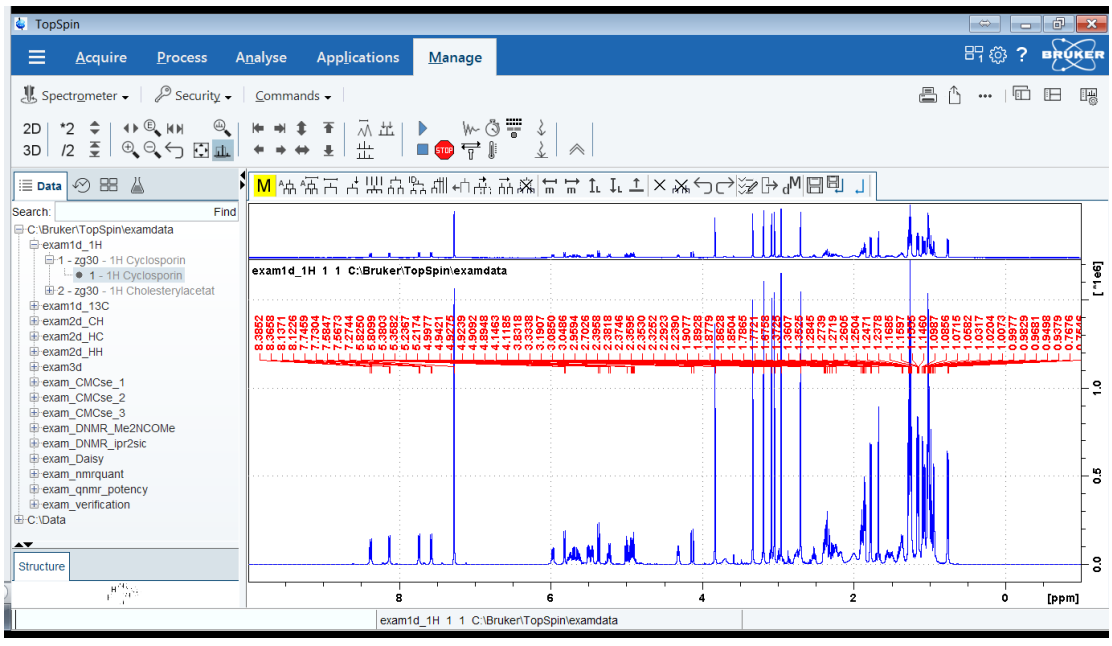
## 7.12 mana

### NAME

mana - Switch to multiplet analysis mode (1D)

### DESCRIPTION

The command **mana** switches to multiple analysis mode.




It can be started as follows:

- Click **Analyse | Multiplets**.

or

- enter **mana** in the command line

or

- open it from the Multiplet Analysis Guide (command **managuide**).

A full description of the Multiplet Analysis package can be found under:

**Help | Manuals | Analysis and Simulation | Structure Analysis Tools**

### SEE ALSO

*managuide* [▶ 224]

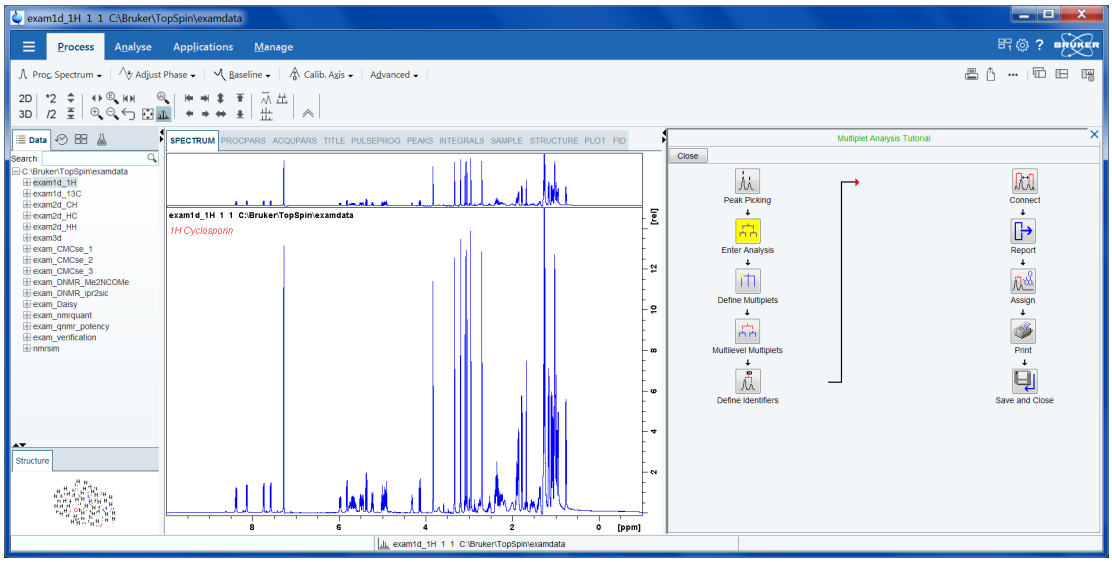
## 7.13 managuide

### NAME

managuide - Open the Multiplet Analysis Guide (1D)

### DESCRIPTION

The command **managuide** opens the Multiplet Analysis Guide which guides you through the multiplet analysis procedure.




A full description of the Multiplet Analysis package can be found under:

**Help | Manuals | Analysis and Simulation | Structure Analysis Tools**

### SEE ALSO

*mana* [▶ 223]

## 7.14 mldcon

### NAME

**mldcon** – Mixed Gaussian/Lorentzian deconvolution using machine learning based algorithm (1D)

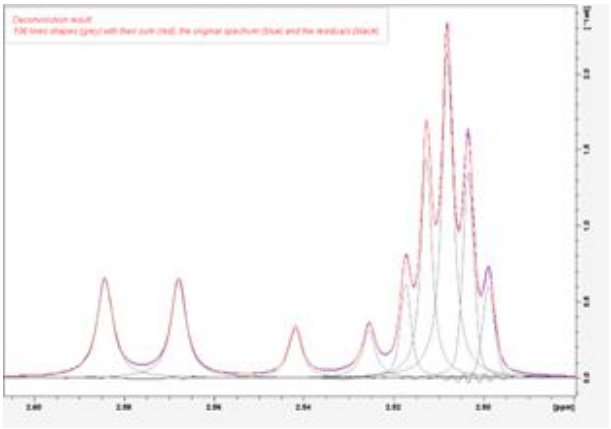
**DESCRIPTION**

The command **mldcon** deconvolves a 1D spectrum by fitting a mixed Lorentzian/Gaussian function to the peaks. The algorithm was developed to work on 1H 1D spectra, and it is based on a deep learning algorithm trained for peak detection. The algorithm requires a phase and baseline corrected 1D spectrum as input, and outputs a list of peaks that can be used to describe the spectrum. The algorithm works based on two deconvolution steps performed automatically: (i) initial peaks detection and peak parameters guess using deep learning, and (ii) accurate fit of peaks position, amplitude, width and line shape to match spectrum amplitudes.

The algorithm automatically optimizes the mixed Lorentzian/Gaussian ratio of each peak to obtain the best possible fit to the input data. The percentage of Gaussian contribution to the line shape is expressed by the lineshape parameter, and it can range from 0 (pure Lorentzian – line shape parameter equal to 0) to 100% (pure Gaussian – line shape parameter equal to 1). The ability to fit both Lorentzian and Gaussian line shapes makes the algorithm suitable to deconvolve spectra that can be described by either line shape or by a mix of the two.

The algorithm stores the deconvolution results as mldcon.csv file in the corresponding PROCNO directory. For each detected peak, the mldcon.csv file reports peak frequency in ppm, intensity in absolute values, full width at half maximum in Hz, line shape as ratio of Gaussian contribution to the total line shape, and absolute area. The peak positions and intensities are also stored as peaklist.xml file, to allow visualization of the predicted peak positions in the spectrum and for the Plot Editor.

After successful completion of the deconvolution algorithm, **mldcon** displays by default the result in a deconvolution visualization window. An example of the deconvolution visualization window is shown in the figure below. This window displays the input spectrum (in blue), the detected peaks (in grey), their sum (reconstructed spectrum, in red), and the difference between the input and the reconstructed spectrum (residuals, in black).



Use the command **mldcon -d** (or **mldcon -display**) to display the visualization window without running the deconvolution algorithm. This option builds a deconvolution visualization window using the results stored in the mldcon.csv file (i.e., it shows the results of a previous mldcon run). Use the -s (or -silent) option to run the deconvolution command silently, without automatically opening the deconvolution visualization window (for more details, see USAGE below).

The reconstructed spectrum is not saved by default. It is possible to store the reconstructed spectrum using the command **mldcon -sr** (or **-store**), which would run the deconvolution and save the reconstructed spectrum in the PROCNO 999 directory of the corresponding EXPNO. Use the additional -procno and -o options to respectively change the PROCNO directory and force overwriting (for more details, see USAGE below).

Note that the algorithm offers a range of options that can be used to tailor your results and personalize your workflow. Examples are: deconvolve only a given part of the spectrum, run the deconvolution using an existing user-defined guess instead of the deep learning

prediction (starts from a peaklist.xml or mldcon.csv file), increase the number of fit iterations to have a more accurate fitting, and more. Many of these options can also be combined together. An overview of all the options is given in the USAGE paragraph below.

**USAGE**

The following options run the full deconvolution algorithm (deep learning initial guess + accurate fitting). A slash separates matching options.

**mldcon** - Run the deconvolution and display the results in a deconvolution visualization window.

**mldcon -s/-silent** - Run the deconvolution silently (without opening the deconvolution visualization window).

**mldcon -r=<value1>,<value2>/-range=<value1>,<value2>** – Run the deconvolution in the region of the spectrum between <value1> and <value2> ppm. The **-r/-range** option cannot be combined with the -intrng and -e/-exclude options.

**mldcon -e=<value1>,<value2>/-exclude=<value1>,<value2>** – Run the deconvolution for the full spectrum leaving out the region between <value1> and <value2> ppm. The **-e/-exclude** option cannot be combined with the **-r/-range** and **-intrng** options.

**mldcon -intrng** – Run the deconvolution in the integral regions. The **- intrng** option cannot be combined with the **-r/-range** and **-e/-exclude** options.

**mldcon -sr/-store** – Run the deconvolution, display the results in a deconvolution visualization window, store the reconstructed spectrum in PROCNO 999.

**mldcon -sr/-store -procno=<value>** – Run the deconvolution, display the results in a deconvolution visualization window, store the reconstructed spectrum as PROCNO <value>. <value> must be a positive integer higher than 0.

**mldcon -sr/-store -o** – Run the deconvolution, display the results in a deconvolution visualization window, store the reconstructed spectrum in PROCNO 999 and force overwriting if the PROCNO directory already exists. The **-o** option can also be combined with the **-procno**.

**mldcon -sino=<value>** – Run the deconvolution, filter out peaks smaller than <value> times the noise level, display the results in a deconvolution visualization window. The default **-sino** value is 5. Note that the noise level is roughly estimated by the algorithm, and it might slightly deviate from the correct value

**mldcon -fit=<value>** – Run the deconvolution using <value> times the number of fitting iterations of the accurate fitting, display the results. Use this option to get a more accurate fitting if the default result has a large residual. <value> must be a positive integer higher than 0. (example: **mldcon -fit=2** will double the number of fitting iterations)

**mldcon -f** – Force the deconvolution algorithm to run on spectra that are not 1H spectra.

The following options run only the accurate fitting part of the deconvolution algorithm and allow users to define the initial peak parameters guess. A slash separates matching options.

**mldcon -pp** – Read the content of the peaklist.xml file (created e.g. using the manual peak picking window), run the deconvolution using peaks at the positions defined in the peaklist.xml file as initial guess, display the results. Note that if the peak positions are changed during the fitting, the peaklist.xml file will be updated.

**mldcon -csv** – Read the content of the mldcon.csv file, run the accurate fitting using the peaks defined in mldcon.csv as initial guess, display the results. Note that if the peak parameters are changed during the fitting the mldcon.csv file will be updated.

**mldcon -pp/-csv -tolppm=<value>** – Run the deconvolution restricting the chemical shift adjustment made by the accurate fitting to <value> ppm from the initial guess, display the results. This option can be combined with both **-csv** and **-pp**. (example **mldcon -pp -tolppm=0.01**: deviates peak positions max 0.01ppm from the input guess)

**mldcon -csv -tolamp=<value>** – Run the deconvolution restricting the amplitudes adjustment made by the accurate fitting to <value>% of the initial guess, display the results. This option can be combined with **-csv** only. (example **mldcon -csv -tolamp=10**: deviates peak amplitudes max 10% from the input guess)

**mldcon -csv -tolwidth=<value>** – Run the deconvolution restricting the line width adjustment made by the accurate fitting to <value>% of the initial guess, display the results. This option can be combined with **-csv** only. (example **mldcon -csv -tolwidth=10**: deviates peak line widths max 10% from the input guess)

**mldcon -csv -tolls=<value>** – Run the deconvolution restricting the lineshape parameter adjustment made by the accurate fitting to <value> units from the initial guess, display the results. This option can be combined with **-csv** only. (example **mldcon -csv -tolls=0.1**: deviates peak lineshape parameters max 0.1 units from the input guess)

The following options do not run the deconvolution algorithm. A slash separates equivalent options.

**mldcon -d/-display** – Open a deconvolution visualization window with the results stored in the mldcon.csv file.

**mldcon -srcsv/-storecsv** – Reconstruct the spectrum from the peak list in the mldcon.csv file, and store it as PROCNO 999. This option can be combined with the **-procno** and **-o** options to respectively change the PROCNO directory and force overwriting.

Note that most of the options can be used together to combine their results.

### INPUT FILES

*<dir>/<name>/<expno>/pdata/<procno>/*

1r, 1i - processed 1D data (frequency domain)

*mldcon.csv* (input of **mldcon -csv** and **mdcon -srcsv/storecsv**)

*peaklist.xml* (input of **mldcon -pp**)

### OUTPUT FILES

*<dir >/<name>/<expno>/pdata/<procno>/*

*mldcon.csv* – peak list created by **mldcon**

*peaklist.xml* – peak list for the plot editor

*auditp.txt* – processing audit trail

*1r* – real processed 1D data of the reconstructed spectrum (output of **mdcon -sr**/-store and **mdcon -srcsv/storecsv**)

### USAGE IN AU PROGRAMS

MLDCON(**option_string)

All the options in the USAGE paragraph can be used in automation and passed as option_string. Example MLDCON("-range=4,5 -fit=2"). Note that the function always requires an option string, so the default **mldcon** command can be used in AU programs using MLDCON("").

### SEE ALSO

*dcon2d, dcon [▶ 211]*; *gdcon, ldcon, mdcon, ppp, dconpl, dcon [▶ 215]*

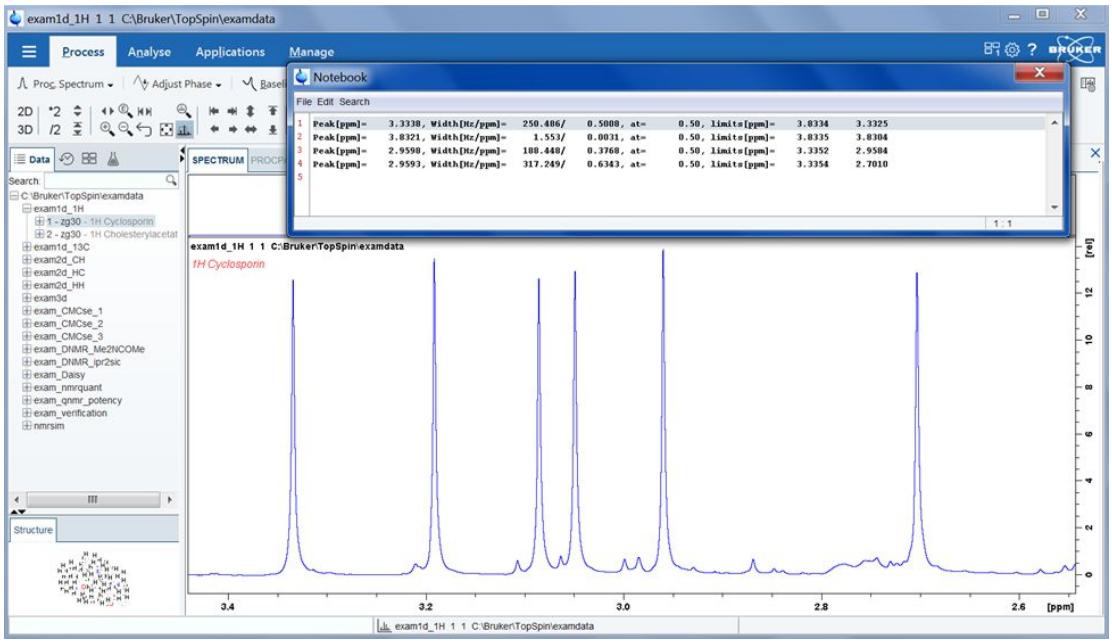
## 7.15 peakw

### NAME

peakw - Calculate width of highest peak in displayed region (1D)

### DESCRIPTION

The command **peakw** calculates the peak width at half height of the highest peak in the displayed region. The result is appended to the notebook and displayed on the screen:




The command can also be used with one argument: the height at which the width will be calculated.

**peakw <height>**

For example, **peakw 0.66** calculates the width of the highest peak in the displayed region at 66% of the height.

### OUTPUT FILES

*<userprop>/notebook.txt* - notebook text file

### SEE ALSO

*nbook* [▶ 367]

## 7.16 pps, ppf, ppl, pph, ppj, pp

### NAME

pps - Perform peak picking on displayed region

ppf - Perform peak picking on full spectrum

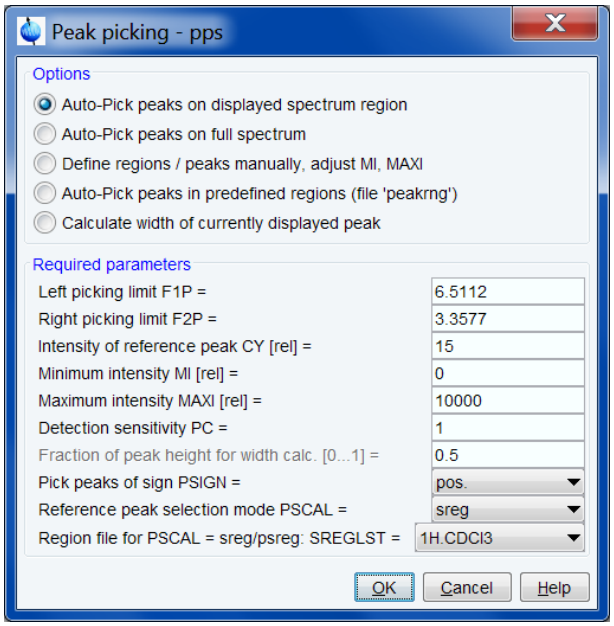
ppl - Perform peak picking in predefined regions

pph - Perform peak picking and also show an intensity histogram

ppj - Perform peak picking and store peaks in JCAMP-DX forma

## DESCRIPTION

Peak picking commands can be started from the command line or from the peak picking dialog box:




All peak picking commands open the dialog box with the corresponding option selected. The command **pp**, however, selects the last used option.

### Auto-Pick peaks on displayed spectrum region

This option selects the command **pps** for execution. It determines all peaks within the displayed region. The following table shows an example of its output.

| # | ADDRESS | FREQUENCY | | INTENSITY |
|---|---|---|---|---|
| | | [Hz] | [PPM] | |
| 1 | 648.7 | 3698.825 | 7.3995 | 0.17 |
| 2 | 658.4 | 3687.649 | 7.3771 | 0.21 |

The peak list is created according to several criteria which are determined by various parameters. A data point is added to the peak list if:

- its intensity is higher than its two neighboring points
- its relative intensity is smaller than MAXI
- its relative intensity is larger than MI
- its absolute intensity is larger than PC*noise
- it lies within the displayed region as expressed by F2P and F1P

Where MAXI, MI and PC are processing parameters and noise is calculated from the first 32th part of the spectrum.

The values of MI and MAXI must be chosen in relation to the plot parameter CY; the intensity (in cm) of the reference peak. The reference peak is the highest peak in the spectrum or in a certain part of it. The spectral region which contains reference peak, is determined by the parameter PSCAL. For PSCAL = global, this is entire spectrum. The next table shows all possible values of PSCAL and the corresponding regions. For PSCAL = ireg or pireg, the *reg* file is interpreted. To create a *reg* file click  to switch to integration mode, click  and select **Save regions to reg**. The *reg* file can be viewed or edited with the command **edmisc reg**.

| PSCAL | Peak used as reference for vertical scaling |
|---|---|
| *global* | The highest peak of the entire spectrum. |
| *preg* | The highest peak within the plot region. |
| *ireg* | The highest peak within the regions specified in the *reg* file. If it does not exist, *global* is used. |
| pireg | as *ireg*, but the peak must also lie within the plot region. |
| *sreg* | The highest peak in the regions specified in scaling region file. This file is specified by the parameter SREGLST. If SREGLST is not set or it specifies a file which does not exist, *global* is used. |
| *psreg* | as *sreg* but the peak must also lie within the plot region. |
| *noise* | The intensity height of the noise of the spectrum. |

For PSCAL = sreg or psreg, the scaling region file is interpreted. This is used to make sure the solvent peak is not used as reference. The name of a scaling region file is typically of the form NUCLEUS.SOLVENT, e.g. 1H.CDCl3. For most common nucleus/solvent combinations, a signal region file is delivered with TopSpin. In several 1D standard parameter sets which are used during automation, PSCAL is set to *sreg* and SREGLST to NUCLEUS.SOLVENT as defined by the parameters NUCLEUS and SOLVENT.

**pps** evaluates the parameter PSIGN which can take three possible value:

- pos - only positive peaks appear in the list
- neg - only negative peaks appear in the list
- both - both positive and negative peaks appear in the list

### Auto-Pick peaks on full spectrum

This option selects the command **ppf** for execution. It works like **pps** except that it picks peaks on the full spectrum.

### Auto-Pick peaks in predefined regions (file *peakrng*)

This option selects the command **ppl** for execution. It picks the peaks in predefined regions. To define those regions:

1. Click **Define regions/peaks manually** in the peaks dialog box or click  in the toolbar to switch to peak picking mode.
2. Click  and drag the cursor inside the data window to define the regions.
3. Right-click inside the data window and select **Pick Peaks on ranges** or enter **ppl** on the command line.

### Like 1st option but peak list with histogram

This option selects the command **pph** for execution. It works like **pps**, except that it also shows an intensity histogram. This allows to get a quick overview over the intensity distribution.

**Like 1st option but peak in JCAMP format**

This option selects the command **ppj** for execution. It works like **pps**, except that the peak list is stored in JCAMP-DX format in the file *pp.dx*. This file resides in the processed data directory and can be used for external programs which require JCAMP peak lists. As the file created by **tojdx** it contains the acquisition and processing parameters but instead of data points it contains a list of peaks. The last part of the file *pp.dx* looks like:

| **##NPOINTS= 4** | |
|---|---|
| ##PEAK TABLE= (XY..XY) | |
| 2.3241 | 1.58 |
| 2.2962 | 1.18 |
| 1.9943 | 10.00 |
| 1.8725 | 1.36 |

The **pp** command can be used on 1D, 2D or 3D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

**INPUT PARAMETERS**

Set by the user with **edp** or by typing **mi**, **maxi** etc.:

MI - minimum relative intensity (cm)

MAXI - maximum relative intensity (cm)

PC - peak picking sensitivity

PSIGN - peak sign (pos, neg, or both)

PSCAL - determines the region with the reference peak for vertical scaling

SREGLST - name of the scaling region file used for PSCAL = sreg/psreg

ASSFAC - assign the highest or second highest peak as reference for scaling

ASSWID - region excluded from second highest peak search

Set by the user with **edp** or by typing **f1p**, **f2p** etc.:

F1P - low field (left) limit of the plot region in ppm

F2P - high field (right) limit of the plot region in ppm

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r* - real processed 1D data

*proc* - processing parameters

*reg* - region with the reference peak for PSCAL = *ireg* or *pireg*

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*peaks* - peak list containing all peaks in the entire spectrum

*peaklist.xml* - peak list created by **pp** and **pps** for the Plot Editor

*peak.txt* - peak list created by **pp** and **pps** ( TopSpin 2.0 and older) or by **convertpeaklist** ( TopSpin 2.1 and newer)

*peakhist.txt* - peak list with histogram, created by **pph**

*pp.dx* - peak list in JCAMP-DX format created by **ppj**

### USAGE IN AU PROGRAMS

PP

PPL

PPH

PPJ

### SEE ALSO

*peakw* [▸ 209], *gdcon, ldcon, mdcon, ppp, dconpl, dcon* [▸ 209], *li, lipp, lippf, int* [▸ 209]

## 7.17 ppd

### NAME

ppd - Perform peak picking with derivative-based algorithm

### DESCRIPTION

The command **ppd** can be useful to pick peak shoulders which are not found by other peak picking commands.


### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r* - real processed 1D data

*proc* - processing parameters

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*peaklist.xml* - peak list created for the Plot Editor

### SEE ALSO

*pps, ppf, ppl, pph, ppj, pp* [▶ 209]

## 7.18 pp2d

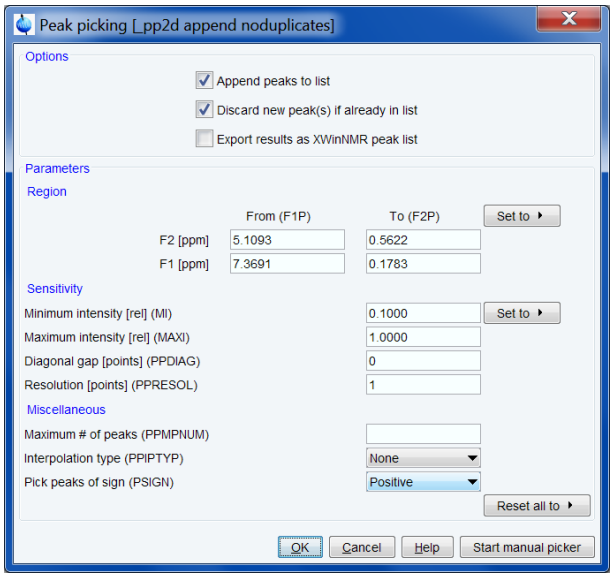
### NAME

pp2d - Perform peak picking (2D)

pp - Open peak picking control dialog (1D, 2D, 3D)

### DESCRIPTION

2D peak picking can be started from the command line or from the peak picking dialog box. The latter can be opened with the command **pp:**




In this dialog window, you can set the following options:

- **Append peaks to list**: When it is checked, the found peaks are appended to a possibly existing list. When it is unchecked, a new list is created [**pp2d append**]
- **Discard new peak(s) if already in list**: Check this option to avoid duplicate peaks [**pp2d noduplicates**]
- **Export results as XwinNmr peak** list: In addition to TopSpin XML format, the result is also stored in XWIN-NMR format (file *peak.txt*) [**pp2d txt**]. This file is typically used with XWIN-NMR AU programs.

Furthermore, you can set the following peak picking parameters:

#### Region parameters

Here you can set the region limits **From (F1P)** and **To (F2P)** for both the F2 and F1 direction. Only peaks within this region will be picked. Note that the limits can be specified in the text fields or set with the button **Set to**. The latter allows you to select from:

- *Full range* - full spectrum
- *Displayed range* - range displayed in the data window
- *Range defined by stored parameters* - range stored in parameters F1P/F2P

- *Most recent range stored in peak list* - range on which last automatic peak picking was done (Only active when peak picking was already done).

**Sensitivity parameters**

Here you can set the peak picking parameters MI and MAXI which are also used for 1D peak picking. Note that MI can also be set interactively with the button **Set to**, to *the lowest contour level*, *the current value of MI* or *the most recent value stored in the peak list*. Furthermore, you can set the parameters:

- PPDIAG - diagonal gap; minimum distance between picked peaks and diagonal signals. Mainly used for homonuclear spectra.
- PPRESOL - peak picking resolution

**Miscellaneous parameters**

Here you can set the following parameters:

- PPMPNUM: Maximum number of picked peaks. Note that 0 or no value specified means unlimited.
- PPIPTYP: Peak picking interpolation type (parabolic or none).
- PSIGN: The sign of the picked peaks (positive, negative or both).

To start peak picking:

- Click **OK**.

The peak picking progress will be shown in the TopSpin status line. When the peak picking process has finished:

- The number of found peaks is displayed in the status line. Note that if the option **Append peaks to list** is checked, only additional peaks are reported as found.
- The peaks and parameters are stored in the processing directory.

To view the peak list, click the **Peaks** tab of the data window toolbar.

The peak picking dialog window has two extra buttons:

- **Reset all to**: Allows you to reset all parameters to the stored parameters or to the most recent values stored in the peak list. Note that the stored parameters and the parameters in the peak list can be different since parameters can also be set with **edp** or from the command line. However, right after peak picking they are the same.
- **Start manual picker**: To switch to interactive peak picking mode (equivalent to clicking  in the TopSpin upper toolbar).

The options specified in square brackets in the dialog window and further options can also be specified on the command line. For example:

- **pp append**: Open peak picking dialog with the **Append..** option checked.
- **pp noduplicates**: Open peak picking dialog with the **Discard new peaks..** option checked.
- **pp silent**: Perform peak picking on the displayed region with the last stored options (no dialog). Equivalent to the command **pps**.
- **pp nodia**: Perform peak picking on the last stored region with the last stored options (no dialog).
- **pp append noduplicates nodia**: Perform peak picking on the last stored region with the specified options.

The **pp** command can be used on 1D, 2D or 3D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

**INPUT PARAMETERS**

Set from the **pp** dialog box, with **edp** or by typing **f1p**, **mi** etc.:

F1P - low field (left) limit of the peak picking region in F2 and F1

F2P - high field (left) limit of the deconvolution region F2 and F1

MI - minimum relative intensity (cm)

MAXI - maximum relative intensity (cm)

PC - peak picking sensitivity

PPDIAG - diagonal gap; minimum distance to spectrum diagonal

PPRESOL - peak picking resolution

PPMPNUM - maximum number of picked peaks

PPIPTYP - interpolation type

PSIGN - peak sign (pos, neg, or both)

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*2rr* - real processed 2D data

*proc* - F2 processing parameters, including peak picking parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*procs* - F2 processing parameters, including peak picking parameters

*peaklist.xml* - 2D peak list in XML format

*peak.txt* - 2D peak list in TXT format

*<userhome>/<.topspin-hostname/prop/*

*globals.prop* - peak picking setup

**USAGE IN AU PROGRAMS**

PP2D

**SEE ALSO**

*pp3d [▸ 235]*, *pps, ppf, ppl, pph, ppj, pp [▸ 228]*

## 7.19 pp3d

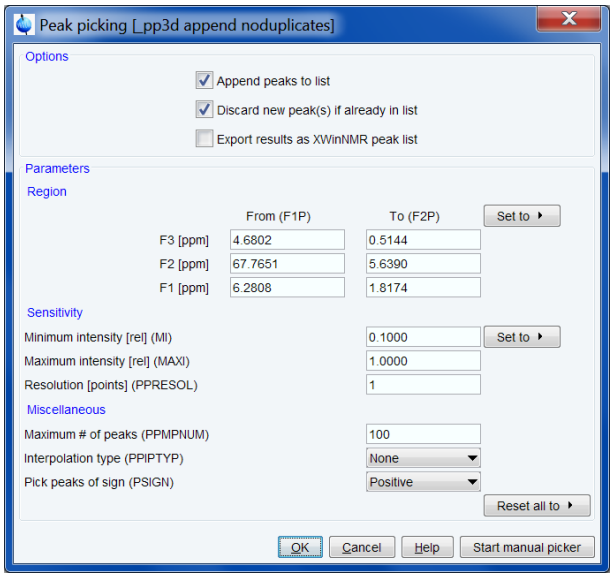
**NAME**

pp3d - Perform peak picking (3D)

pp - Open peak picking control dialog (1D, 2D, 3D)

**DESCRIPTION**

3D peak picking can be started from the command line or from the peak picking dialog box. The latter can be opened with the command **pp:**




In this dialog window, you can set the following options:

- **Append peaks to list**: When it is checked, the found peaks are appended to a possibly existing list. When it is unchecked, a new list is created [**pp3d append**].
- **Discard new peak(s) if already in list**: Check this option to avoid duplicate peaks [**pp3d noduplicates**].
- **Export results as XwinNmr peak list** In addition to TopSpin XML format, the result is also stored in XWIN-NMR format (file *peak.txt*) [**pp3d txt**]. This file is typically used with XWIN-NMR AU programs.

Furthermore, you can set the following peak picking parameters:

### Region parameters

Here you can set the region limits **From (F1P)** and **To (F2P)** for the F3, F2 and F1 direction. Only peaks within this region will be picked. Note that the limits can be specified in the text fields or set with the button **Set to** to:

- **Full range** - full spectrum.
- **Displayed range** - range displayed in the data window.
- **Range defined by stored parameters** - range stored in parameters F1P/F2P (To store displayed region: right-click in the data window and select **Save display region to**).
- **Most recent range stored in peak list** - range on which last automatic peak picking was done (Only active when peak picking was already done).

### Sensitivity parameters

Here you can set the peak picking parameters MI and MAXI, which are also used for 1D peak picking. Note that MI can also be interactively set to the current value of MI, or the lowest contour level, using the **Set to** button. Furthermore, the parameter PPRESOL for peak picking resolution can be set.

### Miscellaneous parameters

Here you can set the following parameters:

- **PPMPNUM** - Maximum number of picked peaks. Note that 0 or no value specified means unlimited.
- **PPIPTYP** - Peak picking interpolation type (parabolic or none).
- **PSIGN** - The sign of the picked peaks (positive, negative or both).

To start peak picking click **OK**.

The peak picking progress will be shown in the TopSpin status line. When the peak picking process has finished:

- The number of found peaks is displayed in the status line. Note that if the option **Append peaks to list** is checked, only additional peaks are reported as found.
- The peaks and parameters are stored in the processing directory.

To view the peak list, click the **Peaks** tab of the data window toolbar.

The peak picking dialog window has two extra buttons:

- **Reset all to**: Allows you to reset all parameters to the stored parameters or to the most recent values stored in the peak list. Note that the stored parameters and the parameters in the peak list can be different since parameters can also be set with **edp** or from the command line. However, right after peak picking they are the same.
- **Start manual picker**: To switch to interactive peak [picking mode (equivalent to clicking  in the TopSpin upper toolbar).

The options specified in square brackets in the dialog window and further options can also be specified on the command line. For example:

- **pp append**: Open peak picking dialog with the **Append..** option checked.
- **pp noduplicates**: Open peak picking dialog with the **Discard new peaks..** option checked.
- **pp silent**: Perform peak picking on the displayed region with the last stored options (no dialog). Equivalent to the command **pps**.
- **pp nodia**: Perform peak picking on the last stored region with the last stored options (no dialog).
- **pp append noduplicates nodia**: Perform peak picking on the last stored region with the specified options.

The **pp** command can be used on 1D, 2D or 3D data. It recognizes the data dimensionality and opens a dialog box with the appropriate options and parameters.

## INPUT PARAMETERS

Set from the **pp** dialog box, with **edp** or by typing **f1p**, **mi** etc.:

F1P - low field (left) limit of the peak picking region in F3, F2 and F1

F2P - high field (left) limit of the deconvolution region F3, F2 and F1

MI - minimum relative intensity (cm)

MAXI - maximum relative intensity (cm)PC - peak picking sensitivity

PPRESOL - peak picking resolution

PPMPNUM - maximum number of picked peaks

PPIPTYP - Interpolation type

PSIGN - peak sign (pos, neg, or both)

## INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*3rrr* - real processed 3D data

*proc* - F3 processing parameters, including peak picking parameters

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*procs* - F3 processing parameters, including peak picking parameters

*peaklist.xml* - 3D peak list in XML format

*peak.txt* - 3D peak list in TXT format

*<userhome>/<.topspin-hostname/prop/*

*globals.prop* - peak picking setup

### SEE ALSO

*pp2d, pp* [▸ 209], *pps, ppf, ppl, pph, ppj, pp* [▸ 209]

## 7.20 sino

### NAME

sino - Calculate signal to noise ratio (1D)

### SYNTAX

sino [real] [noprint]

### DESCRIPTION

The command **sino** calculates the signal to noise ratio of a 1D spectrum according to the formula:

$$SINO = \frac{maxval}{2 \cdot noise}$$

Where *maxval* is highest intensity in the signal region. The signal region is determined by the processing parameters SIGF1 and SIGF2. If SIGF1 = SIGF2, the signal region is defined by:

- The entire spectrum without the first 16th part of the data points, unless the scaling region file is defined (see next bullet item).
- The regions defined in the solvent table entry NUC1.SOLVENT where NUC1 and SOLVENT are acquisition status parameters.

The factor *noise* is calculated according to the algorithm shown in:

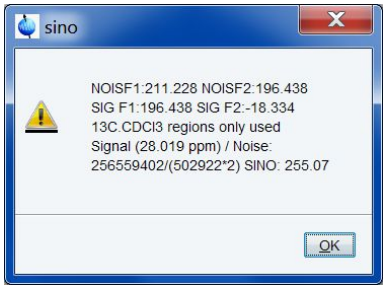
$$noise = \sqrt{\frac{\sum_{i=-n}^{n} y(i)^2 - \frac{1}{N}\left(\left(\sum_{i=-n}^{n} y(i)\right)^2 + \frac{3 \cdot \left(\sum_{i=1}^{n} i(y(i) - y(-i))\right)^2}{N^2 - 1}\right)}{N - 1}}$$

Where N is the total number of points in the noise region, n = (N-1)/2, and y(i) is the nth point in the noise region. The limits of the noise region are determined by the processing parameters NOISF1 and NOISF2. If they are equal, the first 16th part of the spectrum is used as the noise region.

The parameters SIGF1, SIGF2, NOISF1 and NOISF2 can be set from the command line, from the **Procpars** tab (command **edp**) or, interactively, in Signal/Noise display mode. The latter can be entered by clicking **Analyse | SiNo | Signal/Noise ratio: Calculate (sino)** or by entering **.sino** on the command line.

**sino** internally performs a peak picking to determine the highest peak in the signal region.

The result of **sino** appears on the screen, for example:




**sino** without any argument calculates the signal to noise on the real part of data (1r)

**sino real** performs in the same way as **sino**: only real part of spectra is used

**sino mag** first performs a magnitude calculation of the signal region and then uses this data for the calculation. This makes the S/N value independent on a signal phase.

**sino noprint** does not show the result on the screen. The *noprint* option is automatically set when **sino** is part of an AU program. The result of **sino** is stored in the processing status parameter SINO which can be viewed with **s sino** or **dpp**.

The **noprint** option can be combined with the options described above: **sino mag noprint** would calculate the S/N on magnitude data without showing the dialog.

The parameter SINO exists as processing parameter (**edp**) and as processing status parameter (**dpp**) and they have different functions. The latter is used to store the result of the command **sino** as discussed above. The former can be used to specify a signal to noise ratio which must be reached in an acquisition (see the parameter SINO in *List of Processing Parameters [▷ 20]* and the AU program **au_zgsino**).

## INPUT PARAMETERS

Set in **.sino** display mode, with **edp** or by typing **noisf1**, **noisef2** etc.:

NOISF1 - low field (left) limit of the noise region

NOISF2 - high field (right) limit of the noise region

SIGF1 - low field (left) limit of the signal region

SIGF2 - high field (right) limit of the signal region

Set by the acquisition, can be viewed with **dpa** or by typing **s nuc1** etc.:

NUC1 - observe nucleus

SOLVENT - sample solvent

## OUTPUT PARAMETERS

Can be viewed with **dpp** or by typing **s sino** :

SINO - signal to noise ratio

## INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r* - real processed 1D data

*1i* - imaginary processed data (not used for **sino real**)

*proc* - processing parameters

**OUPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*procs* - processing status parameters

**USAGE IN AU PROGRAMS**

SINO

**SEE ALSO**

*abs, absf, absd, bas [ 42], Analysis Commands [ 209], List of Processing Parameters [ 20]*

## 7.21 sino2d

**NAME**

sino2d

**SYNTAX**

sino2d

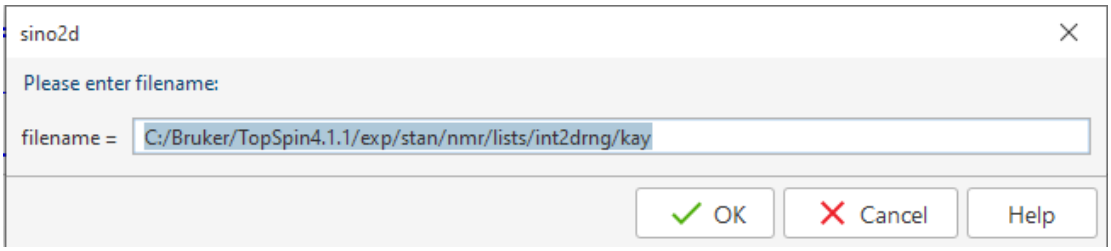
**Description**

sino2d - Signal/Noise calculation for 2D spectra

The command is implemented as a Jython script sino2d.py.

You may start it either either by typing **xpy sino2d** or **sino2d** in the command line.

sino2d requests the name of region file:




Enter the filename (e.g. by completing the proposed path) and click **OK**.

The result is displayed in a text window. It shows detailed information about noise calculation.

**INPUT PARAMETERS**

2D sino requires a 2D signal region and a 2D noise region stored in an "int2drng" formatted file according to the following example:

```
0 0
a 1024 165 249 130.750166 117.206324
1024 279 314 7.416665 7.140323
a 1024 529 639 71.886546 54.175368
```

```
1024 375 421 6.667867 6.311296
```

Please see the format description below.

You may setup the file by hand, by a program, or most comfortably in the 2D integration mode:

1. Open a 2D spectrum in TopSpin.
2. Enter interactive integration mode using a menu entry or tool button or by entering the command **.int**.
3. Click on the tool button **delete all regions** to start from scratch.
4. Click on the tool button **define new integration region.**
5. Drag a region around a signal while keeping the left mouse button depressed. When the button is released, a popup menu is opened. Click on an **integrate** entry, e.g. the first one (which one doesn't matter).
6. Move the mouse to a signal-free region and drag again the mouse to mark the region. Again click on an **integrate** entry when releasing the left mouse button.
7. Click on the icon **Export integration regions**. The **wmisc** window is opened. Click on **Write new....** Enter a filename. The file is stored in the *.../list/intrng2d* directory which can be inspected using the **rmisc** command.

Format description of int2drng file:

```
Mode SI_F1 row1 row2 row1(ppm) row2(ppm)
SI_F2 col1 col2 col1(ppm) col2(ppm)
```

**SEE ALSO**

*sino* [▶ 238]

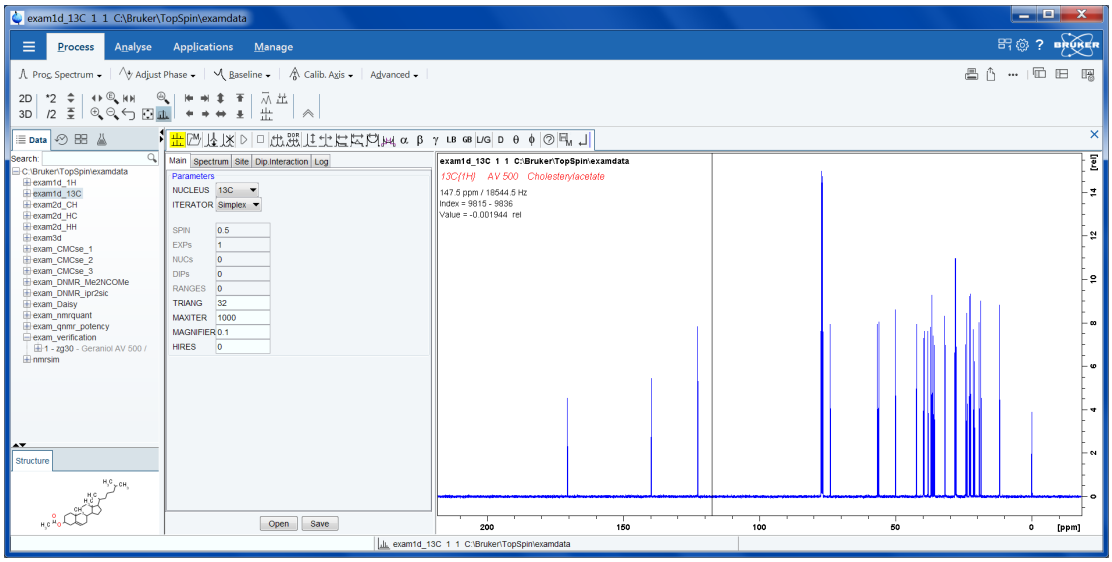
## 7.22 sola

**NAME**

sola - Switch to solids line shape analysis mode.

**DESCRIPTION**

The command **sola** switches to solids line shape analysis mode.




This procedure is completely described in the TopSpin Users Guide. To open this:
Click **Help | Manuals | Analysis and Simulation | Structure Analysis Tools**

### SEE ALSO

*solaguide [▶ 242]*

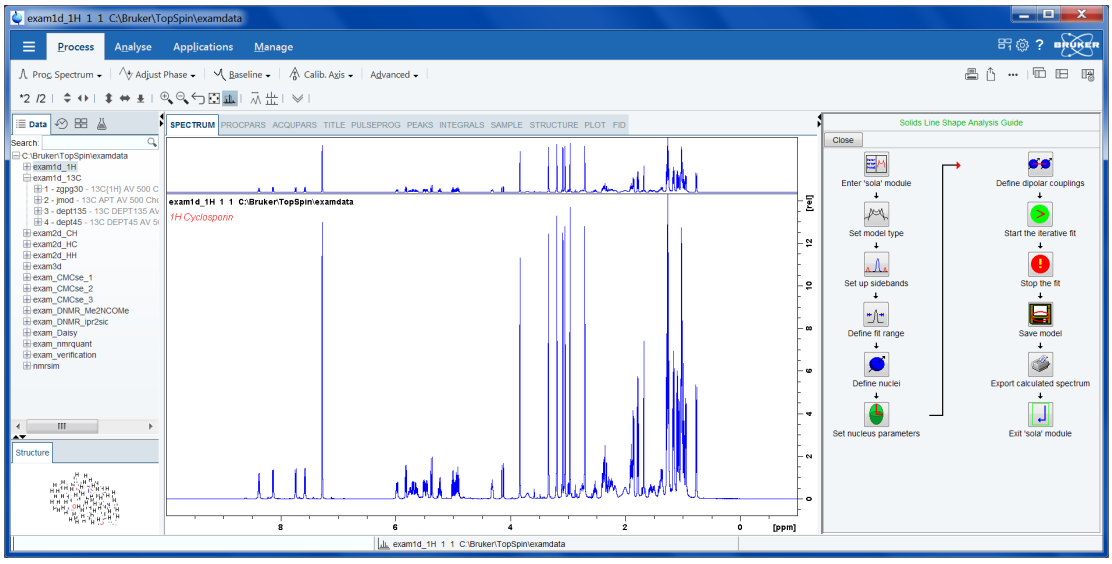
## 7.23 solaguide

### NAME

solaguide - Open the solids analysis guide (1D)

### DESCRIPTION

The command **solaguide** opens a dialog box with a workflow for Solids Line Shape Analysis.




This procedure is completely described in the TopSpin Users Guide. To open this:

Click **Help | Manuals | Analysis and Simulation | Structure Analysis Tools**

### SEE ALSO

*sola [▶ 241]*

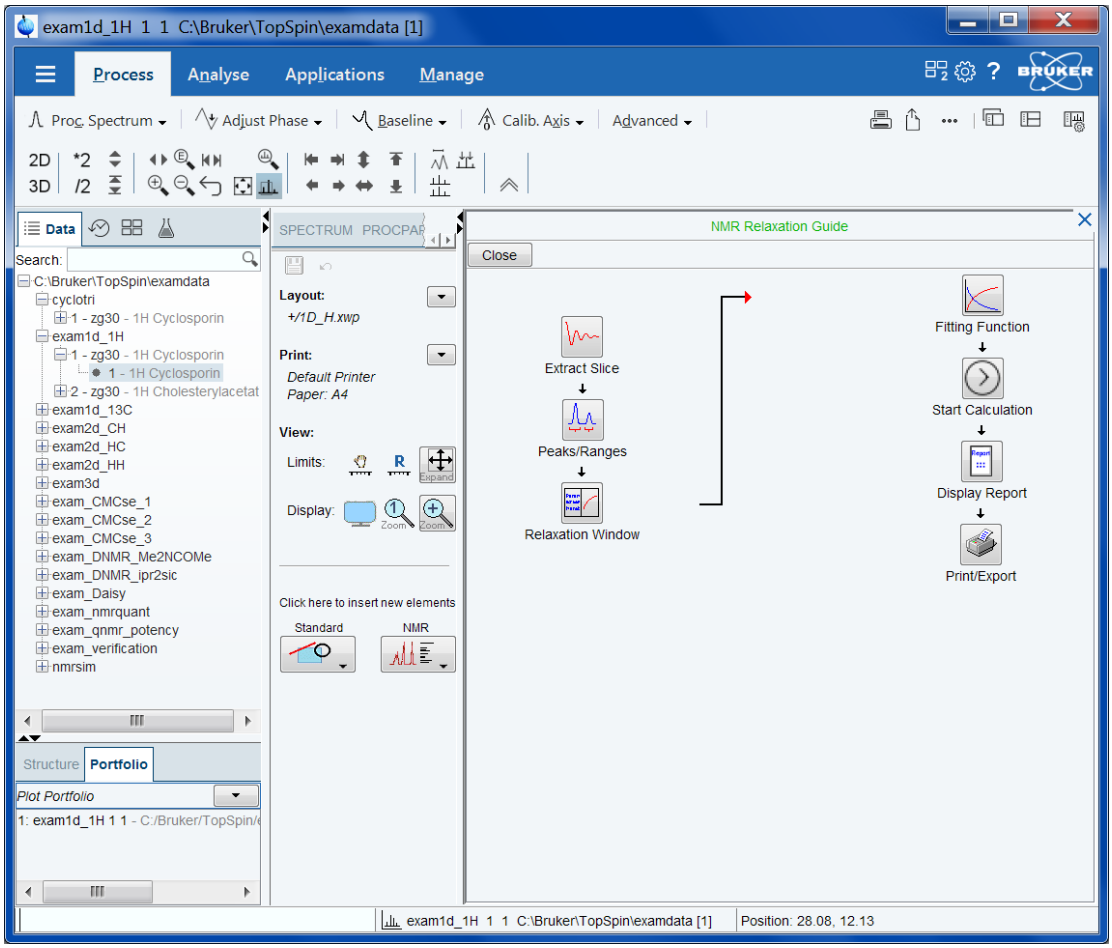
## 7.24 t1guide

### NAME

t1guide - Open the relaxation analysis guide (2D)

### DESCRIPTION

The command **t1guide** opens a dialog box with a workflow for relaxation analysis including T1/T2.




This procedure is completely described in the TopSpin Users Guide. To open this:

Click **Help | Manuals | General | User Manual**.

# 8 Print/Export Commands

This chapter describes TopSpin print, plot and export commands. Printing can be done directly from the TopSpin interface or from the Plot Editor. The data window can be exported into a graphics file. Commands are available for setting the plot title and, for 2D and 3D data, the contour levels.

## 8.1 autoplot

### NAME

autoplot - Plot data according to Plot Editor layout (1D, 2D)

### DESCRIPTION

The command **autoplot** plots the current dataset according to a Plot Editor layout. The layout must be specified with the processing parameter LAYOUT. This layout can be a standard Plot Editor layout which is delivered with TopSpin or a user defined layout which has been set up from the Plot Editor.

**autoplot** can take the following arguments:

**-s setup.prt**

Use printer setup file `setup.prt` instead of the printer setup that was saved with the layout (not available in Windows version).

**-l N**

Remove N data sets from the portfolio and print again.

**-n**

Don‘t reset before printing.

**-f**

Force all 1D and/or 2D objects in the layout to use axis limits as used in TopSpin (uses the `F1P/F2P` parameter for each direction).

**-e output.ps**

Create e.g. a Postscript file instead of printer output. Use the `-?` option to see a complete list of supported file formats.

**-v**

Show **autoplot** version number.

**-h**

Show help text.

**-?**

Same as **-h**.

For an extended description of **autoplot** please refer to the Plot Editor online help.

### INPUT PARAMETERS

Set with **edp** or by typing **layout** etc.:

LAYOUT - Plot Editor layout

CURPLOT - Default plotter for Plot Editor

### INPUT FILES

*<tshome>/plot/layouts/*.xwp* - Bruker library Plot Editor layouts

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r* - real processed 1D data

*procs* - processing status parameters

*intrng* - integral regions

*parm.txt* - ascii file containing parameters which appear on the plot

*title* - default title file

*outd* - output device parameters

*portfolio.por* - Plot Editor portfolio (input file is it exists)

For a 2D dataset, the files *2rr*, *proc2s* and *clevels* are also input.

### USAGE IN AU PROGRAMS

AUTOPLOT

AUTOPLOT_WITH_PORTFOLIO

AUTOPLOT_TO_FILE(outputfile)

AUTOPLOT_WITH_PORTFOLIO_TO_FILE(outputfile)

### SEE ALSO

*plot* [▶ 251], *print* [▶ 252], *prnt* [▶ 254]

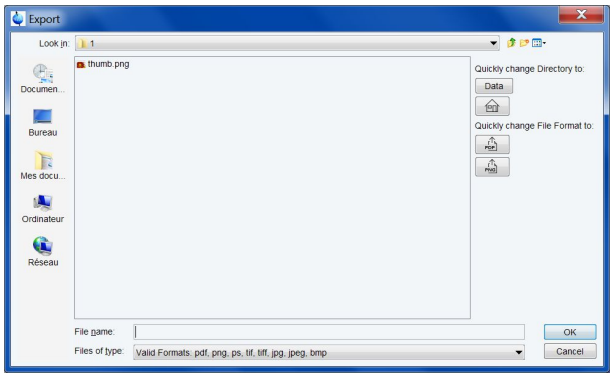
## 8.2 exportfile

### NAME

exportfile - Export data window to graphics file (1D,2D,3D)

### DESCRIPTION

The command **exportfile** saves the contents of a data window in a graphics file of selectable type, e.g. *.png*, *.tif*, *.wmf* etc. It opens an Explorer window.




Here you can:

- Click or enter the name of the output file.
- Click **OK**.

The resolution of such a *screen dump* equals the resolution of your screen. When you import a graphics file into another program, you may lose information when resizing the graphics.

Entering **exportfile** on the command line is equivalent to clicking **File | Export**....

The pathname of the destination graphics file is available in the Windows clipboard.

### OUTPUT FILES

*<outputdir>*

*outputfile*[*.png*, *.jpg*, *.jpeg*, *.bmp*, *.emf*, *.wmf*] - graphics file

### SEE ALSO

*plot* [▷ 251], *autoplot* [▷ 244], *prnt* [▷ 254], *print* [▷ 252]

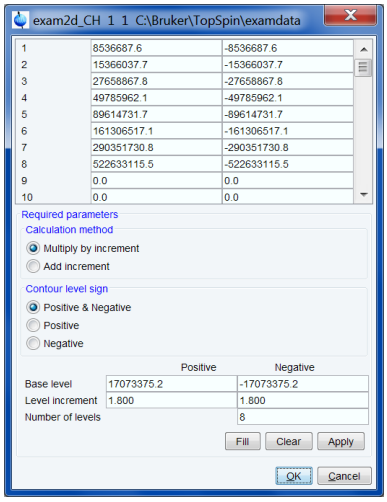
## 8.3 edlev

### NAME

edlev - Edit contour levels (2D,3D)

### DESCRIPTION

The command **edlev** opens a dialog box in which you can set the contour levels of a 2D dataset:




#### Manual setup

This allows you to create an arbitrary sequence of levels

1. Enter the level values in the fields 1, 2, ... at the top of the dialog box.
2. Click **Apply** to update the display or **OK** to store the levels, update the display and close the dialog box**.**

#### Calculation

This allows you to easily create a geometric or equidistant sequence of levels.

1. Click one of the following items:
   - **Multiply with increment**
   - to create a geometric sequence of levels.
   - **Add increment**

– to create a equidistant sequence of levels.

2. Enter the desired *Base level* , *Level increment* and *Number of levels.*
3. Click **Fill** to display and activate the sequence**.**
4. Click **Apply** to update the display or **OK** to store the levels, update the display and close the dialog box**.**

The Contour level sign allows you to select positive or negative levels, or both.

Note that if you change the intensity interactively, for example with the buttons *2, /2 or ↕, the contour levels are automatically adjusted. Entering **edlev** will show the adjusted levels and clicking 💾 will save them to disk.

### INPUT AND OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*clevels* - Contour levels

### SEE ALSO

*ls, rs commandr* [▶ 68], (.ls, .lt)

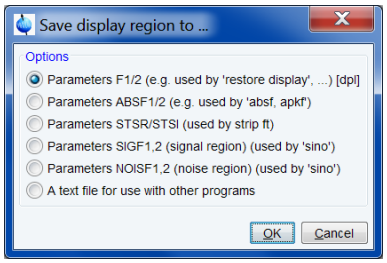
## 8.4 dpl

### NAME

dpl - Save the displayed region (1D, 2D)

### DESCRIPTION

The command **dpl** saves the displayed region in the parameters F1P and F2P. The command can also be executed by right-clicking in the data window and selecting *Save Display Region To...* This will open the dialog box shown:




Here select *Parameters F1/2* and click **OK**.

### OUTPUT PARAMETERS

Can be viewed with **edp** or by typing **f1p** or **f2p** :

F1P - low field (left) limit of the plot region in ppm

F2P - high field (right) limit of the plot region in ppm

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*proc* - plot title

**SEE ALSO**

*plot* [▷ 251], *prnt* [▷ 254], *print* [▷ 252], *autoplot* [▷ 244]

## 8.5 .md, .md no_load, .md write

**NAME**

**.md** - displays spectra in multiple display

**.md no_load** - entering multiple display by ignoring other sessions

**.md write** - writes the assoc file containing the data set list for multiple display

**DESCRIPTION**

The following arguments of **.md** for controlling data sets from command line, AU-programs or Python programs are available:

1. Specified data set names can be shown in the display by command **.md**:
2. Enter command and full pathname for a specified dataset in the TopSpin command line:
3. *.md <PathToDataset1>\<expno1>\pdata\<procno> <PathToDataset2>\ <expno2>\pdata\<procno>*
4. The command **.md no_load** ignores the datasets stored in the last multiple display session and enters the multiple display
5. The command **.md write** writes only the assoc file containing the data set list for multiple display. Please note that the multiple display module is not started with this command. Enter command and full pathname of specified dataset in the TopSpin command line:
6. **.md write** *<PathToDataset1>\<expno1>\pdata\<procno> <PathToDataset2>\ <expno2>\pdata\<procno>*

Multiple display mode is supported for 1D and 2D spectra. For spectra with a dimension > 2 the selected slice (subplane) is shown.

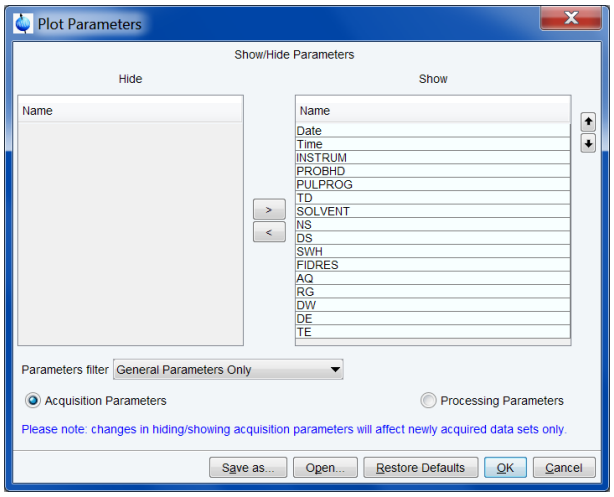
## 8.6 parplot

**NAME**

parplot - select parameters to appear on the plot (1D,2D)

**DESCRIPTION**

The command **parplot** opens a dialog where you can select the acquisition and processing parameters that must appear on the plot:




To select the acquisition parameters to be shown on the plot:

1. Enable the radio button **Acquisition Parameters**. By default, all acquisition parameters are shown and the *Hide* column is empty.
2. In the *Show* column: select the parameters to be hidden.
3. Click the **<** button in the center of the dialog.
4. If desired, you can also select experiment specific (**ased**) parameters by selecting the respective *Parameter filter* and repeating step 2 and 3.

To select the processing parameters to be shown on the plot:

1. Enable the radio button **Processing Parameters**.
2. By default, some processing parameters are shown while most are hidden.
3. In the *Show* column: select the parameters to be hidden.
4. Click the **<** button in the center of the dialog.
5. In the *Hide* column: select the parameters to be shown.
6. Click the **>** button in the center of the dialog.

After selecting the acquisition and/or processing parameters click **OK** to save the selection.

The dialog offers the following buttons:

- **Save as**... : save the current selection under a user defined name
- **Open**... : open a user defined selection
- **Restore Defaults** : restore the TopSpin default selection
- **OK** : save the current selection
- **Cancel** : Close the dialog

The **Save as...** and **Open** button allow to store several selections. Note that these can only be activated from the **parplot** dialog by using the **Open** and **OK** buttons, respectively and then count for all data set.

Only parameters selected with **parplot** will appear on the plot (on datasets created with TopSpin 1.3 or older, first remove the files format.temp in the dataset EXPNO and parm.txt in the dataset PROCNO).

This counts for both interactive plotting (command **plot**) and automated plotting (command **autoplot**).

## INPUT AND OUTPUT FILES

*<tshome>/exp/stan/nmr/form/acqu.l*

*normpl* - acquisition parameters that appear on the plot

*<tshome>/exp/stan/nmr/form/proc.l*

*normpl* - processing parameters that appear on the plot

*<tshome>/exp/stan/nmr/form/*

*<name>* - user defined selection of acquisition/processing parameters

**INPUT AND OUTPUT FILES**

*parplot* [▷ 248]

## 8.7 edti

**NAME**

edti - Set the data set title (1D, 2D, 3D)

**DESCRIPTION**

The command **edti** allows you to define the data set title. Entering this command is equivalent to clicking the Title tab. Changes in the title will automatically appear in the data window after clicking the Spectrum or Fid tab.

The title defined with **edti** will also appear on plots created with **prnt** or **autoplot**.

The command **edti** replaces the formerly used command **setti** which is still available.

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*title* - plot title

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*title* - plot title

**SEE ALSO**

*edtix* [▷ 250], *plot* [▷ 251], *prnt* [▷ 254], *print* [▷ 252], *autoplot* [▷ 244]

## 8.8 edtix

**NAME**

edtix - Set the data set title (1D, 2D, 3D)

**DESCRIPTION**

The command **edtix** allows you to define the data set title with an external editor. It uses the editor that is defined in the User Preferences. To set this editor:

- Click **Preferences | Text Editors | Preferred text editor | Change**

The title will appear in the data window and on plots created with **prnt** or **autoplot**.

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*title* - plot title

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*title* - plot title

### SEE ALSO

*edti* [▶ 250], *plot* [▶ 251], *prnt* [▶ 254], *print* [▶ 252], *autoplot* [▶ 244]

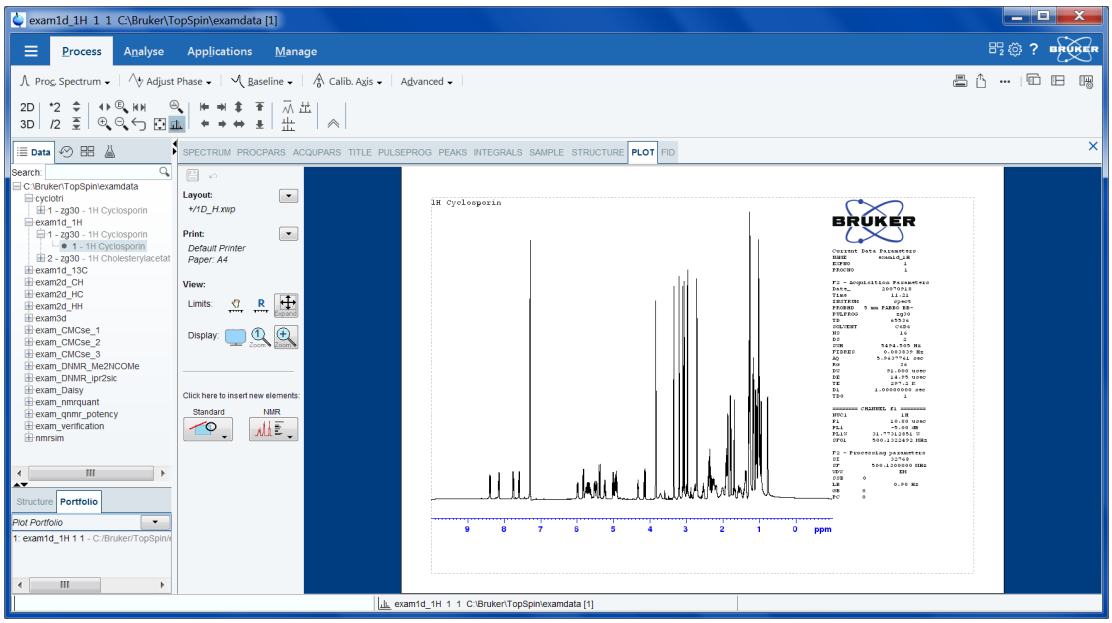
## 8.9 plot

### NAME

plot - Open the Plot Editor (1D, 2D)

### DESCRIPTION

The command **plot** starts the Plot Editor with the current dataset and the layout defined by the processing parameter LAYOUT.




The plot limits of all data objects will be the same as in TopSpin. The command plot can take various arguments and can be used as follows:

The command **plot** can be used with the following arguments:

*(no option)* Force all data objects to use limits from TopSpin

**-r** Apply *Reset Actions* on all objects after loading the layout

**-n** Do not change anything after loading the layout

**-p** myfile.por Load the portfolio file *myfile.por*

**-i** Ignore a *portfolio.por* file found in the data set

The main window of the Plot Editor consists of a drawing area, a menu bar and a toolbar which offers various graphical objects. Here you can display objects like FIDs, one- or two-dimensional NMR spectra, Stacked Plots, parameter lists and titles. You can add integral curves and peak lists to a spectrum, combine several spectra to a stacked plot draw projections around a 2D spectrum.

Furthermore, the Plot Editor offers a set of so-called graphic primitives like lines, text, rectangles and bezier curves. You can place these objects anywhere on the screen and change their appearance. They can be superimposed on NMR-related graphics. All objects can be moved and resized interactively and for each object a range of editing modes is available.

The TopSpin command **autoplot** allows you to plot a spectrum using a Plot Editor layout.

For a full description, please click:

Click **Help | Manuals | Automation and Data Publishing | Data Publishing**

### INPUT PARAMETERS

Set with **edp** or by typing **layout** etc.:

LAYOUT - Plot Editor layout

CURPLOT - Default plotter for Plot Editor

### INPUT AND OUTPUT FILES

*<tshome>/plot/layouts/*.xwp* - Bruker library Plot Editor layouts

*portfolio.por* - Plot Editor portfolio (input file is it exists)

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*layout.xwp* - Plot Editor layout

*last_plot.xwp* - Last stored Plot Editor layout

*portfolio.por* - Plot Editor portfolio

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r* - real processed 1D data

*procs* - processing status parameters

*intrng* - integral regions

*parm.txt* - ascii file containing parameters which appear on the plot

*title* - default title file

*outd* - output device parameters

For a 2D dataset, the files *2rr*, *proc2s* and *clevels* are also input.

### SEE ALSO

*print* [▸ 252], *prnt* [▸ 254], *autoplot* [▸ 244]

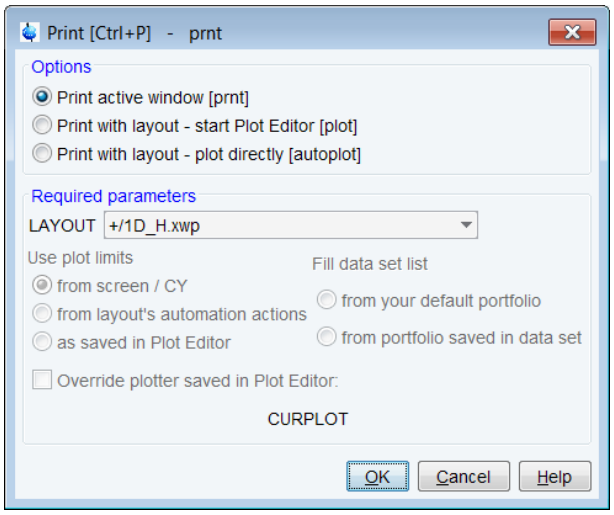
## 8.10 print

### NAME

print - Open print dialog box (1D, 2D, 3D)

### DESCRIPTION

The command **print** opens the following dialog box:




Here, you can choose from three print options:

- *Print active window* [**prnt**]
- The data window is printed as it is displayed on the screen. Before printing starts, the operating system print dialog box will appear where you can, for example, select the printer and printer properties.
- *Print with layout - start Plot Editor* **[ plot ]**
- If you select this option and click OK , the Plot Editor will be started. This option is equivalent to entering **plot** on the TopSpin command line.
- *Print with layout - plot directly* **[ autoplot ]**
- Selecting this option activates the Plot Editor layout list box. Select the desired layout and click OK to print. Standard layouts are delivered with TopSpin. They use the Windows default printer. User defined layouts use the printer defined in the Plot Editor. On a 1D dataset, only 1D layouts are listed, on a 2D dataset only 2D layouts are listed etc.

For the last two options, the following required parameters are available:

**Use plot limits**

- *from screen/ CY* - the plot limits and maximum intensity are used as they are on the screen (processing parameter F1P, F2P and CY, respectively)
- *from Plot Editor Reset Actions* - the plot limits and maximum intensity are set according to the Plot Editor Reset Actions (right-click inside the Plot Editor data field and choose *Automation* to set the Reset Actions).
- *as saved in Plot Editor* - the plot limits and maximum intensity are set in the specified layout

**Fill dataset list**

- *from your default portfolio* - the portfolio contains the current TopSpin dataset plus the data from the default Plot Editor portfolio
- *from port folio saved in dataset* - the portfolio contains the current TopSpin dataset plus the data from the portfolio stored in this dataset

**Override Plotter saved in Plot Editor**

If enabled, the plotter defined in the Plot Editor layout will be overridden by the plotter defined by the processing parameter CURPLOT.

For each Option/Required Parameter combination, the corresponding command line command is shown in the title bar of the dialog box. In the example above this is the command **plot -f**.

### INPUT FILES

See the description of **prnt**, **plot** and **autoplot**

### SEE ALSO

*prnt* [ 254], *plot* [ 251], *autoplot* [ 244]

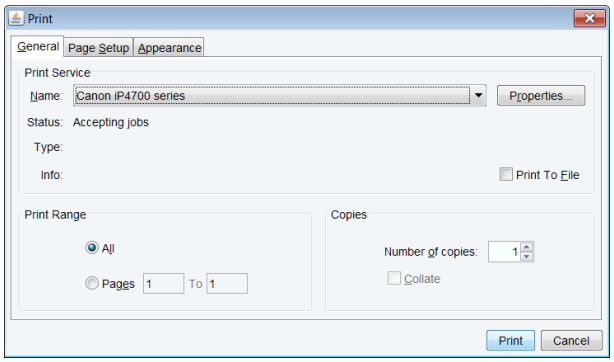
## 8.11 prnt

### NAME

prnt - Print the current dataset (1D, 2D, 3D)

### DESCRIPTION

The command **prnt** prints the current dataset as it is shown on the screen. Before printing starts, the operating system print dialog box will appear. Here you can, for example, select the printer and printer properties.




### SEE ALSO

*print* [ 252], *plot* [ 251], *autoplot* [ 244]

## 8.12 savelogs

### NAME

savelogs - Save logfiles

### DESCRIPTION

savelogs is mainly used for debugging purposes. This tool will collect support information about the current TopSpin installation (log and configuration files, by default no NMR data) and allows to transfer it to Bruker. It offers a *Comments* field to enter a description of your issue.

**Note**: If already in contact with Bruker, give a reference to a mail or phone call or ticket number.

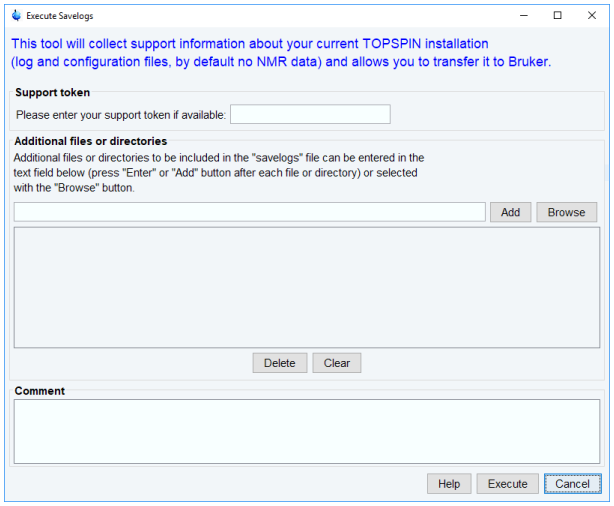
If issues with spectra are observed, please add the respective NMR data with the *Additional files or directories* option.

The file transfer process has been changed from ftp to a https secured transfer method.

This tool is also available in the menu bar:

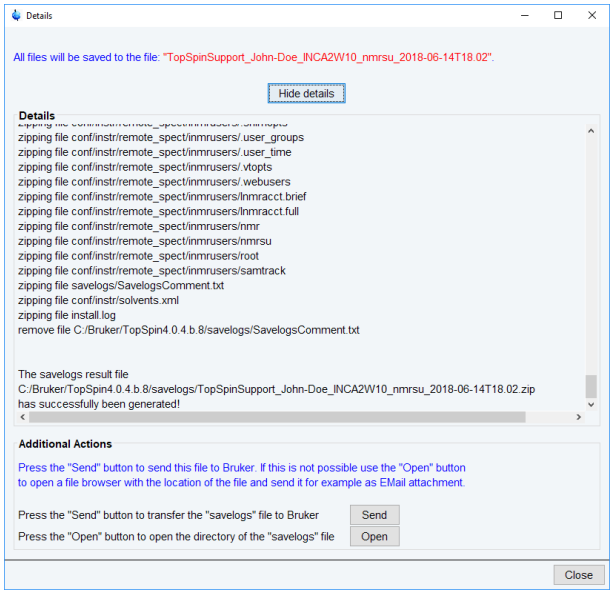
- Click **Manage | Commands | Collect & Save LogFiles**

The Execute Savelogs window is displayed:



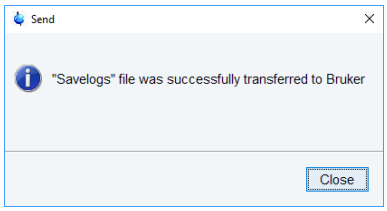

The recommended token will be provided by the Bruker support. If not available, enter your name and the name of your institution or company.

Once the **savelogs** command has created the savelogs file, the window changes and offers a direct upload of the file to Bruker.




- Click **Send** to transfer the file and notify your Bruker Support team member once the upload has been completed.
- Click **Open** to see the resulting savelogs file for other transfer options.

When the transfer has finished a message window is displayed. Click **Close**.




If TopSpin cannot be started:

- Under Windows:
  - Click the **Bruker Utilities<topspin version>** icon on your desktop. An Explorer will be opened.
  - Double-click **Miscellaneous** .
  - Execute the script **savelogs .**
- Under Linux:
  - Open a shell.
  - Enter **savelogs** .
- Under macOS:
  - Open **Applications - <topspin version> Utilities** .
  - Execute **savelogs** .

## INPUT FILES

User-specific installation files like history files etc. named:

*<tshome>/prog/curdir/<user>/**

## OUTPUT FILES

*<TS home>\savelogs\TopSpinSupport_<Token><user><YY><MM><DD><HH><MM>.zip*

## SEE ALSO

*hist* [▶ 365]

# 9 Dataset Handling

This chapter describes all TopSpin commands which can be used to read or write or delete datasets.

## 9.1 copy

### NAME

Copy - Copy the contents of the current data window to the Clipboard (nD)

### DESCRIPTION

Under Windows, the command **copy** copies the contents of the current data window to the clipboard. The data are copied as a bitmap (in TopSpin 2.0 and older, data were copied in WMF format). To copy the data as a windows metafile, use the command **copy wmf**.

On Linux is the screen dump (**png** format) copied to a temporary file, the pathname of this file is copied to clipboard.

Entering **copy** on the command line is equivalent to clicking **File | Copy** in the menu.

### SEE ALSO

*paste [▶ 278]*

## 9.2 dalias

### NAME

dalias - Create an alias name for a dataset (nD)

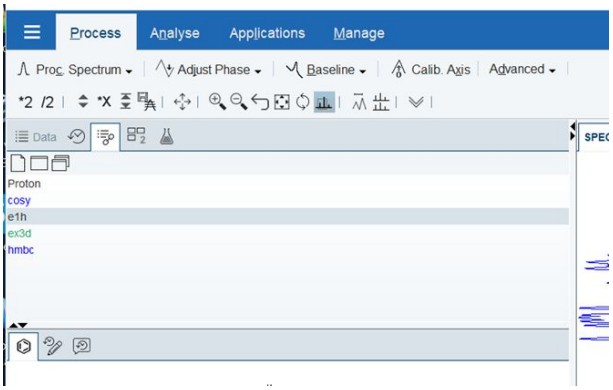
### DESCRIPTION

The Alias Tab in Topspin allows to work with short dataset names (aliases) for frequently used data. Each alias is linked to one dataset with fixed EXPNO and PROCNO.

Please note, that the Alias Tab can be switched on or off in Set/Browser Settings.

The aliases may be managed through the GUI - a popup menu on the Alias Tab. An alternative option is the **dalias** command.

The aliases given for single datasets are different handled from the aliases in the Data Tab. They are only a placeholder for complete directories.



Entering the command **dalias** without arguments displays a help message with a summary of all options:


The command requires various arguments and can be used as follows:

**Create alias name for a dataset**

**dalias add <alias> <name> <eno> <pno> <dir>**

or

**dalias add <alias> <pathname>**

Note: the alias name must start with a letter!

Create the alias name <alias> for the specified dataset, e.g.:

**dalias add e1h exam1d_1H 1 1 C:/bio**

or

**dalias add e1h C:/bio/data/guest/nmr/exam1d_1H/1/pdata/1**

**Show dataset path or name behind alisases**

**dalias pr <alias>**

Print the name, expno, procno and dir of the specified alias name.

**dalias prgen <alias>**

Print the full pathname of the specified alias name.

**dalias prall**

Print the name, expno, procno, dir of all alias names.

**dalias prallgen**

Print the full data path of all alias names.

**Remove alias names**

**dalias rm <alias>**

Remove the specified alias name.

**dalias rmall**

Remove all alias names.

Note: Removing aliases does not remove corresponding data from disk.

**SEE ALSO**

*re, rep commandr [▷ 279]*

## 9.3 del, dela, delp, deldat, delete

**NAME**

del - Delete data (nD)

dela - Delete raw data (nD)

delp - Delete processed data (nD)

deldat - Delete data acquired at certain dates (nD)

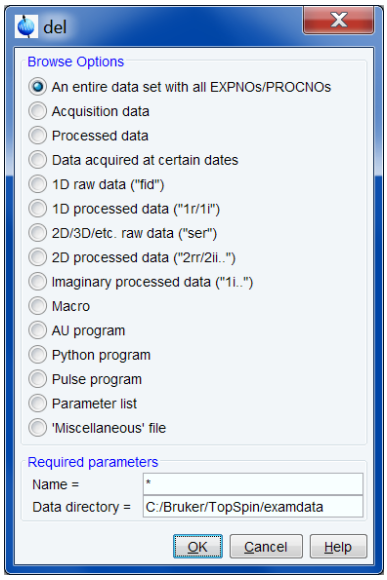
delete - Open the delete dialog box (nD)

**SYNTAX**

del* [<name>]

**DESCRIPTION**

Delete commands can be started from the command line or from the delete dialog box. The latter is opened with the command **delete**:

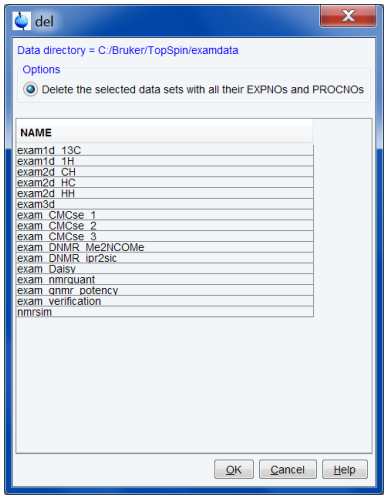



This dialog box has several options, each of which selects a certain command for execution.

The commands **del**, **dela**, **delp** and **deldat** allow you to display a list of datasets. Such a list includes datasets containing raw and/or processed data as well as empty datasets which only contain parameter files. You can select one or more datasets in the list to mark them for deletion and then click **OK** to actually delete them.

### An entire dataset with all expnos/procnos

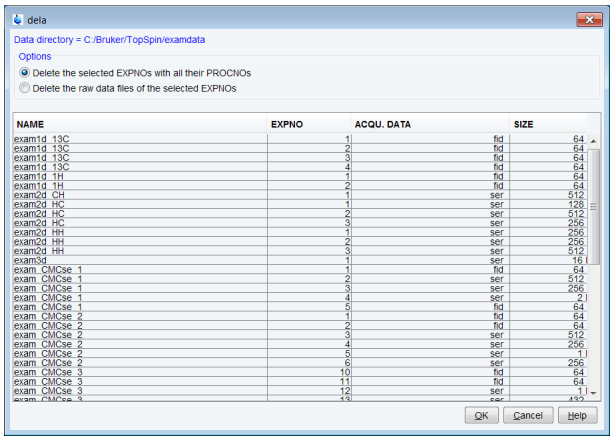
This option selects the command **del** for execution. It lists datasets, only showing the dataset name. To delete data, select one or more datasets and click **OK**. The marked datasets are entirely deleted, including data files, parameter files and the data name directory.




### Acquisition data

This option selects the command **dela** for execution. It. It lists datasets showing a separate entry for each experiment number (*expno*). Each entry shows the dataset NAME, EXPNO, ACQU.DATA and SIZE. Datasets which do not contain raw data are displayed with ACQU.DATA *none*. To delete data, select one or more datasets and check one of the following check boxes:

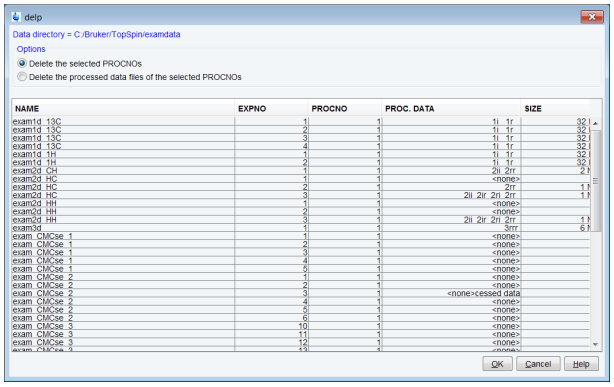
- **Delete the selected EXPNOs with all their PROCNOs** to delete the *expno* directory.
- **Delete the raw data files of the selected EXPNOs**.




### Processed data

This option selects the command **delp** for execution. It lists datasets showing a separate entry for each processed data number (*procno*). Each entry shows the dataset NAME, EXPNO, PROCNO, PROC.DATA and SIZE. Datasets which do not contain processed data are displayed with PROC.DATA *none*. To delete data, select one or more data sets and check one of the following check boxes:

- **Delete the selected PROCNOs** to delete the *procno* directories.
- **Delete the processed data files of the selected PROCNOs**.




### Data acquired at certain dates

This option selects the command **deldat** for execution and lists all data sets chronologically.

When started from the command line, **del*** commands can take one argument which may contain wild cards. Examples:

**dela exam1d*** - List all data sets whose name starts with *exam1d*

**dela exam1d???** - List all data sets whose name is *exam1d* plus three extra characters

**del*** commands only list and delete the datasets of current user. The current user here refers to the *user* part of the data path of the currently selected dataset.

Please distinguish:

- The user part of the data path.
- The owner of the data set.
- The user who runs TopSpin.

Usually these three things are the same, i.e. a user works on his own data. However, the user part of the data path can be any character string and does not have to correspond to a user account on the computer. Furthermore, the user who runs TopSpin might work on someone else's data. In this case, he/she may or may not have the permission to delete this dataset. In the latter case, the **del*** commands will not delete the dataset but show an error message instead.

**OUTPUT FILES**

**For dela: Delete raw data files of the selected EXPNOs:**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*audita.txt* - acquisition audit trail

**For delp: Delete processed data files of the selected PROCNOs:**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*auditp.txt* - processing audit trail

**SEE ALSO**

*delf, dels [▸ 262]*

## 9.4 delf, dels, delser, del2d, deli

**NAME**

delf - Delete raw data (1D)

dels - Delete processed data (1D)

delser - Delete raw data (2D,3D)

del2d - Delete processed data (2D,3D)

deli - Delete imaginary processed (nD)

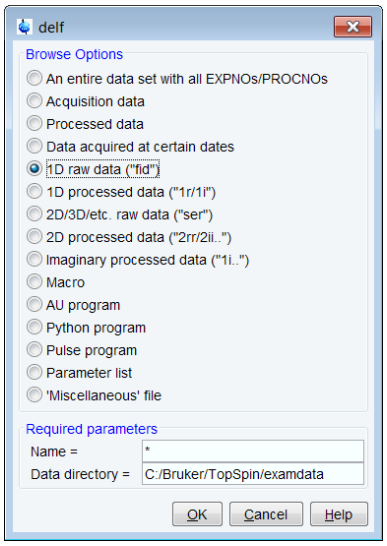
delete - Open delete dialog box (nD)

**SYNTAX**

del* [<name>]

**DESCRIPTION**

Delete commands can be started from the command line or from the delete dialog box. The latter is opened with the command **delete**:



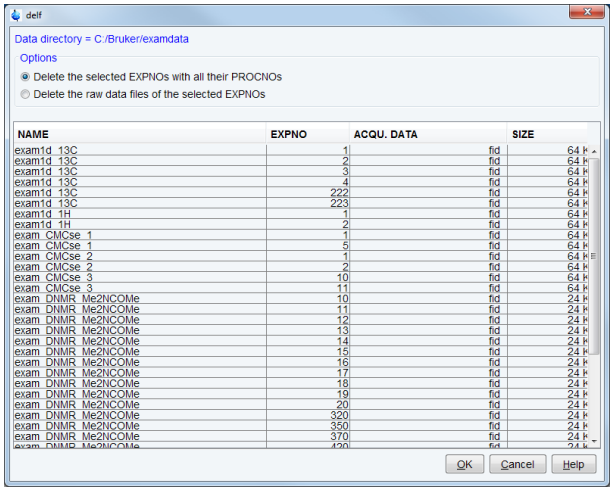

This dialog box has several options, each of which selects a certain command for execution.

The commands **delf**, **dels**, **delser**, **del2d** and **deli** display a list of data sets. Such a list only includes data sets which contain data files. As opposed to commands like **del** and **dela**, they do not show empty data sets. You can select one or more data sets to mark them for deletion and then click **OK** to actually delete them.

### 1D raw data

This option selects the command **delf** for execution. It lists 1D datasets which contain raw data showing a separate entry for each experiment number (*expno*). Each entry shows the dataset NAME, EXPNO, ACQU.DATA and SIZE. To delete data, select one or more data sets and check one of the following check boxes:

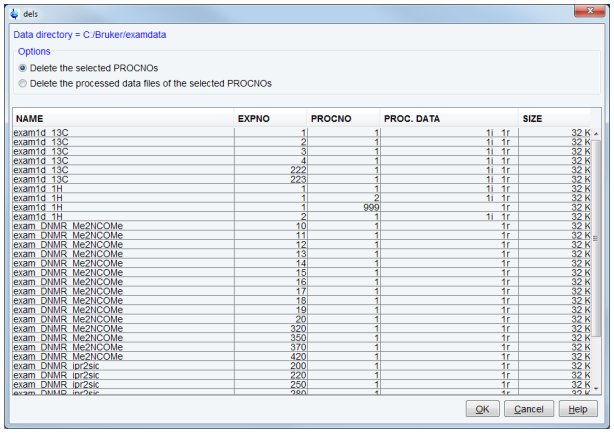
- **Delete selected EXPNOs** to delete the *expno* directory.
- **Delete raw data files** *of the selected EXPNOs*.



### 1D processed data

This option selects the command **dels** for execution. It lists 1D datasets which contain processed data showing a separate entry for each processed data number (*procno*). Each entry contains the data set NAME, EXPNO, PROCNO, PROC.DATA and SIZE. To delete data, select one or more data sets and check one of the following check boxes:

- **Delete the selected PROCNOs** to delete the *procno* directories.
- **Delete processed data files of the selected PROCNOs**.




### 2D/3D raw data

This option selects the command **delser** for execution. It lists 2D and 3D data sets which contain raw data showing a separate entry for each experiment number (*expno*). Each entry shows the data set NAME, EXPNO, ACQU.DATA and SIZE. To delete data, select one or more data sets and check one of the following check boxes:

- **Delete selected EXPNOs** to delete the *expno* directory.
- **Delete raw data files of the selected EXPNOs**.

### 2D processed data

This option selects the command **del2d** for execution. It lists 2D data sets which contain processed data showing a separate entry for each processed data number (*procno*). Each entry shows the data set NAME, EXPNO, PROCNO, PROC.DATA and SIZE. To delete data, select one or more data sets and check one of the following check boxes:

- **Delete selected PROCNOs** to delete the *procno* directories.
- **Delete processed data files of the selected PROCNOs**.

### Imaginary processed data

This option selects the command **deli** for execution. It lists data sets which contain 1D, 2D or 3D imaginary data showing a separate entry for each processed data number (*procno*). Each entry shows the dataset NAME, EXPNO, PROCNO, PROC.DATA and SIZE. Only the imaginary processed data files are deleted. Raw data, processed data and parameter files are kept. To delete data, mark one or more data sets and check:

**Delete imaginary processed data of the selected PROCNOs.**

When started from the command line, **del*** commands can take one argument which may contain wild cards. Examples:

**delf exam1d***

List all data sets whose name starts with *exam1d*

**delf exam1d???**

List all data sets whose name is *exam1d* plus three extra characters

**del*** commands only list and delete the data sets of current user. The current user here refers to the *user* part of the data path of the currently selected data set. Please distinguish:

- The user part of the data path.
- The owner of the dataset.
- The user who runs TopSpin.

Usually these three things are the same, i.e. a user works on his own data. However, the user part of the data path can be any character string and does not have to correspond to a user account on the computer. Furthermore, the user who runs TopSpin might work on someone else's data. In this case, he/she may or may not have the permission to delete this data set. In the latter case, the **del*** commands will not delete the data set but show an error message instead.

### OUTPUT FILES

**For delf/delser: Delete raw data files of the selected EXPNOs:**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*audita.txt* - acquisition audit trail

**For dels/del2d/deli: Delete processed data files of the selected PROCNOs:**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*auditp.txt* - processing audit trail

### SEE ALSO

*del, dela [▶ 259]*

## 9.5 dir, dira, dirp, dirdat, browse

### NAME

dir - List datasets (nD)

dira - List raw data (nD)

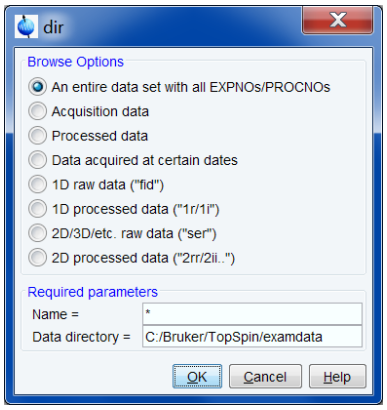
dirp - List processed data (nD)

dirdat - List data acquired at certain dates (nD)

browse - Open data list dialog box (nD)

### DESCRIPTION

Commands to list data directories can be started from the command line or from the directory dialog box. The latter is opened with the command **browse**:



This dialog box has several options, each of which selects a certain command for execution.

The commands **dir**, **dira**, **dirp** and **dirdat** display all data sets containing raw and/or processed data as well as empty data sets which only contain parameter files. You can mark one or more entries in the list and click:

**Display selected data** - to display the data in the current data window.

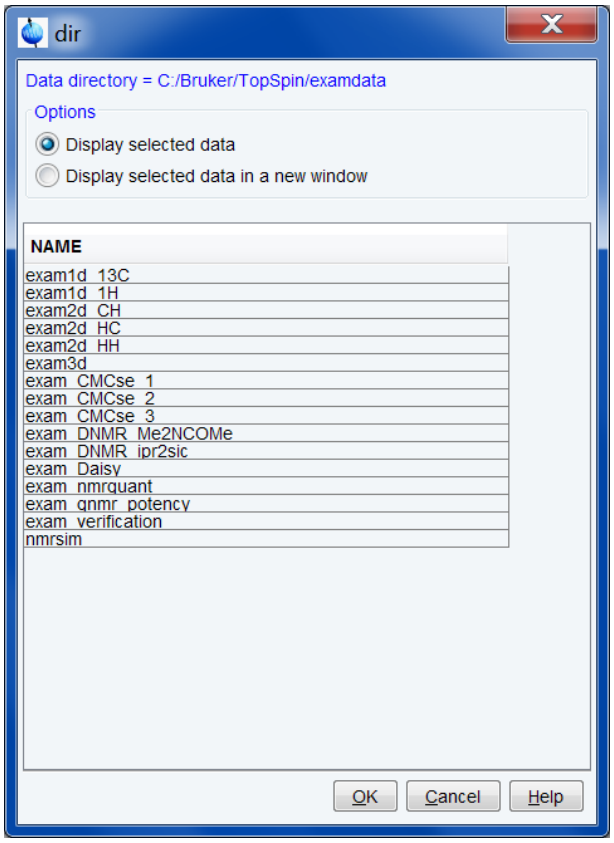
or

**Display selected data in a new window** - **t**o display the data in a new data window.

When multiple entries were marked, they will be shown in one data window in multi-display mode.

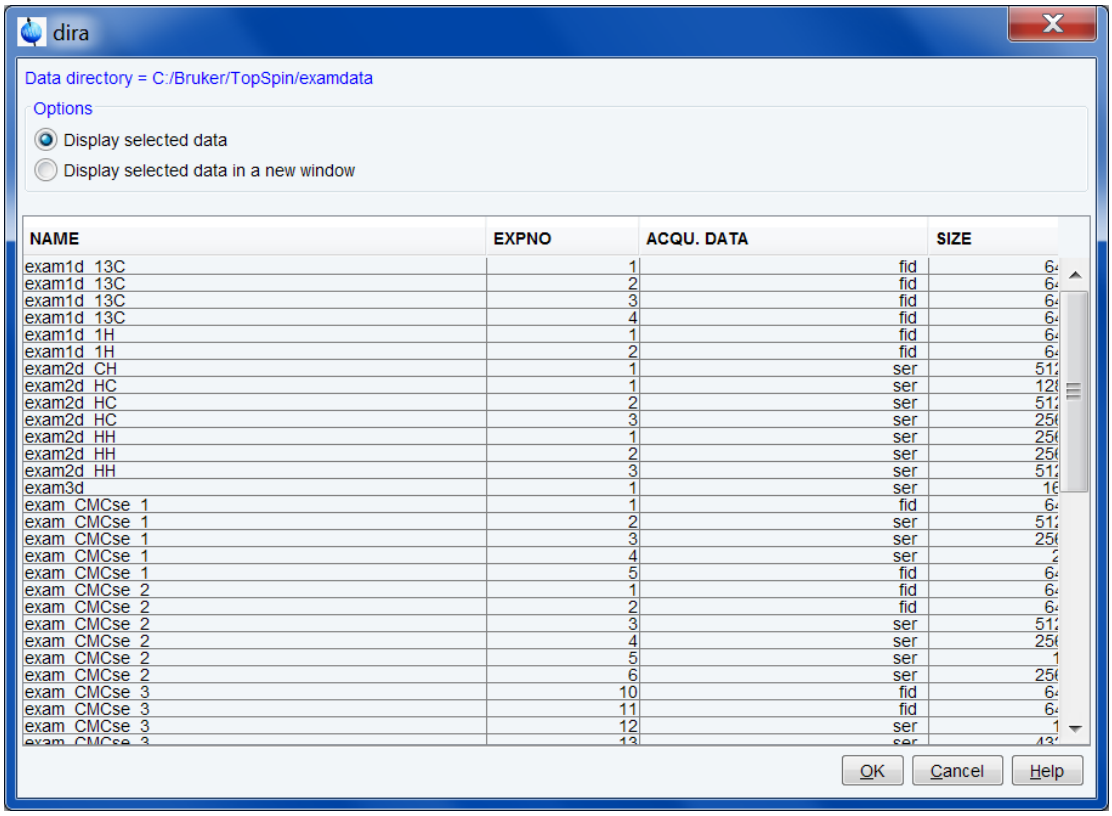
### An entire data set with all EXPNOs/PROCNOs

This option selects the command **dir** for execution. It lists data sets, showing the data names only.



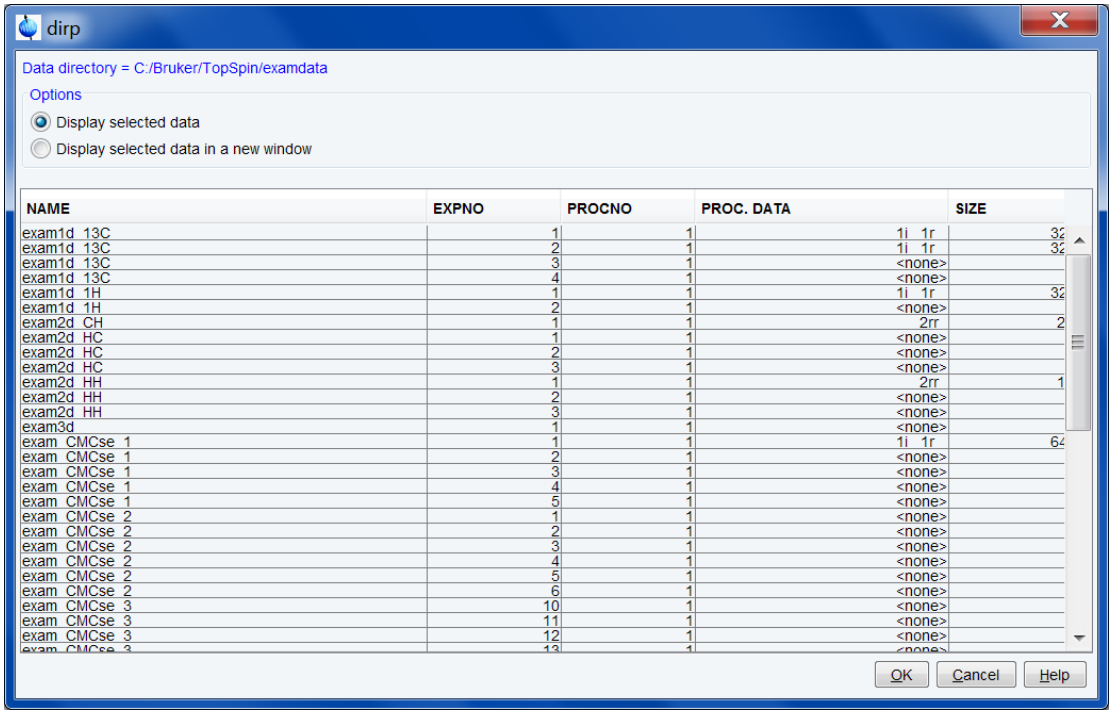

### Acquisition data

This option selects the command **dira** for execution. It lists data sets showing a separate entry for each *expno*. Each entry shows the data set NAME, EXPNO, ACQU.DATA and SIZE. The entry *file* refers to the data files and can be *fid* (1D raw data), *ser* (2D or 3D raw data) or *no raw data*.



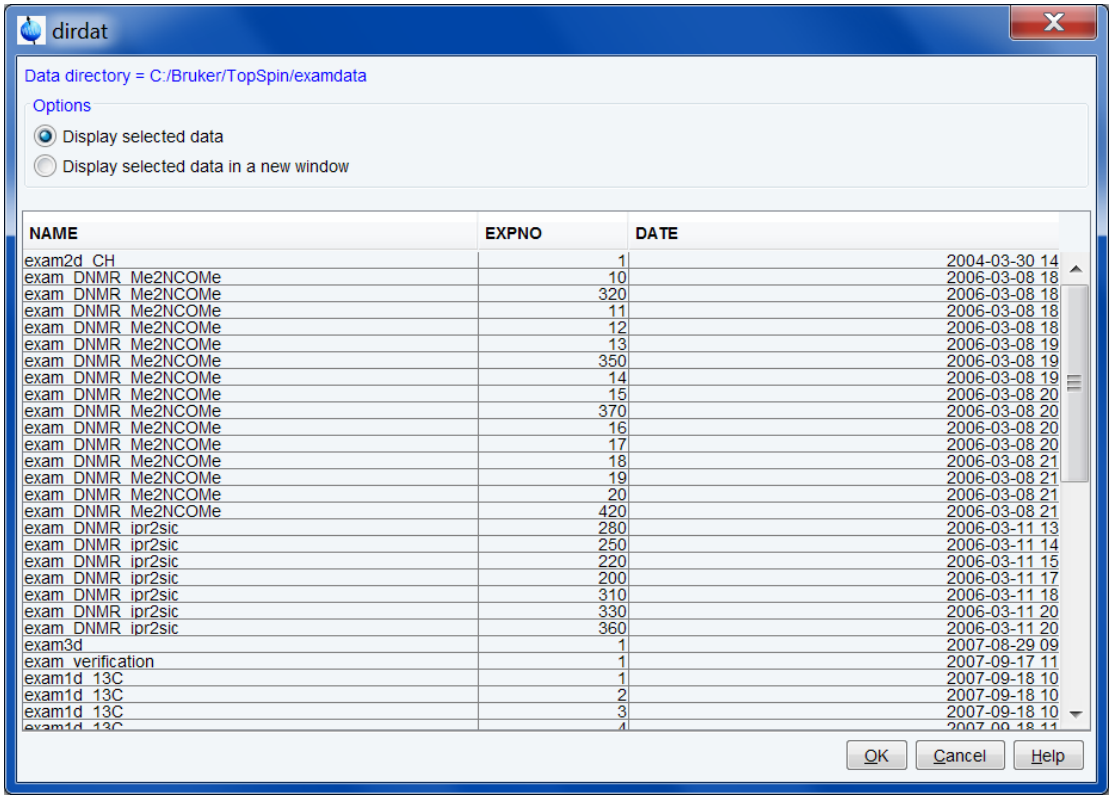

### Processed data

This option selects the command **dirp** for execution. It lists data sets showing a separate entry for each processed data number (*procno*). Each entry shows the data set NAME, EXPNO, PROCNO, PROC.DATA and SIZE. The type refers to the name of the data files and can be *1r 1i* (processed 1D data), *2rr 2ir 2ri 2ii* (2D raw data), *3rrr, 3rri, ..* (processed 3D data) or *no processed data*.



### Data acquired at certain dates

This option selects the command **dirdat** for execution and lists all data sets chronologically.




When started from the command line, **dir*** commands can take one argument which may contain wild cards. Examples:

**dir exam1d***

List all data sets whose name starts with *exam1d*.

**dir exam1d???**

List all data sets whose name is *exam1d* plus three extra characters.

## INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - 1D raw data

*ser* - 2D or 3D raw data

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data

*2rr*, *2ir*, *2ri*, *2ii* - processed 2D data

*3rrr*, *3irr*, *3rir*, *3iir* - processed 3D data

## SEE ALSO

*dirf, dirs, dirser, dir2d, browse* [▷ 269], *find, search* [▷ 271], *open* [▷ 277], *re, rep, rew, repw* [▷ 279], *reb* [▷ 281]

## 9.6 dirf, dirs, dirser, dir2d, browse

### NAME

dirf - List raw data (1D)

dirs - List processed data (1D)

dirser - List raw data (2D,3D)

dir2d - List processed data (2D,3D)

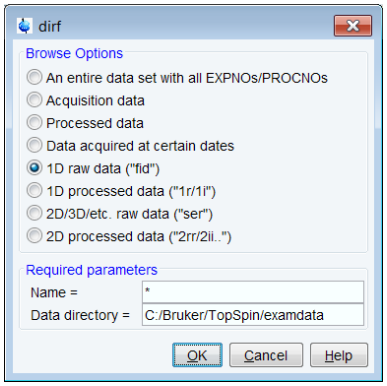
browse - Open the list data dialog box (nD)

### SYNTAX

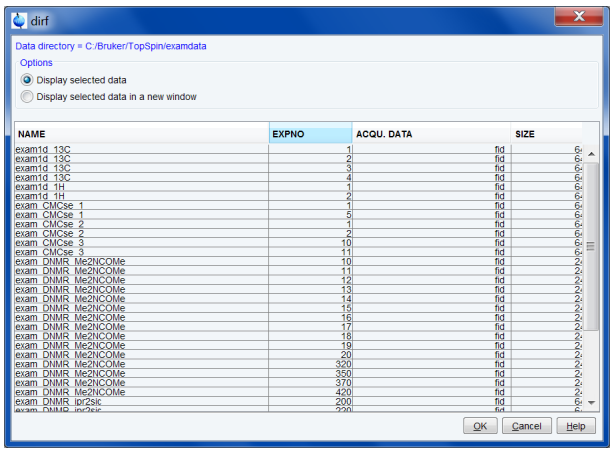
dir* [<name>]

### DESCRIPTION

The **dir*** commands display a list of data sets according to certain criteria. They can be started from the command line or from the **browse** dialog box:




- Click **OK** to display the raw data.




The commands **dirf**, **dirs**, **dirser** and **dir2d** display a list of data sets. This list only includes data sets which contain certain data files. As opposed to commands like **dir** and **dira**, they do not show empty data sets. You can mark one or more datasets in the list and click:

**Display**

To display the data in the current data window.

or

**Display in new window**

To display the data in a new data window.

When multiple entries were marked, the will be shown in one data window in multi-display mode.

**1D raw data**

This option selects the command **dirf** for execution. It lists 1D data sets which contain raw data showing a separate entry for each experiment number (*expno*). Each entry shows the data set NAME, EXPNO, ACQU.DATA and SIZE.

**1D processed data**

This option selects the command **dirs** for execution. It lists 1D data sets which contain processed data showing a separate entry for each processed data number (*procno*). Each entry shows the data set NAME, EXPNO, PROCNO, PROC.DATA and SIZE.

**2D/3D raw data**

This option selects the command **dirser** for execution. It lists 2D and 3D data sets which contain raw data showing a separate entry for each experiment number (*expno*). Each entry shows the data set NAME, EXPNO, ACQU.DATA and SIZE.

**2D processed data**

This option selects the command **dir2d** for execution. It lists 2D data sets which contain processed data showing a separate entry for each processed data number (*procno*). Each entry shows the data set NAME, EXPNO, PROCNO, PROC.DATA and SIZE.

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - 1D raw data

*ser* - 2D or 3D raw data

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data

*2rr*, *2ir*, *2ri*, *2ii* - processed 2D data

*3rrr*, *3irr*, *3rir*, *3iir* - processed 3D data

**SEE ALSO**

*dir, dira, dirp, dirdat, browse* [▶ 265], *find, search* [▶ 271], *re, rep, rew, repw* [▶ 279], *reb* [▶ 281]

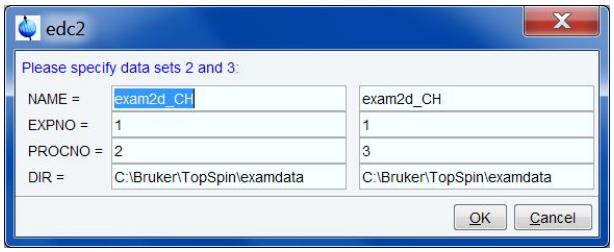
## 9.7 edc2

**NAME**

edc2 - Define second and third data set.

**DESCRIPTION**

The command **edc2** opens a dialog box in which you can define the second and third data set:




You can define the NAME, EXPNO, PROCNO and DIR (disk unit). Note that these are all parts of the data path name:

*<dir>\<name>\<expno>\pdata\<procno>*

The second data set is used by 1D commands like **add**, **duadd**, **mul**, **div** and **addfid** and by 2D commands like **add2d**, **mul2d** and **addser**. The second data set is, however, usually set from the add/multiply dialog box (command **adsu**).

The third data set is used by the 1D command **add** when entered from the command line and in various AU programs (macro DATASET3).

### INPUT AND OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*curdat2* - definition of the second data set

### SEE ALSO

*mul, mulc, nm, div [▶ 69]*, *add2d, mul2d, addser [▶ 99]*, *add, duadd, addfid, addc, adsu [▶ 44]*

## 9.8 find, search

### NAME

Find - Find data according to specified criteria (nD).

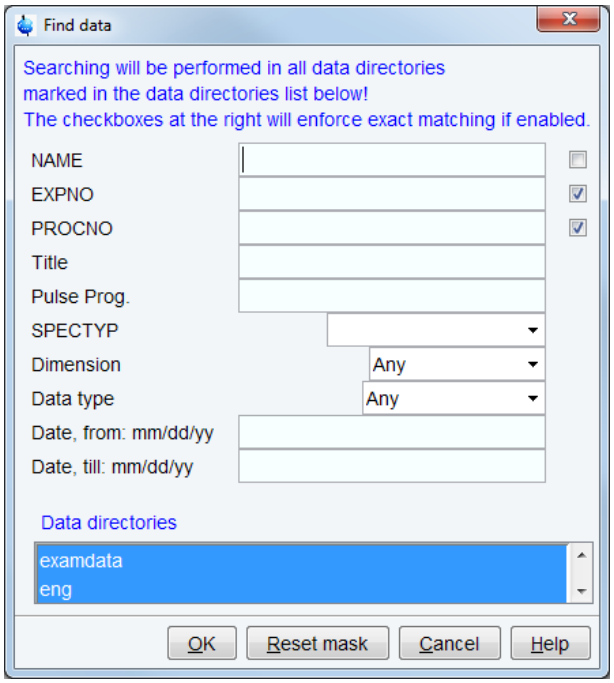
### DESCRIPTION

The command **find** allows to find TopSpin data according to various criteria.

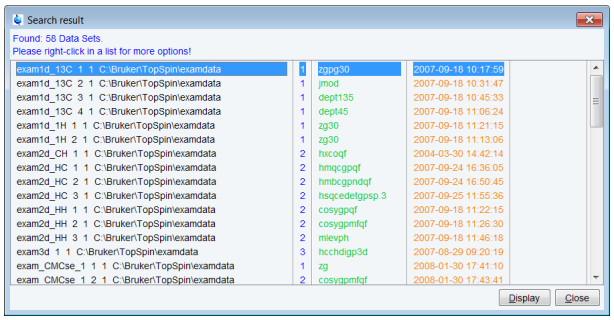
- To open the Find data window (see figure below)
  - In the Browser and search window click **Find**

  - or in the command line enter **Find**
  - or enter **Ctrl+f find** ].
- Enter the search items in the upper part of the dialog. Note that:
  - It will be searched for items containing the specified string.
  - Exact matching is performed for data set variables, NAME, EXPNO, PROCNO and USER, if the checkboxes at the right are enabled.
  - The search is restricted to data created between the specified dates. Note that this refers to the acquisition date.
  - The **Reset mask** button resets the default criteria.
- Select the **Data directories** to be searched in the lower part of the dialog. If no directories are selected, all will be searched.




- Click **OK t**o start the search and display the result.



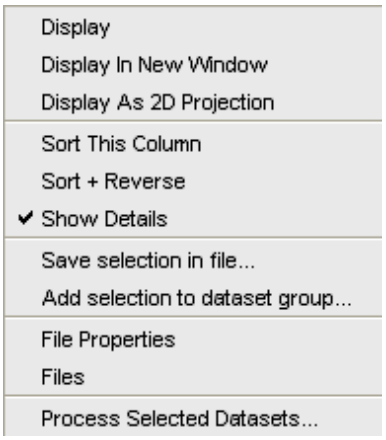

**Note**: when exiting TopSpin, the search criteria will be saved as default.

### How to Display one of the Found Data Sets

In the  search result window:

1. Click one or more data sets to select them.
2. Click **Display** to display the selected data set(s) in the current data window. If multiple data sets are selected they are displayed in the new data window in multiple display mode.

The search result window offers a right-click context menu with various options:




**Display**

Display the selected data set(s) in the current data window. If multiple data sets are selected they are displayed in the same data window in multiple display mode. Equivalent to clicking the **Display** button or pressing **Enter**.

**Display in New Window**

Display the selected data set(s) in a new window. If multiple data sets are selected they are displayed in the one new data window in multiple display mode.

**Display as 2D Projection**

Display the selected data set as a projection of the current 2D data set. A dialog will appear allowing you to choose F1-projection, F2-projection or both. If multiple data sets are selected, only the first one is considered. If the current data set is not a 2D data set, nothing happens.

**Sort This Column**

Sort the selected column in ascending order.

**Sort + Reverse**

Sort the selected column in descending order.

**Show Details**

Show/hide the data set details Dimension, Pulse program and Acquisition date.

**Save Selection to File..**

Save the list of selected data sets in a text file. First opens a file dialog where you can select or specify a file name. The saved data set list can, for example, be used for serial processing (command **serial**, see also **Process Selected Data sets** below).

**Add Selection to data set group..**

Add the list of selected data sets to a data set group. You will be prompted to enter the group name. The created or modified group can be accessed from the browser.

**File properties**

Show main data set parameters like *Dimension*, *Pulse program*, *Acquisition Date*, *Nuclei*, *Spectrometer frequency* and *Solvent*.

**Files**

Show the files in the processed data directory of the selected data set.

**Process Selected Data sets**

Perform serial processing on the selected data sets. Opens a dialog where you can change or edit the data set list and specify the command, macro or Jython program to be executed (starts the command **serial**).

The **Close** button allows you to close the search result dialog.

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - 1D raw data

*acqu* - acquisition parameters

*acqus* - acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data

*proc* - processing parameters

*procs* - processing status parameters

Note that these are only the main 1D data files.

**SEE ALSO**

*dir, dira, dirp, dirdat, browse* [▷ 265], *new* [▷ 275], *open* [▷ 277], *re, rep, rew, repw* [▷ 279], *reb* [▷ 281]

## 9.9 lockdataset

The command **lockdataset** applies permission changes on the current data set. Content of the EXPNO and PROCNO directories will be protected against further overwrite/append/delete operations, and the directory objects itself will lose permissions to add file and subdirectories in it. Effectively, the directory will be frozen. It is still possible to add and process new PROCNOs for the same raw data while the initial PROCNO remains protected. This is especially useful in GLP environments and allows to implement a standard procedure like e.g. the following:

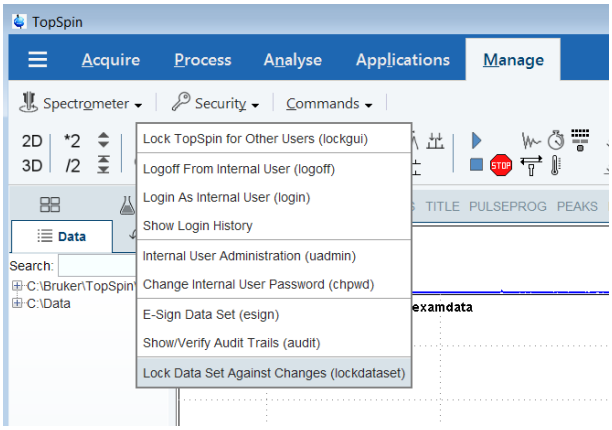
automatically acquire and process data set in PROCNO 1→ digitally sign data by command **esign**

→ apply **lockdataset** to protect against modification

→ use command **wrp 2** to create new PROCNO → change to it by **rep 2**

→ perform interactive processing there (without touching original signed data)

The command **lockdataset** can be used as part of AU scripts like e.g. the one defined by AUNMP. It is also available by interactive menu selection *Manage/Security/Lock Data Set Against Changes*



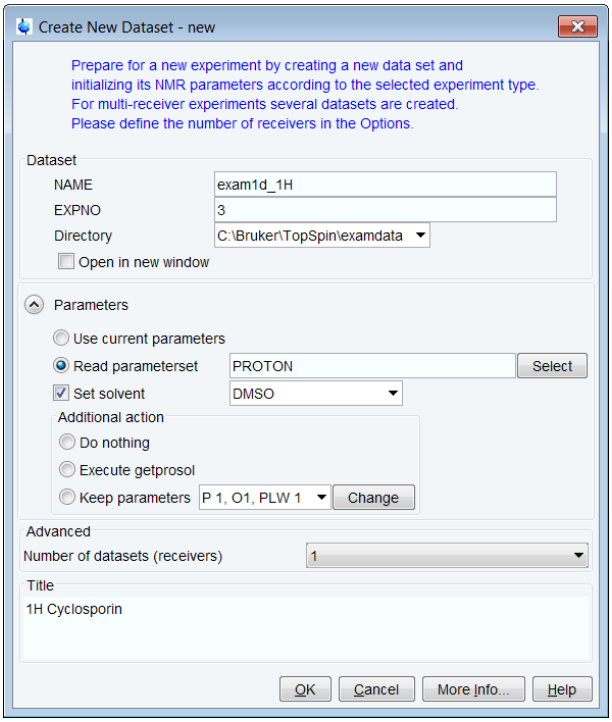

## 9.10 new

### NAME

new - Define a new dataset (nD)

### DESCRIPTION

The command **new** [**Ctrl-n**] opens a dialog box in which you can define a new data set.




**Dataset:**

Here, you can specify the data set NAME, EXPNO and Directory (disk unit). Note that these are all parts of the data path name:

*<dir>\<name>\<expno>\pdata\<procno>*

**Parameters:**

- **Use current parameters** – creates the new dataset with the parameters of the current dataset.
- **Read parameterset** - copies the acquisition and processing parameters from the selected experiment.
- **Set Solvent** - sets the acquisition parameter SOLVENT. Default is the solvent of the current data set.

**Additional action:**

- **Do nothing** – no addional actions are performed.
- **Execute getprosol** – reads the probe and solvent specific parameters.
- **Keep parameters** – keeps the listed parameters from the current dataset.

**Advanced:**

- **Number of datasets (receivers)** – defines the number of datasets for multi-receive experiments.

**Title:**

- Enter a description for the new dataset.

The command **new** remembers the last selected options. When you click **OK**, the data set is created and displayed as the current data window. If the specified data set already exists, you will be prompted to overwrite it or not. Note that this will only overwrite the parameters, not the data files.

**new** is equivalent to the command **edc**.

**INPUT FILES**

*<tshome>/prog/curdir/<user>/*

*curdat* - current data set definition

If *Experiment = Use current params*:

*<dir>/data/<user>/nmr/<name>/<expno>/*

*acqu* - acquisition parameters

*acqus* - acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*proc* - processing parameters

*procs* - processing status parameters

If *Experiment ≠ Use current params.*:

*<tshome>/exp/stan/nmr/par/<experiment>/*

*acqu* - acquisition parameters

*proc* - processing parameters

**OUTPUT FILES**

*<tshome>/prog/curdir/<user>/*

*curdat* - current data set definition

If the data set specified with **new** does not exist yet, the current data set is copied:

*<dir>*/data/*<user>*/nmr/*<name>*/*<expno>*/

*acqu* - acquisition parameters

*acqus* - acquisition status parameters

<*dir*>/data/<*user*>/nmr/<*name*>/<*expno*>/pdata/<*procno*>/

*proc* - processing parameters

*procs* - processing status parameters

For 2D and 3D data the files *acqu2*, *acqu2s* etc. are also output.

**SEE ALSO**

*dir, dira, dirp, dirdat, browse* [▷ 265], *find, search* [▷ 271], *open* [▷ 277], *re, rep, rew, repw* [▷ 279]

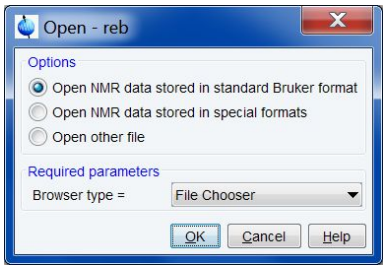
## 9.11 open

**NAME**

open - Open a dataset, pulse program, AU program etc. (nD)

**DESCRIPTION**

Opening data, parameters, lists and various other files can be started from the command line or from the open dialog box. The latter is opened with the command **open** [**Ctrl-o**]:




This dialog box has three options each with several file types. Each file type selects a certain command for execution.

**Open NMR data stored in standard Bruker format**

This option allows you to open Bruker format data in the following ways:

- File chooser [**reb**]
- RE dialog [**re**]
- PROCNO dialog [**rep**]

**Open NMR data stored in special formats**

This option allows you to open the following NMR data types (formats):

- JCAMP-DX [**fromjdx**]
- Zipped TopSpin [**fromzip**]
- WIN-NMR [**winconv**]
- A3000 [**conv**]
- VNMR [**vconv**]
- JNMR [**jconv**]
- Felix [**fconv**]

**Open other file:**

This option allows you to open the following lists and programs:

- Pulse programs [**edpul**]
- Au programs [**edau**]
- Gradient programs [**edgp**]
- CPD programs [**edcpd**]
- Miscellaneous files [**edmisc**]
- Parameter lists [**edlist**]
- Jython program [**edpy**]

The corresponding command line commands are specified in square brackets.

After clicking **OK**, a new dialog box will appear according to the selected option and file type.

**SEE ALSO**

*conv* [▸ 333], *edau, xau, delau, xauw* [▸ 317], *edlist, dellist* [▸ 291], *edmisc, rmisc, wmisc, delmisc* [▸ 292], *edpul, edcpd, edpy, edpy3, edmac* [▸ 298], *fconv* [▸ 336], *fromjdx* [▸ 338], *fromzip* [▸ 340], *jconv* [▸ 342], *re, rep, rew, repw* [▸ 279], *reb* [▸ 281], *reb* [▸ 281], *vconv* [▸ 350], *winconv* [▸ 353]

## 9.12 paste

**NAME**

paste - Open the dataset that was last copied (nD)

**DESCRIPTION**

The command **paste** opens the dataset which was previously copied from a TopSpin data window or from the File Explorer. This involves two steps:

1. **Copy**

   In the File Explorer:

   – Go to a dataset

   – Right-click a dataset folder or file, e.g. the data *name*, *expno* or *procno* folder or any file in it and click **Copy**

2. **Paste**

   In TopSpin:

   – Click **File | Paste** or type **paste**

Note that if you select and copy a the data set in the File Explorer, its data path is copied to the Clipboard. The command **Paste** reads this path from the Clipboard. If you run **Paste** without first copying a data set from the Explorer, TopSpin tries to read whatever is currently stored in the Clipboard. If that is a data path, TopSpin will read it, otherwise you will get an error message.

**OUTPUT FILES**

*<tshome>/prog/curdir/<user>/*

*curdat* - current data definition

**SEE ALSO**

*copy* [▸ 257]

## 9.13 re, rep, rew, repw

### NAME

re - Read data of specified *name* or *expno* (nD)

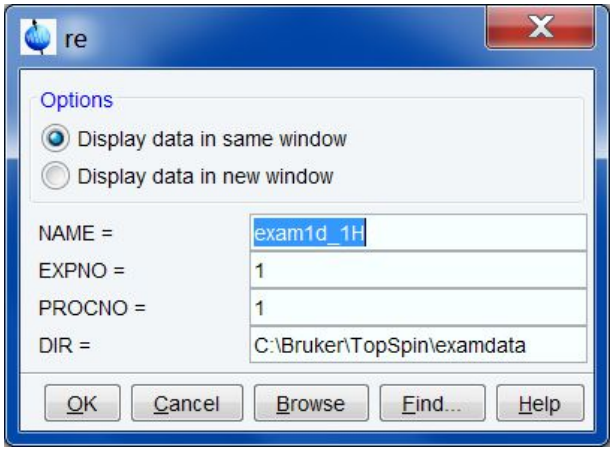
rep - Read data of specified *procno* (nD)

rew - Read data of specified *name*/*expno* in new window (nD)

repw - Read data of specified *procno* in new window (nD)

### DESCRIPTION

The commands **re** and **rew** allow you to read and display a new data set. They open a dialog box with the corresponding option selected:



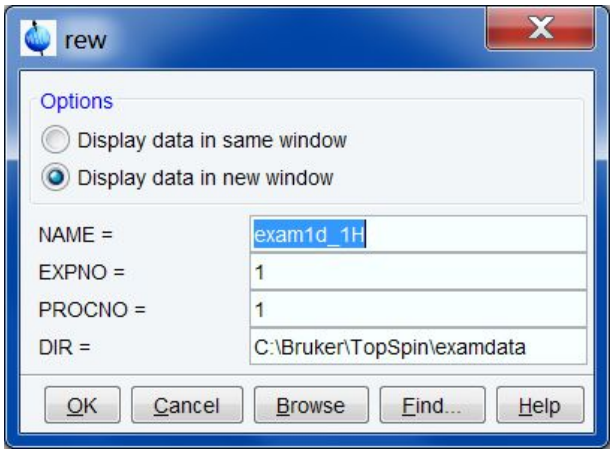

These options are:

**Display data in same window**

Selects the command **re** for execution. It reads the specified data set in the current data window.

**Display data in new window**

Selects the command **rew** for execution. It reads the specified data set in a new data window.




Specify the data path variables. A full data path is:

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

**re** replaces the data set in the current data window (if it exists).

The data path variables can also be specified on the command line. In this case, the dialog box is not opened and the missing data path variables are taken from the current data set. Examples:

**re <name>**

**re <expno>**

**re <name> <expno>**

**re <expno> <procno>**

**re <name> <expno> <procno>**

**re <name> <expno> <procno> <dir> <user>**

Alternatively, **re** and **rew** can be entered with an alias name as argument, i.e.:

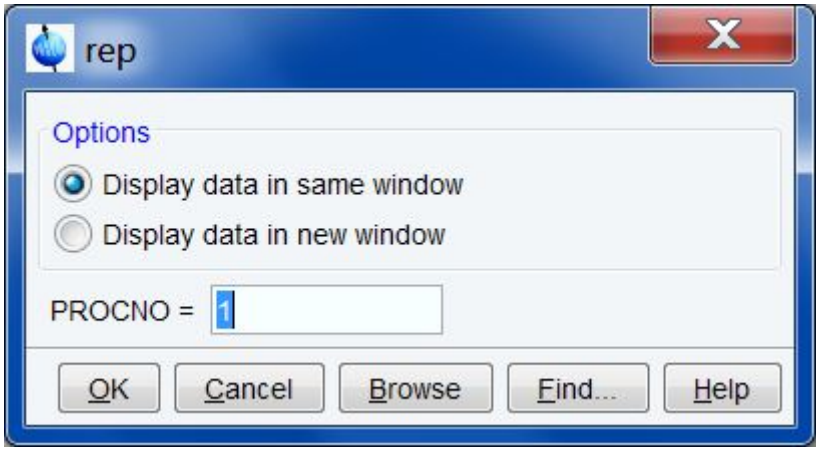
**re <aliasname>**

Note that the first alphanumeric argument is always interpreted as the name (or alias name) and the first numeric argument as experiment number.

The commands **rep** and **repw** allow you to read and display a new processed data number (*procno*) of the current data set. They open a dialog box with the corresponding option:

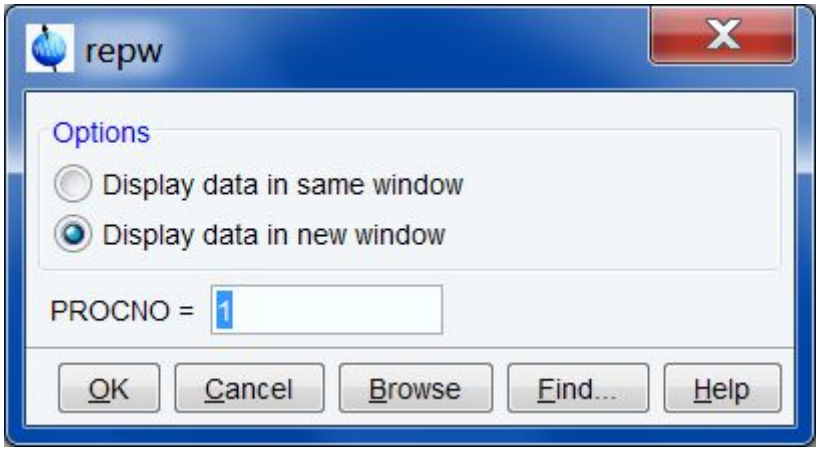
### Display data in same window

Selects the command **rep** for execution. It reads the specified PROCNO in the current data window.




### Display data in new window

Selects the command **repw** for execution. It reads the specified PROCNO in a new data window.



The destination *procno* can also be specified on the command line, e.g.: **rep 77**

**INPUT FILES**

**For re and rew:**

*<dir>/data/<user>/nmr/<name1D>/<expno>/*

*fid* - 1D raw data

*acqu* - acquisition parameters

*acqus* - acquisition status parameters

**For re, rew, rep and repw:**

*<dir>/data/<user>/nmr/<name1D>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed 1D data

*proc* - processing parameters

*procs* - processing status parameters

Note that these are only the main files of a 1D dataset.

**OUTPUT FILES**

*<tshome>/prog/curdir/<user>/*

*curdat* - current data definition

**USAGE IN AU PROGRAMS**

RE(name)

**SEE ALSO**

*reb* [▶ 281], *open* [▶ 277], *new* [▶ 275], *find, search commands* [▶ 271], *dir, dira* [▶ 265]

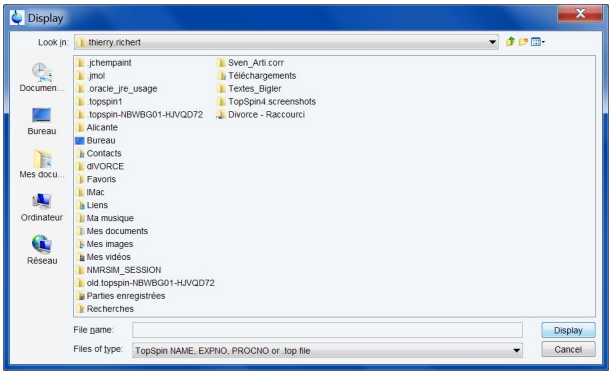
## 9.14 reb

**NAME**

reb - Open a data browser at the level of data names (nD)

**DESCRIPTION**

The command **reb** opens a file browser:



Here you see a list of data set names under the same <dir> and <user> as the currently selected data set. Note that TopSpin data are stored in a directory:

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

From the browser, you can:

- Select the data name to be displayed in the current data window
- Move up in the data directory tree to select a different *user* and/or *dir*
- Double-click a data *name* to move down the directory tree and select a desired *expno*/*procno*.

Once you have selected the desired *name*, *expno* or *procno*, click **Display** or hit **Enter** to display the data set in the current data window.

**reb** allows opening data sets stored in the following directories structures:

*<mydata>/<dataname>/<expno>/pdata/<procno>*

Note that this will create a copy the data set in the standard TopSpin data path:

*<tshome>/data/<user>/nmr/<dataname>/<expno>/pdata/<procno>*

Where <user> is the current internal TopSpin user. This copy can be processed, deleted or overwritten, even if the original data set is write protected. The original data set remains unchanged.

### SEE ALSO

*open* [▶ 277], *re, rep* [▶ 279], *new* [▶ 275], *find, search* [▶ 271]

## 9.15 rel, repl

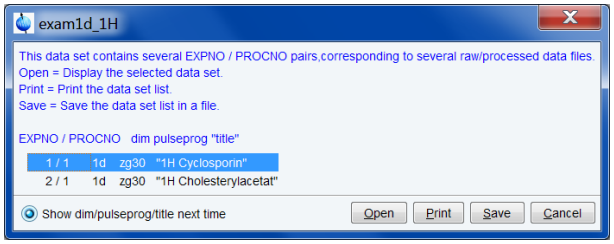
### NAME

rel - Open a list of expnos/procnos in current dataset

repl - Open a list of procnos in the current expno

### DESCRIPTION

The command **rel** lists the available expnos/procnos under the current data set and allows to select and open one:




If the current data set contains only one expno/procno combination, it is automatically opened.

The dialog offers the following buttons:

**Open** : Open the highlighted dataset (equivalent to pressing the **Enter** key)

**Print** : Print the dialog contents

**Save** : Print the dialog contents to a text file

**Cancel** : Close the dialog

The command **repl** works like **rel**, except that it lists the available procnos under the current expno.

Another dateset can also be selected directly on the command line, e.g.:

**rel 2** - selects the expno 2 of the current dataset.

**repl 2** - selects the procno 2 of the current dataset.

If no data set is open, **rel** refers to the last active dataset. If no data set has been open yet during the current TopSpin session, an error message is displayed.

### SEE ALSO

*re, rep commandr* [▷ 279], *new* [▷ 275]

## 9.16 reopen

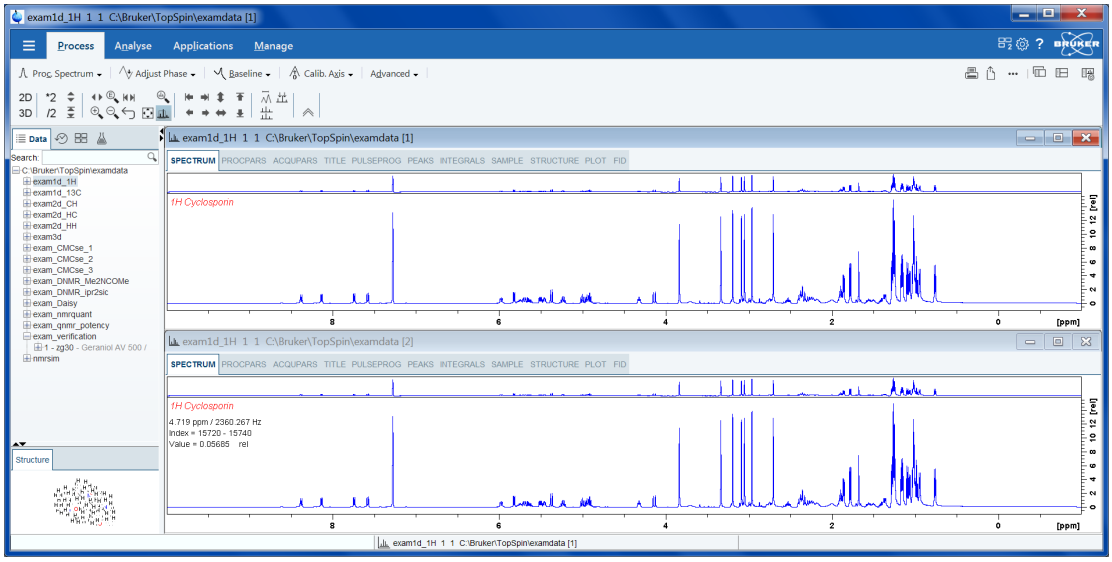
### NAME

reopen - Reopen current dataset in new data window (nD)

### DESCRIPTION

The command **reopen** reopens the current data set in a new data window. This is, for example, convenient to view various regions or various objects (spectrum, fid, parameters etc.) of the same data set. Multiple data windows are indicated with a number in square brackets, e.g. [1], in the title bar.

Entering **reopen** on the command line is equivalent to clicking **File | Reopen** in the menu.




### SEE ALSO

*open* [▷ 277]

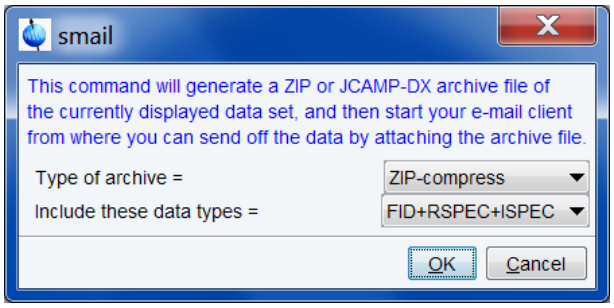
## 9.17 smail

### NAME

smail - Send the current data set by E-mail (1D, 2D, 3D)

## DESCRIPTION

The command **smail** sends the current data set by E-mail. It opens a dialog box where you can specify the required information or accept the default values.




In the dialog box, you can select the:

- Archive type: ZIP or JCAMP
- Data type(s) included: FID, spectrum and/or parameters

For ZIP format data you can choose between compression and no compression.

For JCAMP format, you can choose between the following compression modes:

- **FIX** (=0) : Table format
- **PACKED** (=1) : No spaces between the intensity values
- **SQUEEZED** (=2) : The sign of the intensity values is encoded in the first digit
- **DIFF/DUP** (=3) : The difference between successive values is encoded, suppressing repetition of successive equal values (default = **DIFF/DUP**)

For the included data types, you have the following choices:

- **FID+RSPEC+ISPEC**: Raw + real and imaginary processed data
- **FID+RSPEC**: Raw + real processed data
- **FID**: Raw data
- **RSPEC+ISPEC**: Real and imaginary processed data
- **RSPEC**: Real processed data
- **PARAMS**: Parameter files

Before you can send the data you must fill in the fields:

- **To**: The E-mail address of the recipient
- **From**: Your own E-mail address
- **SMTP mail server**:
- **Subject**:
- **Text:**

## INPUT FILES

*<tshome>/prog/curdir/<user>/*

*curdat* - current data definition

**If data type includes FID :**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - 1D raw data

*ser* - 2D raw data

**If data type includes RSPEC :**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*1r* - real processed 1D data

*2rr* - real processed 2D data

**If data type includes ISPEC :**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*1i* - imaginary processed 1D data

*2ir* - F2-imaginary processed 2D data

*2ri* - F1-imaginary processed 2D data

*2ii* - F2/F1-imaginary processed 2D data

All other files which are part of a data set like parameter files, audit trails files etc. are sent for all data types.

**OUTPUT FILES**

*<userhome>/<mydata.dx>* - TopSpin data in JCAMP-DX format

*<userhome>/<mydata.bnmr.zip>* - TopSpin data in ZIP format

**SEE ALSO**

*tojdx* [▶ 344], *tozip* [▶ 347]

## 9.18 wrpa, wra, wrp, wraparam, wrpparam

**NAME**

wrpa - Copy a complete data set, raw and processed data (nD)

wra - Copy raw data (nD)

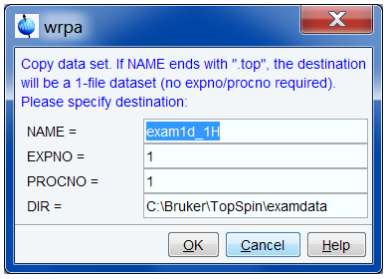
wrp - Copy processed data (nD)

wraparam - Copy acquisition data set (parameters only)

wrpparam - Copy processing data set (parameters only)

**DESCRIPTION**

The command **wrpa** writes (copies) a data set. It opens a dialog box where you can specify the destination data set:




When you click **OK**, the entire *expno* directory is copied including raw data, acquisition parameters, processed data and processing parameters.

**wrpa** takes six arguments:

**<name>** - the data set name

**<expno>** - the experiment number

**<procno>** - the processed data number

**<dir>** - the disk unit (data directory)

**<user>** - the user

**y** - overwrite the destination dataset if it already exists

All arguments are parts of the destination data path (the data path of the foreground data set is displayed above the TopSpin data field), except for the last one which is a flag. You can, but do not have to, specify all of these arguments. If the first argument is a character string, it is interpreted as the destination data name. If the first argument is an integer value, it is interpreted as the destination experiment number. Examples of using **wrpa** are:

**wrpa <name>**

**wrpa <expno>**

**wrpa <name> <expno>**

**wrpa <name> <expno> <procno>**

**wrpa <name> <expno> <procno> <dir> <user> y**

**wra** makes a copy of the current *expno* directory, including raw data, acquisition parameters, and processing parameters. The command takes two arguments and can be used as follows:

**wra** - prompts you for the destination experiment number

**wra <expno>** - copies the raw data to <expno>

**wra <expno> y** - overwrites existing raw data in <expno>

**wrp** makes a copy of the current *procno* directory, including the processed data and processing parameters. The command takes two arguments and can be used as follows:

**wrp** - prompts you for the destination processed data number

**wrp <procno>** - copies processed data to <procno>

**wrp <procno> y** - overwrites existing processed data in <procno>

**wrpparam** works like **wrp**, except that it does not copy the processed data files and *auditp.txt* file.

**wraparam** works like **wra**, except that it does not copy the raw data files and *audita.txt* file.

Note that the **wr*** commands only work if user who started TopSpin has the permission to create the destination data set.

## INPUT AND OUTPUT FILES

For **wrpa**, **wra** and **wraparam**:

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - 1D raw data

*ser* - 2D or 3D raw data

*acqu* - acquisition parameters

*acqus* - acquisition status parameters

For **wrpa** and **wra** :

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - raw data (1D)

*ser* - raw data (nD)

*audita.txt* - acquisition audit trail

For **wrpa**, **wra**, **wrp** and **wrpparam**:

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*proc* - processing parameters

*procs* - processing status parameters

For **wrpa**, **wra** and **wrp**:

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*1r*, *1i* - processed data (1D)

*2rr*, *2ir*, *2ri*, *2ii* - processed data (2D)

*3rrr*, *3irr*, *3rir*, *3rri*, *3iii* - processed data (3D)

*4rrrr*, *4iiii* - processed 4D data

*auditp.txt* - processing audit trail

For 2D data, the additional parameter files *acqu2*, *acqu2s*, *proc2* and *proc2s* will be created. For 3D, 4D etc. data, the respective additional parameter files will be created.

Note that apart from data and parameters several other files are copied.

## USAGE IN AU PROGRAMS

WRPA(name, expno, procno, diskunit, user)

WRA(expno)

WRP(procno)

Note that these macros overwrite possibly existing data.

## SEE ALSO

*dir, dira, dirp, dirdat, browse* [▶ 265], *new* [▶ 275], *open* [▶ 277], *re, rep, rew, repw* [▶ 279], *reb* [▶ 281]

# 10 Parameters, Lists, AU Programs

This chapter describes all TopSpin commands which handle parameters and parameter sets. Furthermore, you will find commands that are used to read or edit lists like pulse programs, gradient programs, frequency lists etc. Note that several commands in this chapter are acquisition related rather than processing related. Nevertheless they play a role in the processing part of TopSpin.

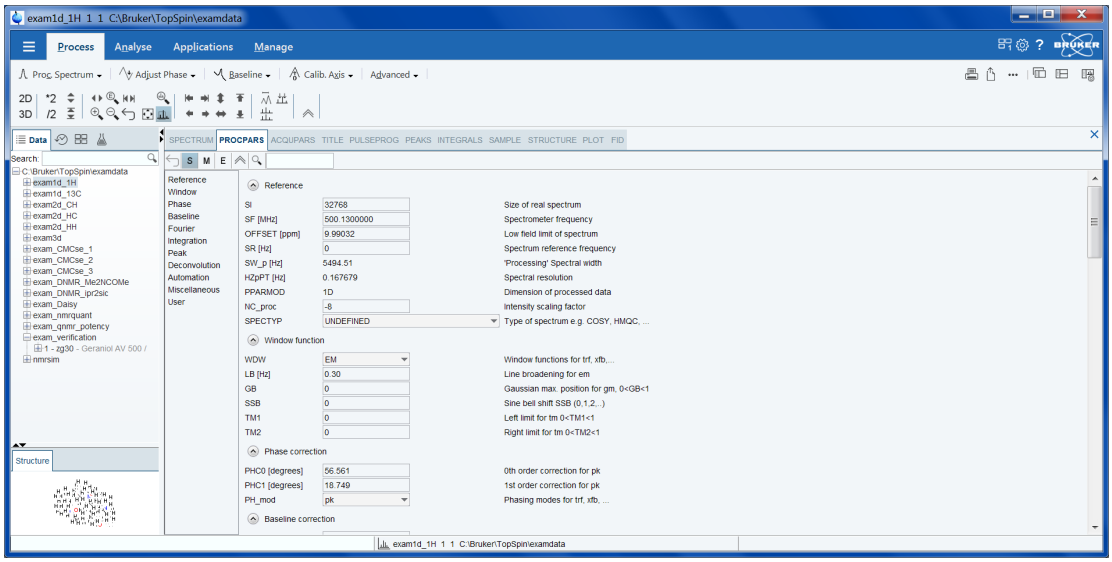
## 10.1 dpp

### NAME

dpp - Displays processing status parameters (1D, 2D, 3D)

### DESCRIPTION

The command **dpp** displays the processing status parameters. Entering **dpp** is equivalent with a click on the ProcPars tab and **Toggle status parameter view** in the dataset window.


The processing status parameters are set by processing commands and represent the status of the processed data. As such, they can only be viewed in the **dpp** window.

The following buttons are available:

 Undo the last modification (unused for status parameters).

 Switches between processing and processing status parameters.

 Changes the processing dimension of the current dataset.

 Show eretic parameters.

 Switches to Maxent parameters view.

 Collapse or  expand all parameters.

 Search for the parameter specified in the search field.

Processing status parameters can also be viewed by entering their names on the command line. For example:

- **s ft_mod** - Display the processing status parameter FT_mod.
- **s nc_proc** - Display the processing status parameter NC_proc.

### INPUT FILES

*<tshome>/classes/prop/*

*pared.prop* - Parameter properties file.

*<tshome>/exp/stan/nmr/form/*

*proc.e* - Processing parameter format file.

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*procs* - Processing status parameters.

On 2D and 3D data the files *proc2s* and *proc3s* are used for the second and third direction, respectively (see also chapter *Parameter files [▶ 20]*).

### SEE ALSO

*edp [▶ 297]*, *dpl [▶ 247]*

## 10.2 eddosy

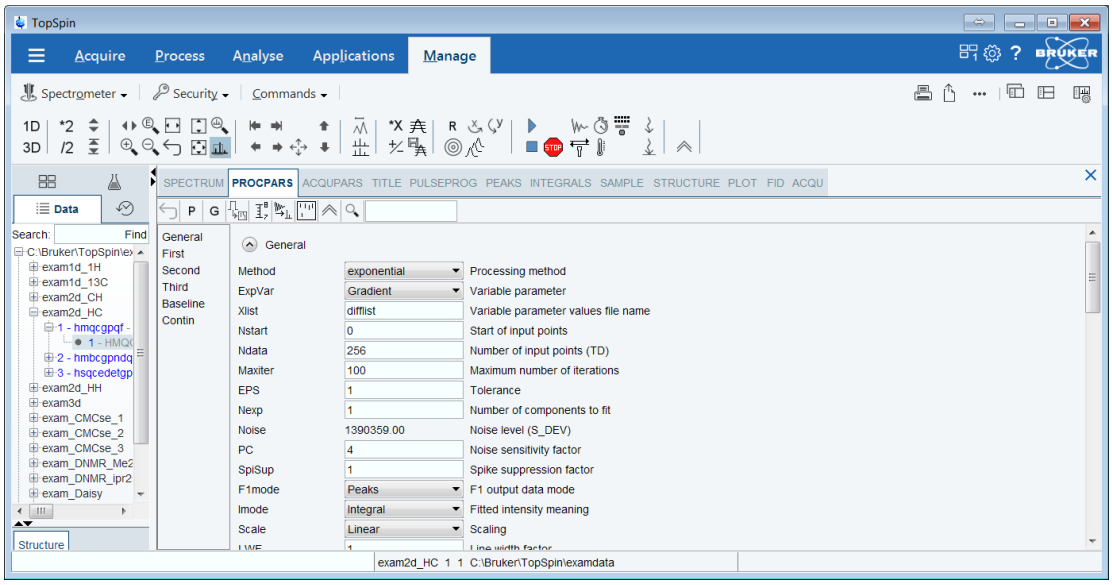
### NAME

eddosy - Edit DOSY processing parameters (2D, 3D)

### DESCRIPTION

The command **eddosy** opens the following dialog box, if no DOSY parameters are available.


- Click **OK** to set the DOSY processing parameters.




These parameters are used by the command **dosy2d** and **dosy3d** on 2D and 3D data, respectively.

The following buttons are available:

Undo the last modification. Can be used repeatedly.

P Switch to processing parameters.

G Switch to Gifa parameters.

Copy parameters from experiment (AU program **setdiffparm**).

Get display limits from data set.

Execute Fourier Transform (command **xf2**).

Start fitting.

Search for the parameter specified in the search field.

For more information on **eddosy**:

Click **Help | Manuals | Acquisition Application Manuals | Dosy**

## INPUT FILES

*<tshome>/exp/stan/nmr/form/*

*dosy.e* - format file for **eddosy**

## INPUT AND OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*dosy* - DOSY processing parameters

## SEE ALSO

*dosy2d* [▸ 213], *dosy3d* [▸ 214]

## 10.3 edlist, dellist

### NAME

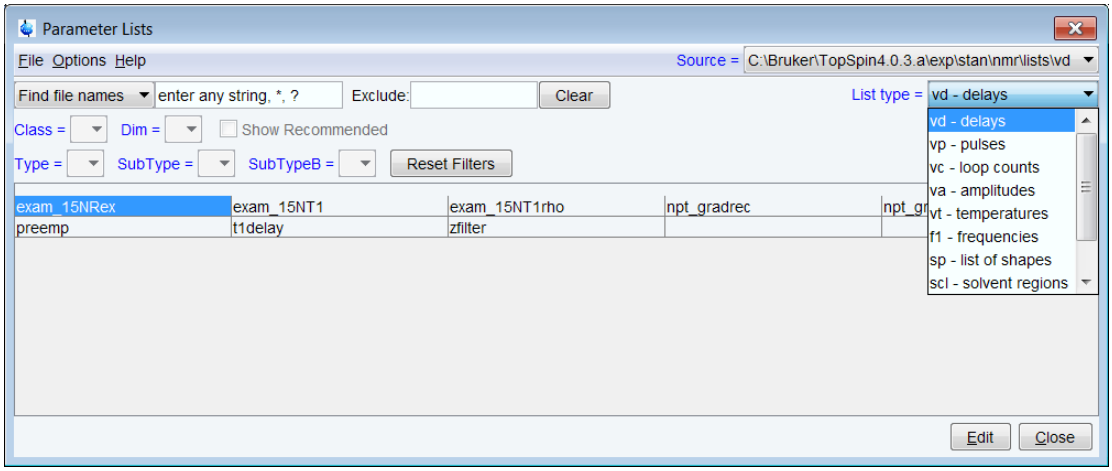
edlist - Edit Parameter lists

dellist – Delete Parameter lists

### DESCRIPTION

The command **edlist** allows to edit parameter lists like VD Delay lists, VP Pulse lists, VC Loop Counts lists, VA Amplitude lists, VT Temperature lists, F1 Frequency lists, SP Shape lists, DS Data Set lists and PHASE Phases lists.

The command edlist opens the Parameter Lists window:




On the top right the source and list type can be filtered. All items shown in the table can be edited in the upcoming text editor.

For detailed information user-specific definition of **Source Directories** and the functionalities of **Manage Source Directories** please refer to the information given in chapter *User specific handling of Source Directories [▶ 13]*.

The dialog shown above offers the following buttons:

**Edit** - to edit a list in a text file, click **Edit** or double-click a parameter list. Saving the modifications will overwrite the existing list.

**Close** - closes the dialog.

The command **dellist** opens the same dialog box as **edlist**. To delete a list, right-click the selected item, and then click **Delete**…

**Hint:** This is also possible with the command **edlist**, so **dellist** is historical and obsolete.

For further information about the commands **edlist** an **dellist** please refer to the Acquisition Commands and Parameters User Manual under:

Help | Manuals (docs) | Acquisition & Processing References | Acqu. Commands and Parameters.

### INPUT/OUTPUT DIRECTORIES

**The default directory for user-defined lists is:**

*<tshome>/exp/stan/nmr/lists/<listname>/user*

### SEE ALSO

*edmisc, rmisc commandr [▷ 292]*

## 10.4 edmisc, rmisc, wmisc, delmisc

### NAME

edmisc - Edit miscellaneous lists

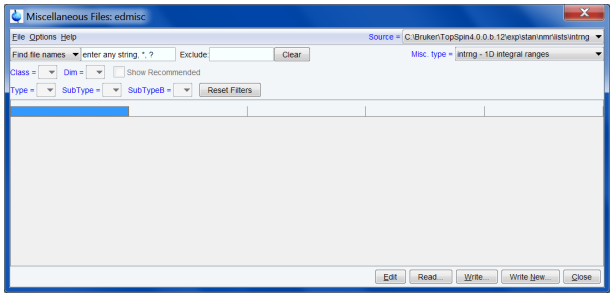
rmisc - Read miscellaneous lists

wmisc - Write miscellaneous lists

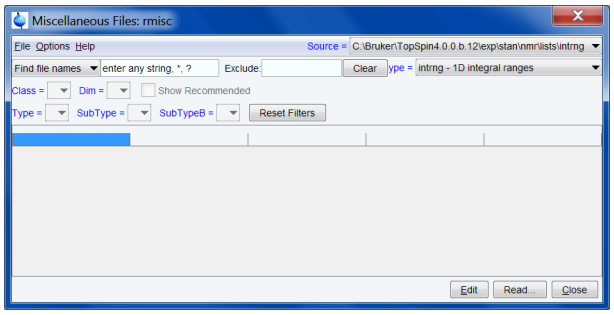
delmisc - Delete miscellaneous lists

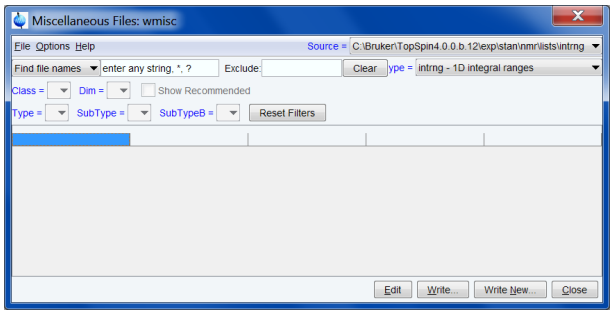
### DESCRIPTION

The commands ***misc** allow to read, edit, write or delete miscellaneous lists. When entered without arguments, they all open a related window for miscellaneous files. The difference is that **wmisc** only offers writing possibilities for miscellaneous files, **rmisc** only offers reading possibility, whereas with **edmisc** and **delmisc** you can read, write and edit the correesponding/selected miscellaneous file:




On the top right you can change the source and specify the miscellaneous type that should be shown in the table (see figure above). All items shown in the table can be edited, read, written or new written. This also corresponds to the commands **edmisc**, **rmisc** and **wmisc**.



For detailed information about user-specific definition of source directories and the functionalities of **Manage Source Directories** please refer to the information given in chapter *User specific handling of Source Directories [ 13]*.

### Types of Miscellaneous Files

The lists which can be edited are shown in the table below:

| list type | contains |
|---|---|
| *intrng* | integral regions, created by interactive integration or automatic baseline correction (**abs**). Used for spectrum display, print and integral listing. |
| *base_info* | *polynomial*, *sine* or *exponential* baseline function, created from the baseline mode (**.basl**). Used by the baseline correction command **bcm**.. |
| *baslpnts* | baseline points created by *def-pts* from the baseline mode (**.basl**). Used by the spline baseline correction command **sab**. |
| *peaklist* | peak information, created by the command **ppp** and **mdcon auto**. Used by the mixed deconvolution command **mdcon**. |
| *reg* | plot regions, created in interactive integration mode (command **.int**). Used by **pp**, **lipp** when PSCAL=ireg or pireg. |
| Miscellaneous list types | |

When entered on the command line, **rmisc** takes two arguments and can be used as follows:

- **rmisc <type> -** Shows all entries of the type <type>. If you select an entry, the corresponding list will be read.
- **rmisc <type> <name>** - Reads the list <name> of the type <type>.

### INPUT/OUTPUT DIRECTORIES

The default directory for user-defined lists is:

*<tshome>/exp/stan/nmr/lists/<listname>/user*

intrng - integral range files

baslpnts - spline baseline points file

base_info - pol. exp. or sine baseline function files

peaklist - peak information files

reg - plot region files

### USAGE IN AU PROGRAMS

RMISC(type, file)

WMISC(type, file)

### SEE ALSO

*edlist, dellist [ 291]*

## 10.5 edshape

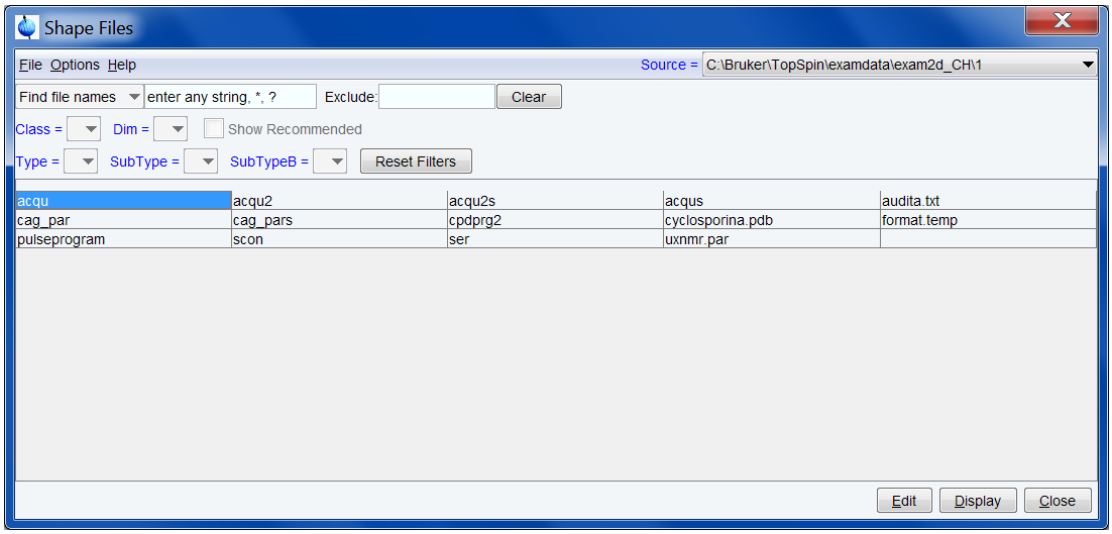
### NAME

edshape - Edit Shape Files

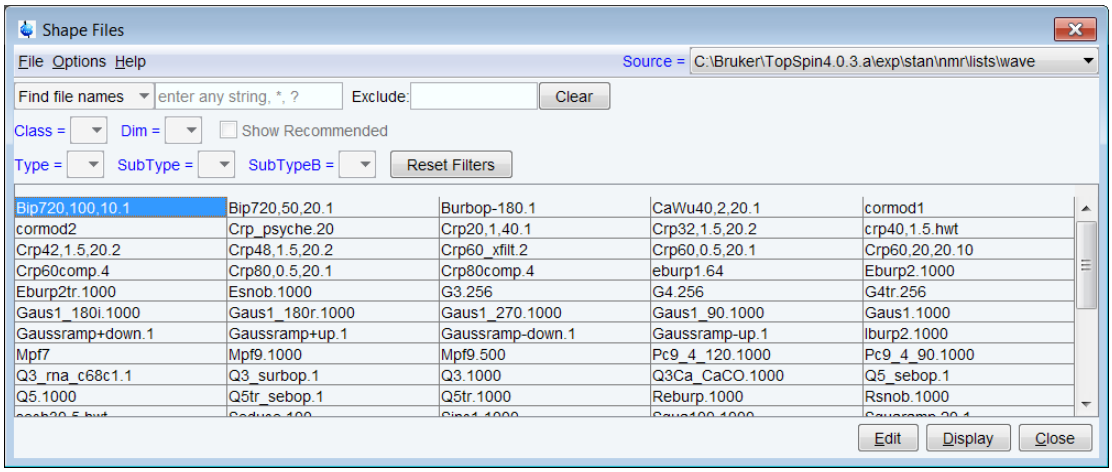
delshape - Delete Shape Files

### DESCRIPTION

When entered without arguments, the Shape File commands **edshape and delshape** all open the AU program dialog box:




On the top right of the upcoming window you can find the sources where the listed Shape files are stored. With pull-down menu and click on the respective Source you can change the Shape file source to let them be listed in this dialog.




The AU programs are selected from the **Source** directory as selected at the upper right of the dialog. Note that:

*<tshome>\exp\stan\nmr\lists\wave* contains all Bruker Shape files.

*<tshome>\exp\stan\nmr\lists\wave\user* contains all user defined Shape files.

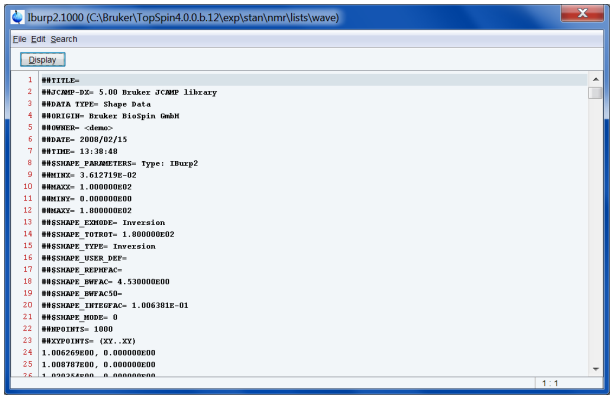
The dialog offers the following buttons:

### Close

Close the dialog.

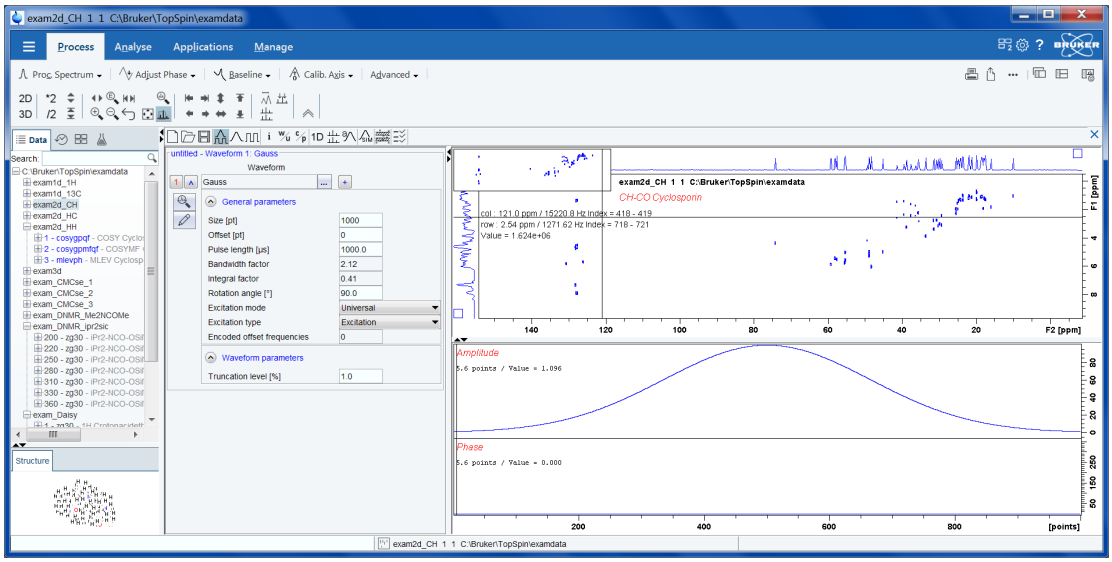
### Edit

Edit the selected Shape file. Equivalent to double-clicking the Shape file name, or entering **edshape <name>** on the command line.




### Display

Display the selected Shape file. The Shape Tool will be opened for display the current Shape file. The result can be seen in the following figure:




## The File menu

The File menu offers the following functions:

### New...

Create a new Shape file. Note that new Shape files can only be stored in user defined directories.

### Save as...

Save the selected Shape files under a new name. A dialog will appear where you can specify the Shape file name and destination directory.

### Delete...

Delete the selected Shape file.

**Rename...**

Rename the selected Shape file. Note that only user defined Shape files can be renamed.

**Export...**

Export the selected Shape file to an arbitrary directory. A file dialog will appear where you can select/specify the destination directory.

**Import...**

Import a Shape file from an arbitrary directory. A file dialog will appear where you can select/specify the Shape file.

**Close**

Close the Shape file lists.

### The Options menu

The Options menu offers the following functions:

**Show Comment**

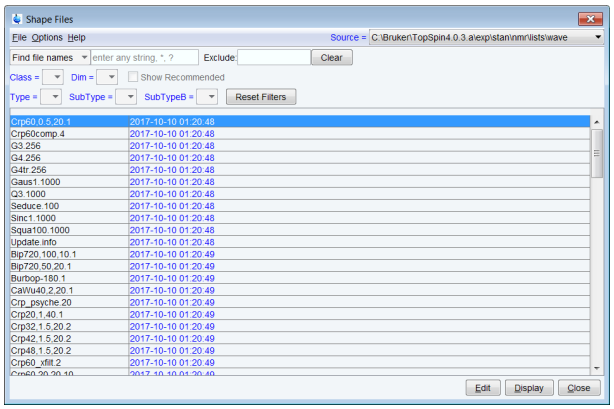
Toggles between displaying Shape file with/without comments (see the figure below).

**Show Date**

Toggles between displaying Shape file with/without date (see figure below).

**Sort by Date**

Sort Shape files by date when selected:




### Manage Source Directories

Add/modify Shape files source directories. Shape files will be searched in the order of the specified directories.

Detailed information about **Manage Source Directories** are described in chapter *User specific handling of Source Directories* [▶ 13].

### INPUT/OUTPUT FILES

The default directory for user-defined files is:

*<tshome>/exp/stan/nmr/lists/<listname>/user*

### SEE ALSO

*edlist, dellist command* [▶ 291]

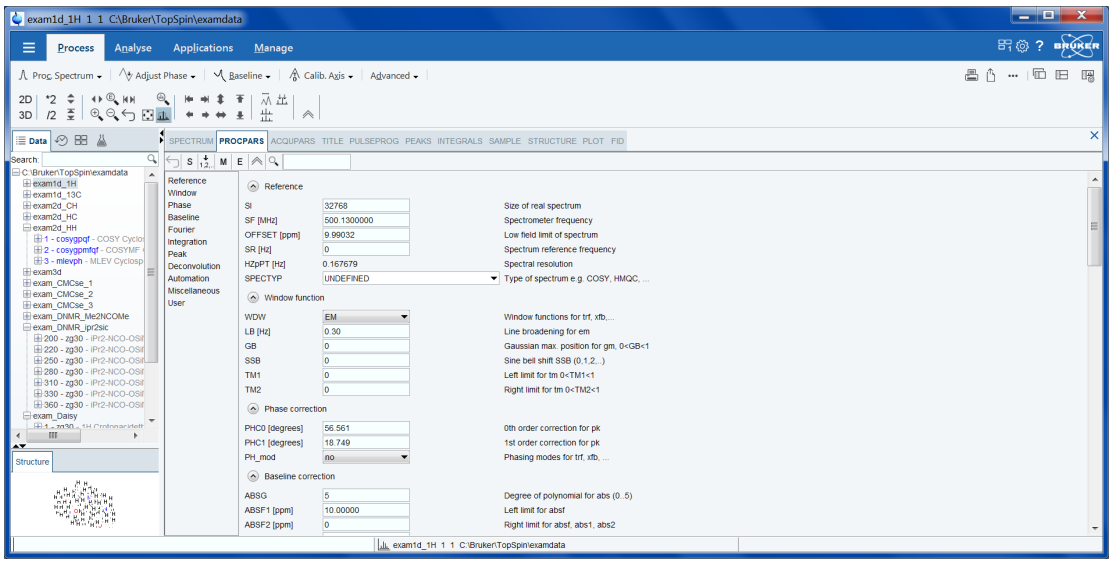
## 10.6 edp

### NAME

edp - Edit processing parameters (1D, 2D, 3D)

### DESCRIPTION

The command **edp** opens a dialog box in which you can set all processing parameters.




Entering **edp** on the command line is equivalent with a click on the ProcPars tab bar of the dataset window.

The following buttons are available:

Undo the last modification. Can be used repeatedly.

Switch to Maxent parameters

Switch to processing status parameters

Change raw data set dimensionality (parameter PPARMOD)

Search for the parameter specified in the search field

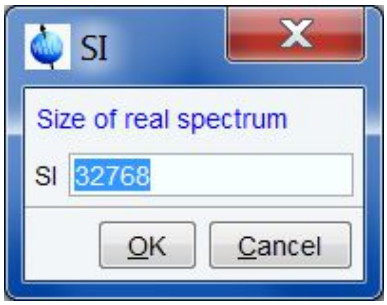
Inside the parameter editor, you can do the following actions:

- Click a processing step, e.g. Window at the left of the dialog box. The step becomes highlighted and the corresponding parameters will appear in the right part of the dialog box.
- Click in a parameter field, e.g. SI to set the parameter value. It is automatically stored.
- Hit the **Tab** key to jump to the next parameter field.
- Hit **Shift-Tab** to jump to the previous parameter field.
- Use the scroll bar at the right of the dialog box to move to parameters further up or down in the dialog box.

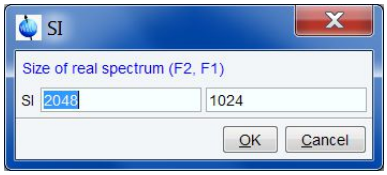
Note that you can also set parameters by entering their names on the command line. A dialog window will appear where you can enter the parameter value(s). For example:

**si**

On a 1D data set.




Or on a 2D data set:




Alternatively, you can specify the parameter value as an argument on the command line, for example:

**si 4k**

The size will be set to 4k (=4096).

### INPUT AND OUTPUT PARAMETERS

All processing parameters.

### INPUT FILES

*<tshome>/classes/prop/*

*pared.prop* - parameter properties file

*<tshome>/exp/stan/nmr/form/*

*proc.e* - format file for **edp**

### INPUT AND OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*proc* - processing parameters

*proc2* - processing parameters for the second direction (2D or 3D)

*proc3* - processing parameters for the third direction (3D)

### SEE ALSO

*dpp* [▶ 288], *edau, xau, delau, xauw* [▶ 317]

## 10.7 edpul, edcpd, edpy, edpy3, edmac

### NAME

edpul - Edit pulse programs

edcpd - Edit composite pulse decoupling (CPD) programs

edpy - Edit Jython programs

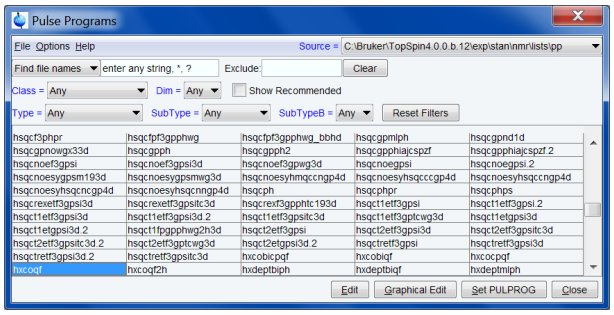
edpy3 – Edit Python 3 programs

edmac - Edit macros

## DESCRIPTION

The commands **edpul**, **edcpd**, **edpy** and **edmac** open a dialog that lists pulse programs, CPD programs, Python programs and macros, respectively. The dialog offers various functions like edit, create, search, delete, import and export. These programs are stored in a database.

The dialog for the command **edpul** is shown in the figure below. The dialogs for **edcpd**, **edpy** and **edmac** have the same menu but can offer different buttons.




### Search List Box

Database items can be searched in two possible ways, as can be chosen from the list box at the upper left of the dialog:

- *Search in names* - to search for a string in the item names.
- *Search in text* - to search for a string in item text contents.

### Search Text Field

Here you can enter one or more characters of the item name or contents. The following wildcards can be used:

* : for zero or more occurrences of any character

? : for a single occurrence of any character

Here are some examples:

- *xxx* finds all occurrences of xxx.
- ??xxx* finds all occurrences of xxx preceded by two arbitrary characters.

A search mask for item names can also be specified on the command line, e.g. **edpul ?? cos***

### Conditional List boxes

These list boxes are only offered if the selected item has the corresponding item defined. For example, most high resolution pulse programs have a Class and Dim definition but not Type or SubType definition.

### Class

Allows to show a particular class of items or all items (any).

### Dim

Allows to show items of a particular dataset dimension or all items (any).

### Type

Allows to show a particular type of items or all items (any).

### SubType

Allows to show items of a particular subtype of items or all items (any).

## Available Buttons

### All

Show items of all classes, dimensions, types and subtypes.

**Edit**

Opens the selected item (pulse program, CPD program, ...) in the TopSpin text editor or viewer, depending on whether the selected item is writable for the current user or not (see below). Writable items can be modified in the editor. They can be saved from the editor as follows:

In the menu click **File | Save** [**Ctrl-s**]

Write-protected items can be saved under a different name as follows:

In the menu click **File | Save as..**

The new item is owned by and writable for the current TopSpin user.

Items can also be created /modified with an external (non-TopSpin) editor. They can then be imported in the database as described below.

*Graphical Edit* (for pulse programs only)

Opens a symbolic graphical display of the selected pulse program, with the possibility of graphical editing.

*Set PULPROG* (for pulse programs only)

Sets the acquisition parameter PULPPROG to the name of the selected pulse program.

### The Options menu

The Options menu offers the following functions:

**Show Comment**

Toggles between displaying items with/without comments.

**Show Date**

Toggles between displaying items with/without date.

**Sort by Date**

Sort items by date when selected.

**Manage Source Directories**

Add/modify item source directories. Items will be searched for in the order of the directories specified.

For detailed information about Source Directory Handling and **Manage Source Directories** please refer to chapter *User specific handling of Source Directories [▷ 13]*.

**Export Sources...**

Opens a dialog to export an entire item library to a user defined directory. Note the difference to the **Export** function under the File menu (see below).

### The File menu

The File menu offers the following functions:

**New**

Opens an empty editor for creating a new item, e.g. a pulse program. Saving the text will prompt you for the item name, and will store it in the database. The owner of the item will be the current TopSpin user.

**Save As...**

Saves the selected item under a new name. Opens a dialog where you can selected a source directory and specify a filename.

**Delete...**

Deletes all selected items from the database (if not write protected). You will be prompted to confirm deletion.

**Rename...**

Allows to rename the selected item in the database (if not write protected).

**Export...**

Exports one or more items to text files. To do that:

1. Mark one or more items in the dialog.
2. Click **File | Export** 

3. Select or enter the storage directory and click **Export...**

The selected item(s) will be stored under their original names, provided there is write permission.

**Import...**

Imports external item (e.g. pulse program) files into the database and lists it in the dialog. First, it opens a file browser where you can navigate to a directory containing your text files (which may have been created outside of TopSpin). Select or enter the desired files in the browser and click **Import**. The dialog will be updated showing the imported item. Please note that:

- The owner of imported items is the current TopSpin user.
- Write-protected items in the database cannot be overwritten by importing items with the same name.
- Writable items with the same name are only overwritten by import, after user confirmation.
-

**Close**

Close the dialog

**Current TopSpin User**

The current TopSpin user can be one of the following users:

- The system login user, i.e. the user who started TopSpin. This is the case if **TopSpin internal login/logoff**is disabled.
- The current internal TopSpin user. This is the case if **TopSpin internal login/logoff** is enabled.

To enable/disable **TopSpin internal login/logoff**, enter **set** and click **Change** to the right of the item *Setup users for internal...*.

**Write Protection**

An item (e.g. pulse program) in the database is write-protected (cannot be modified or deleted), if its owner is *Bruker* or if its owner is not the current TopSpin user.

**Owner**

Each item (e.g. pulse program) in the database has an assigned owner. Please note the following aspects:

- For all items (e.g. pulse programs) delivered by Bruker, the owner is *Bruker*.

- The description of the *Edit, New and Import* functions above shows how an owner is assigned to an item.
- Bruker-owned items are write protected (cannot be changed/deleted). They may, however, be copied to a new name (see *Edit* above).
- Pulse programs names MUST be unique across all owners! The database cannot contain two pulse programs with same name, even if their assigned owners are different.

### Using Pulse/CPD Programs from a User-defined Directory

When you run an acquisition, using commands like **zg**, **gs**, .., the required pulse or CPD program is normally taken from the database. You might, however, want to use pulse programs from an arbitrary, user-defined directory, e.g. for development purposes. You can do this by setting the operating system environment variables *PULPPROG_DIR* and *CPDPROG_DIR*. They can be set in two different ways, with or without a minus sign, determining the item search order.

Examples:

- **PULPPROG_DIR=c:\mydir**
- Will cause **zg**, **gs**... to search for the pulse program in the database and then, if it did not find it there, in *c:\mydir*. So the database is searched first, then the defined directory.
- **PULPPROG_DIR=-c:\mydir**
- Will cause **zg**, **gs**... to search for the pulse program in *c:\mydir*, and then, if it did not find it there, in the database. So the directory is searched first, then the database.

Each time a pulse or CPD program is taken from a directory (rather than from the database), a message is written into the history file (to be viewed with command **hist**).

Please note:

- The commands **edpul** and **edcpd** do not evaluate the above environment variables.
- When TopSpin is running as a client that controls a remote spectrometer, the remote environment variables are evaluated.

### About Macros

Macros are text files which contain a sequence of TopSpin commands and/or Jython commands. A simple macro for processing and plotting the current dataset is:

```
# 1D processing macro
em
ft
apk
sref
autoplot # plot according to Plot Editor layout
```

TopSpin commands can be inserted in lower or uppercase letters. Jython commands must be entered as follows:

**xpy <name>**

All text behind a # character is treated as comment.

### About Jython and Python3 programs

Python programming is extensively described in a separate document available under:

Click **Help | Manuals | Programming Manuals | Python programming**

### INPUT AND OUTPUT FILES

The default directories for pulse programs, CPD programs, Macros and Python programs are listed below, just like Bruker default directories:

*<tshome>/exp/stan/nmr/lists/pp/** - Bruker pulse programs

*<tshome>/exp/stan/nmr/lists/pp/user/** - User defined pulse programs

*<tshome>/exp/stan/nmr/lists/cpd/** - Bruker/CPD programs

*<tshome>/exp/stan/nmr/lists/cpd/user/** - User CPD programs

*<tshome>/exp/stan/nmr/lists/mac/** - Bruker TopSpin macros

*<tshome>/exp/stan/nmr/lists/mac/user/** - User TopSpin macros

*<tshome>/exp/stan/nmr/py/** - Bruker Jython programs

*<tshome>/exp/stan/nmr/py/user/** - User Jython programs

### SEE ALSO

*edlist, dellist* [▶ 291], *delpul, delcpdd* [▶ 303], *xmac* [▶ 309], *xpy* [▶ 309], xpy3

## 10.8 delpul, delcpd, delpy, delmac

### NAME

delpul - Delete pulse programs

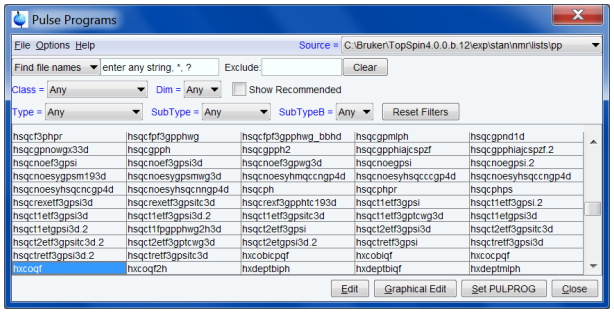
delcpd - Delete composite pulse decoupling (CPD) programs

delmac - Delete macros

delpy - Delete Jython programs

### DESCRIPTION

The commands **delpul**, **delcpd**, **delpy** and **delmac** open a dialog from which you can delete pulse programs, CPD programs, Jython programs and macros, respectively. These programs are stored in a database. The commands open the same dialog as the corresponding commands **edpul**, **edcpd**, etc. (see the description of these commands):




To delete a list, right-click the selected item, and then click **Delete**…

Confirm the warning with **OK**.

### INPUT FILES

*<tshome>/exp/stan/nmr/lists/pp/** - pulse programs

*<tshome>/exp/stan/nmr/lists/cpd/** - CPD programs

*<tshome>/exp/stan/nmr/lists/mac/** - TopSpin macros

*<tshome>/exp/stan/nmr/py/** - Jython programs

### SEE ALSO

*edpul, edcpde* [▶ 298], *xpy* [▶ 309], *xmac* [▶ 309]

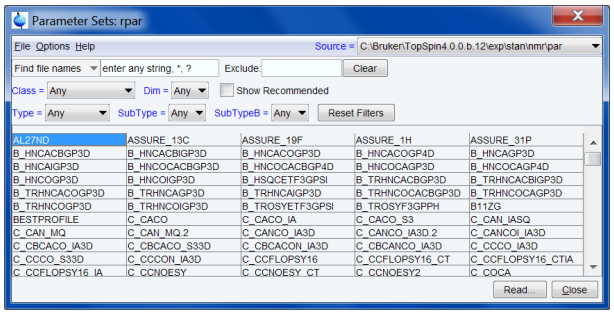
## 10.9 rpar

### NAME

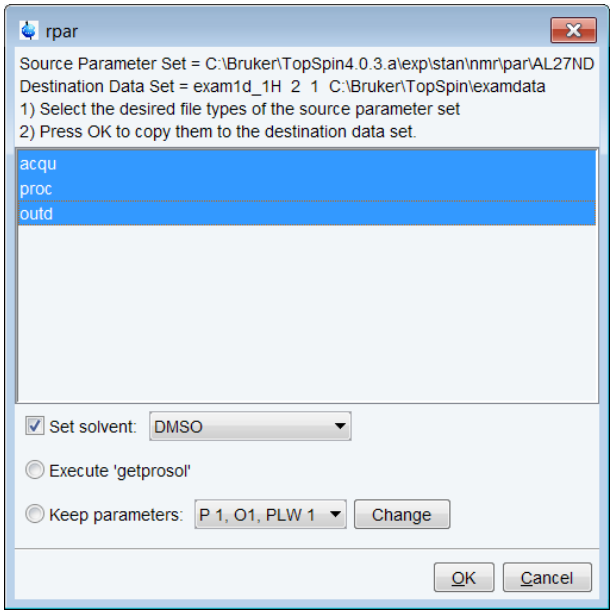
rpar - Read a parameter set (1D, 2D, 3D)

### DESCRIPTION

The command **rpar** reads a parameter set (experiment) to the current data set. When it is entered without arguments, **rpar** opens a dialog box with a list of available parameter sets.




Here you can select a **Source** directory at the upper right of the dialog, then select a parameter set and click **Read**... to read it to the current data set (for detailed information please refer to chapter *User specific handling of Source Directories [▶ 13]*). This will open the dialog:




In this dialog, you can select the file types to be read, or just click **OK** to read all types.

The following buttons are available:

**Read...**

Read the parameters of the selected parameter set to the current data set.

**Close**

Close the **rpar** dialog.

**rpar** can be used with arguments:

- **rpar <name>**
- Opens a dialog box where you can select individual parameter files of the parameter set <name>. Upon clicking **OK**, this file is copied to the current data set.
- **rpar <name> acqu**
- Reads the acquisition parameters (file *acqu*) of the parameter set <name> to the current data set.
- **rpar <name> proc**
- Reads the processing parameters (file *proc*) of the parameter set <name> to the current data set.
- **rpar <name> acqu proc**
- Reads the acquisition and processing parameters (files *acqu* and *proc*) of the parameter set <name> to the current data set.
- **rpar <name> all**
- Reads all parameter files of the parameter set <name> to the current data set.
- **rpar <name> all remove=yes**
- Reads all parameter files of the parameter set <name> to the current data set, deleting all data files and all status parameters.

The first argument may contain wildcards, e.g.:

- **rpar C*** shows all parameter sets beginning with the letter C.

The **remove=yes** argument can be used together with any other argument.

After reading a parameter set with **rpar**, you can modify parameters of the various types with the commands:

- **eda** - acqu parameters
- **edp** - processing parameters

Note that Bruker parameter sets contain all parameter types, but user defined parameter sets contain only those parameter types that were stored when the parameter set was created (see **wpar**). Usually, however, user defined parameter sets are also stored with all parameter types.

Bruker parameter sets are delivered with TopSpin and installed with the command **expinstall**.

User defined parameter sets are created with **wpar**, which stores the parameters of the current data set under a new or existing parameter set name.

**rpar** allows to read parameters sets of various dimensionalities, 1D, 2D, etc. If the dimensionality of the current data set and the parameter set you want to read are the same, e.g. both 1D, the current parameter files are overwritten. If the current data set contains data (raw and/or processed data), these are kept. Furthermore, the status parameters are kept so you still have a consistent data set. However, as soon as you process the data, the new processing parameters are used, the processed data files are overwritten and the processing status parameters are updated. When you start an acquisition, the new acquisition parameters are used, the raw data are overwritten and the acquisition status parameters are updated.

If the dimensionality of the current data set and the parameter set you want to read are different, the current parameter files are overwritten, all data files are deleted and status parameters are kept. If the dimensionality is reduced, the superfluous parameter files are deleted.

## INPUT FILES

*<tshome>/exp/stan/nmr/par/<1D parameter set>/*

*acqu* - acquisition parameters

*proc* - processing parameters
*outd* - output device parameters
*<tshome>/exp/stan/nmr/par/<2D parameter set>/*
*acqu* - F2 acquisition parameters
*acqu2*- F1 acquisition parameters
*proc* - F2 processing parameters
*proc2* - F1 processing parameters
*outd* - output device parameters
*clevels* - 2D contour levels
3D parameter sets also contain the files *acqu3* and *proc3* for the third direction.

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<1D data name>/<expno>/*
*acqu* - acquisition parameters
*<dir>/data/<user>/nmr/<1D data name>/<expno>/pdata/<procno>/*
*proc* - processing parameters
*outd* - output device parameters
*<dir>/data/<user>/nmr/<2D data name>/<expno>/*
*acqu* - F2 acquisition parameters
*acqu2* - F1 acquisition parameters
*<dir>/data/<user>/nmr/<2D data name>/<expno>/pdata/<procno>/*
*proc* - F2 processing parameters
*proc2* - F1 processing parameters
*outd* - output device parameters
*clevels* - 2D contour levels
The default directory for user defined parameter sets is:
*<tshome>/exp/stan/nmr/par/user*

**USAGE IN AU PROGRAMS**

RPAR(name, type)

**SEE ALSO**

*wpar, edpar commande [▶ 306]*, (delpar), (expinstall)

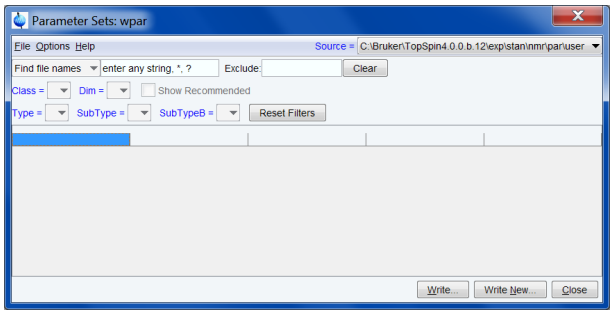
## 10.10 wpar, edpar

**NAME**

wpar - Write a parameter set
edpar - Edit a parameter set

**DESCRIPTION**

The command **wpar** stores the parameters of the current data set in a parameter set. It opens a dialog box where you can select an experiment name and then click **Write**.. to store it or click **Write New**... to store the them under a new name:




The command **edpar** opens a similar dialog as the **rpar** and **wpar** commands. The difference to **wpar** and **rpar** is that with **edpar** parameter sets can be read, written, written new and edited, whereas **rpar** only offer reading possibilities for parameter sets and **wpar** gives the possibility to write and create (button **Write New ...**) parameter sets. Same possibilities as **edpar** offers the command **delpar**.

The following buttons are available:

**Write...**

Write the parameters of the current data set to the selected parameter set.

**Write New...**

Write the parameters of the current data set to a new experiment name. You will be prompted to enter this name.

**Close**

Close the **wpar** dialog.

The parameters are written to the **Source** directory as selected at the upper right of the dialog.

**wpar** can be used with arguments:

- **wpar <name>**
- Opens a dialog box where you can select individual parameter files of the parameter set <name>. Upon clicking **OK**, this file is copied to the current data set.
- **wpar <name> acqu**
- Reads the acquisition parameters (file acqu) of the parameter set <name> to the current data set.
- **wpar <name> proc**
- Reads the processing parameters (file *proc*) of the parameter set <name> to the current data set.
- **wpar <name> acqu proc**
- Reads the acquisition and processing parameters (files *acqu* and *proc*) of the parameter set <name> to the current data set.
- **wpar <name> all**
- Reads all parameter files of the parameter set <name> to the current data set.

The first argument may contain wildcards, e.g.:

- **wpar C*** shows all parameter sets beginning with the letter C

Bruker standard experiment names should not be used when storing your own experiments with **wpar**. The reason is that they are overwritten when a new version of TopSpin is installed.

**wpar** is often used in the following way:

1. Define a new data set with the command **new**.
2. Enter **rpar** to read a Bruker parameter set which defines the experiment you want to do.

3. Modify the acquisition parameters (with **eda**) to your preference and run the acquisition.
4. Modify processing parameters (with **edp**) to your preference and process the data.
5. Store the parameters with **wpar** under a new experiment name for general usage.

The reason is that is that **rpar** with two arguments is used in automation.

**INPUT FILES**

*<dir>/data/<user>/nmr/<1D data name>/<expno>/*

*acqu* - acquisition parameters

*<dir>/data/<user>/nmr/<1D data name>/<expno>/pdata/<procno>/*

*proc* - processing parameters

*outd* - output device parameters

*<dir>/data/<user>/nmr/<2D data name>/<expno>/*

*acqu* - F2 acquisition parameters

*acqu2* - F1 acquisition parameters

*<dir>/data/<user>/nmr/<2D data name>/<expno>/pdata/<procno>/*

*proc* - F2 processing parameters

*proc2* - F1 processing parameters

*outd* - output device parameters

*clevels* - 2D contour levels

**OUTPUT FILES**

*<tshome>/exp/stan/nmr/par/user/<1D parameter set>*

*acqu* - acquisition parameters

*proc* - processing parameters

*outd* - output device parameters

*<tshome>/exp/stan/nmr/par/user/<2D parameter set>*

*acqu* - F2 acquisition parameters

*acqu2*- F1 acquisition parameters

*proc* - F2 processing parameters

*proc2* - F1 processing parameters

*outd* - output device parameters

*clevels* - 2D contour levels

3D parameter sets also contain the files *acqu3* and *proc3* for the third direction.

Note that in TopSpin 2.0 and older, the user subdirectory does not exist and user defined parameter sets are stored in:

*<tshome>/exp/stan/nmr/par*

The same location as Bruker parameter sets.

**USAGE IN AU PROGRAMS**

WPAR(name, type)

**SEE ALSO**

*rpar [▶ 304]*, (expinstall)

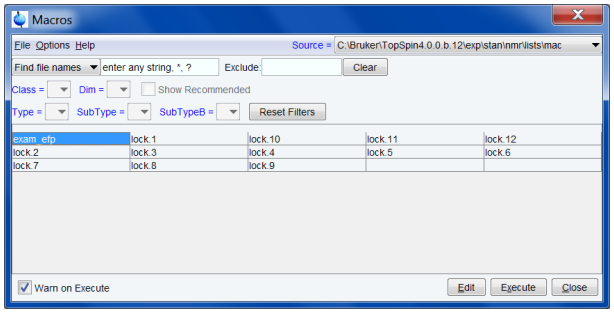
## 10.11 xmac

### NAME

xmac - Execute macro

### DESCRIPTION

The command **xmac** opens a dialog showing all available macros:




Select the desired macro and click **Execute**.

Macros can also be executed from the command line by entering the macro name, e.g.:

**exam_efp**

or

**xmac exam_efp**

The difference is that using the **xmac** command searches for macros only, whereas only entering the name searches for a TopSpin command, AU program, Jython program or macro of that name.

Macros are stored in a database. **xmac** opens the same dialog as the corresponding commands **edmac**. For more details, see the description of this command.

### SEE ALSO

*edpul, edcpde* [▷ 298], *delpul, delcpdd* [▷ 303], *xpy* [▷ 309]

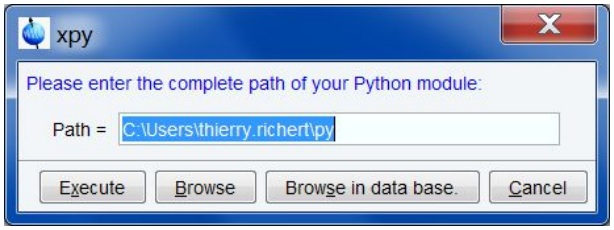
## 10.12 xpy

### NAME

xpy - Execute Jython program

### DESCRIPTION

The command **xpy** opens a dialog where you can select the desired Jython program:

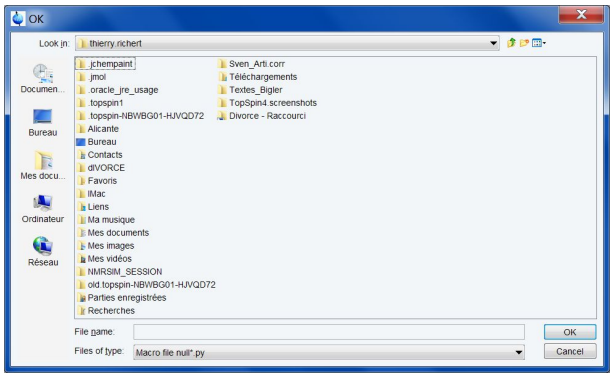



**Path**

Field where you can enter the full path name of the Jython program. Click **Execute** to run it.

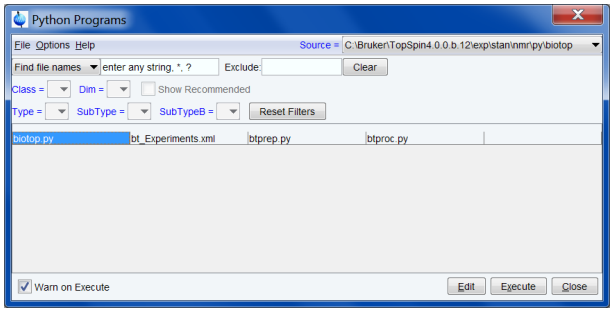
### Browse

Button to open a file browser where you can enter or select the Jython program. Click **Execute** to run it.




### Browse in database

Button to open a dialog showing the available Jython programs in the database:




Select the desired macro and click **Execute**. Jython programs are stored in a database. **xpy** opens the same dialog as the corresponding commands **edpy**. For more details, see the description of this command.

Jython programs can also be executed from the command line by entering the macro name, e.g.:

**ExamCmd4.py**

or

**xpy ExamCmd4.py**

The difference is that using the **xpy** command searches for Jython programs only, whereas only entering just the name searches for a TopSpin command, AU program, Jython program or macro of that name.

### SEE ALSO

*edpul, edcpde* [▶ 298], *delpul, delcpdd* [▶ 303], *xmac* [▶ 309]

## 10.13 xpy3

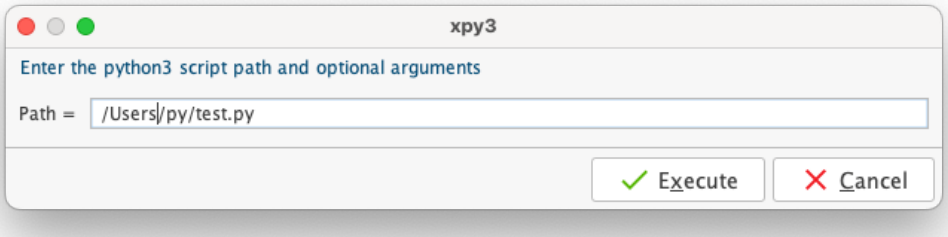
### NAME

xpy3 – Execute Python 3 program

### DESCRIPTION

Python 3 scripts use their own Python environment. (For comparison, often mentioned Jython is a Python 2.7 dialect. Jython scripts run together with TopSpin GUI in a shared Java environment).

The command xpy3 opens a dialog where you can select the desired Python program:




Path is a field where you can enter the full path name of the Python program. Click Execute to run it.

**Parameters, Lists, AU Programs**

Python programs can also be executed from the command line by entering the script name:

e.g.:

*xpy3 ExamCmd4.py*

*xpy3 publish /tmp/mypicture.svg*

**SEE ALSO**

*edpy3* [▸ 298], *edpy* [▸ 298], *xpy* [▸ 309]

# 11 Automation

This chapter describes all TopSpin commands which handle parameters and parameter sets. Furthermore, you will find commands that are used to read or edit lists like pulse programs, gradient programs, frequency lists etc. and, finally, commands which are used to read, edit or run AU programs. Note that several commands in this chapter are acquisition related rather than processing related. Nevertheless they play a role in the processing part of TopSpin.

## 11.1 at

### NAME

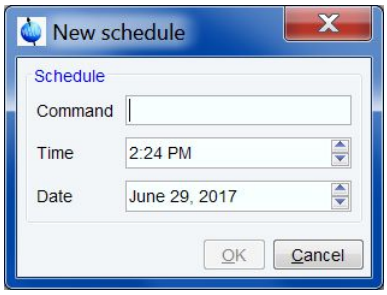
at - schedule a TopSpin command for execution

### SYNTAX

at [HH[:mm]] [DD[.MM[.YY]]] command

### DESCRIPTION

The command **at** performs command scheduling. When entered without arguments, it opens the dialog shown:




Here you can specify the command to be scheduled, e.g. **zg**, and the starting time and date.

The **Time** and **Date** fields are initialized with the current time and date, respectively. By clicking **OK**, the specified is scheduled for execution.

The time and date, as well as the command to be scheduled can also be specified on the command line, using the following syntax:

- **at [HH[:mm]] [DD[.MM[.YY]]] command**

Here are some examples:

- **at 23:30 25.12.07 zg** - will start an acquisition on the 25th of December 2007 at 23.30.
- **at 13 zg** - will start an acquisition today at 13:00.

The command **at** works user specific, i.e. the scheduled command is only executed if TopSpin runs at the specified time and the TopSpin internal user is the user who scheduled the command. For more flexible time definition and user independent scheduling, you can use the command.

Scheduled commands can be viewed in the command spooler, which can be started with the command **spooler** and is available in the spectrometer status bar.

### SEE ALSO

*cron* [▶ 316], *qu* [▶ 323], *qumulti* [▶ 324], *atmulti* [▶ 313], *spooler* [▶ 330]

## 11.2 atmulti

### NAME

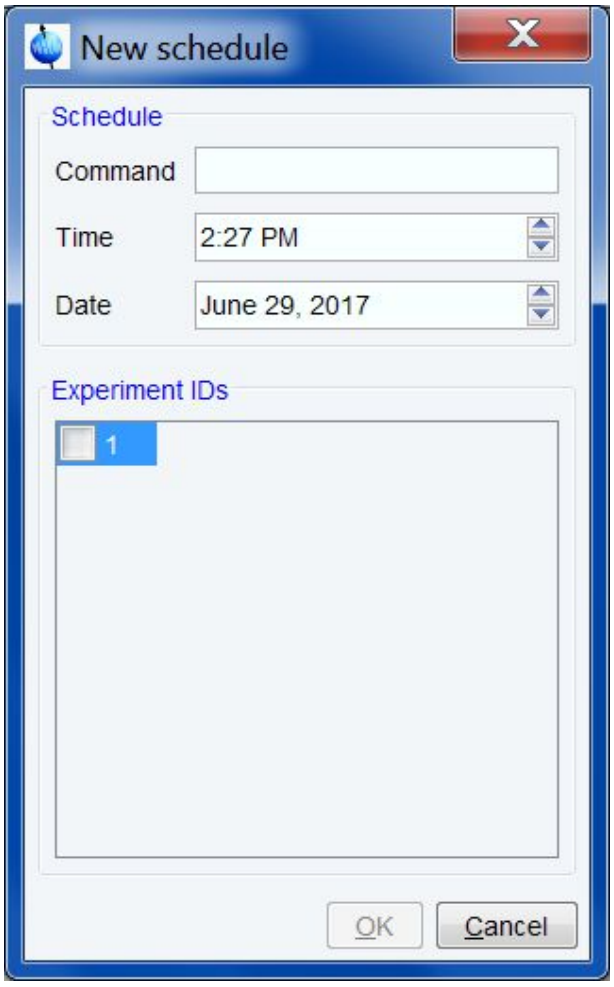
atmulti - schedule a TopSpin command for execution on multiple expnos

### SYNTAX

atmulti [{*|1,2,3|1..7|1-7|1-7,20,21}}]

### DESCRIPTION

The command **atmulti** schedules a command for execution on multiple experiment numbers. It works like **at**, except that it runs on multiple expnos of the current dataset. When entered without arguments, **atmulti** opens the dialog shown:




Here you can enter the command to be executed, specify the time and date of execution and select the target experiments numbers. Clicking **OK** will then schedule the command for execution.

The command **atmulti** takes two arguments, the command to be executed and the target experiment number(s). The dialog will open with the specified arguments preselected. Expnos can be specified in one of the following ways:

**n** : a single experiment number

***** : all expnos under the current data name

**n-m** : expno n through m

**n..m** : equivalent to `n-m`

**n,m** : expno n and m

**n m** : equivalent to n,m

The command to be executed can be specified before or after the expno(s).

Examples of argument strings:

The argument:

**efp 1,3,4-6 8 11** - will preselect the command **efp** and the expnos: 1, 3, 4, 5, 6, 8 and 11

The argument:

**1..8,10 15-20** - will preselect the expnos: 1, 2, 3, 4, 5, 6, 7, 8, 10, 15, 16, 17, 18, 19 and 20

And leave the command field empty.

Specified expnos which do not exist are ignored. The preselected command and expnos can be modified/extended in the dialog.

To select or deselect all expnos in the opened dialog:

- Right-click in the dialog and choose **Select all or Deselect all**, respectively.

On clicking **OK**, a delay job is created for each selected expno, starting with the lowest expno, and sent to the queue.

Scheduled commands can be viewed in the command spooler, which can be started with the command **spooler** and is available in the spectrometer status bar.

Note that if you try to exit TopSpin while a priority job is still active, you will be warned about this and requested to confirm exiting.

### SEE ALSO

*at* [▸ 312], *qu* [▸ 323], *qumulti* [▸ 324], *cron* [▸ 316], *spooler* [▸ 330]

## 11.3 compileall

### NAME

compileall - Compile all Bruker and User AU programs

### DESCRIPTION

The command **compileall** compiles all Bruker and User AU programs. In order to compile Bruker AU programs, these must have been installed. This can be done with the command **expinstall**, with the option "Install Bruker library AU programs" enabled.

For more information on AU programs please refer to the AU reference manual.

### INPUT FILES

*<tshome>/exp/stan/nmr/au/src/**

AU programs (source files)

### OUTPUT FILES

*<tshome>/prog/au/bin/**

AU programs (executable files)

### SEE ALSO

*cplbruk, cpluser* [▸ 315], *edau, xau, delau, xauw* [▸ 317], (xaua, xaup), (expinstall)

## 11.4 cplbruk, cpluser

### NAME

cplbruk - Compile Bruker AU programs

cpluser - Compile user defined AU programs

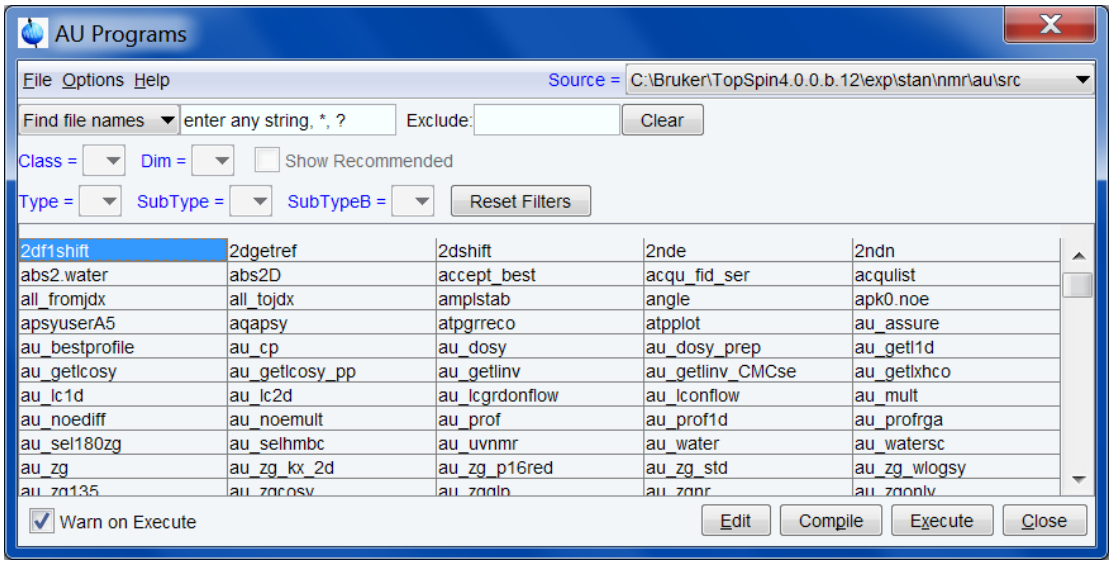
### SYNTAX

cplbruk [<name> | all ]

cpluser [<name> | all ]

### DESCRIPTION

The command **cplbruk** allows to compile one or more Bruker AU programs.




Before you can use it, the command **expinstall** must have been executed once, with the option "Install Bruker library AU programs" enabled. Then you can use **cplbruk** in three different ways:

- **cplbruk <name>** - compile the Bruker AU program <name>
- **cplbruk all** - compile all Bruker AU programs
- **cplbruk** - lists Bruker AU programs; double-click one to compile it

If you specify an argument, then it may contain wildcards; for example:

- **cplbruk a*** compiles all Bruker AU programs which start with *a*.
- **cpluser** works like **cplbruk**, except that it compiles user defined AU programs.

For more information on AU programs please refer to the AU reference manual.

### INPUT FILES

*<tshome>/exp/stan/nmr/au/src/**

AU programs (source files)

### OUTPUT FILES

*<tshome>/prog/au/bin/**

AU programs (executable files)

### SEE ALSO

(expinstall), *compileall* [▷ 314], *edau, xau* [▷ 317], (xaua, xaup)

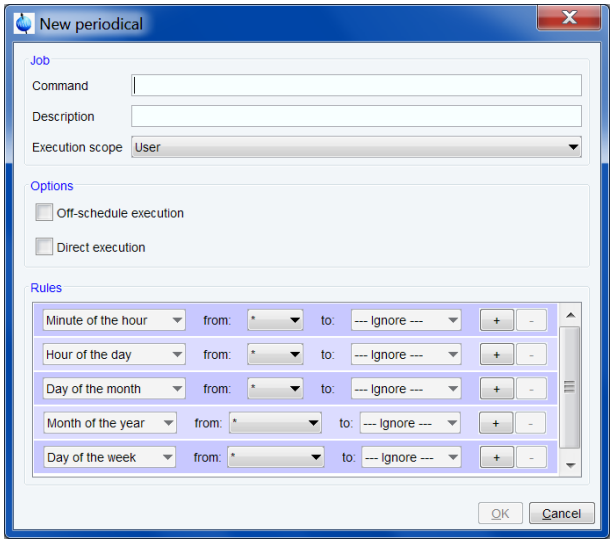
## 11.5 cron

### NAME

cron - schedule a TopSpin command for execution

### DESCRIPTION

The command **cron** performs command scheduling. It allows you to executed commands periodically at predefined times. It is more versatile then the commands **at** and **atmulti** offering full flexibility in time definition, off-schedule execution and user control. When entered without arguments, it opens the dialog shown:




Here you can specify the command to be scheduled, some scheduling options and the starting time and date. The following fields are available:

**Command**

The command to be executed.

**Description**

A description of the command.

**Execution Scope**

The scope of the command execution, *User* or *TopSpin*. For scope *User*, the scheduled command will only be executed if TopSpin is run by the same (internal) user that is active during cron definition. If the scope is TopSpin, the scheduled command will be executed for any (internal) user. Scheduled commands with TopSpin execution scope can only be defined, cancelled or modified after entering the NMR-Administration password.

**Off-schedule execution**

This flag allows you to execute commands that were scheduled to run at the time when TopSpin was not running. These commands are executed after TopSpin startup. Note that commands that were scheduled to run multiple times during TopSpin downtime are only executed once.

**Direct execution**

The option direct execution allows you to run commands directly, i.e. by passing the default queue mechanism. Usually an expired cron job is moved into the priority queue, i.e. the job would wait for any other queued jobs to finish. Setting this flag by passes this mechanism i.e. the job is executed directly when its schedule is due. Please note that however processing commands can be ran in parallel. This is a useful tool to execute for example **nmr_save** and another processing command at the same time.

The following time scheduling rules exist:

**Minute of the hour**: 00 through 59

**Hour of the day**: 00 through 23

**Day of the month**: 00 through 31

**Month of the year**: January through December

**Day of the week**: Sunday through Saturday

For each of these fields, you can define an interval by selecting a value in the **From** and a value in the **To** field. Setting the **To** field to *Ignore*, schedules the command for execution only at the time/date selected in the **From** field. An asterix (*) in the **From** field indicated all possible times. Clicking the **+** button to the right of a field, adds an extra field of the same type, allowing multiple interval definition. Clicking the **-** button removes the extra field.

The cron dialog also offers a right-click menu which allows following options:

- Add new rule - adding new scheduling rules
- Remove rule - removing scheduling rules
- Favorites - define favorites for scheduling rules

**SEE ALSO**

*at* [▶ 312], *atmulti* [▶ 313], *qu* [▶ 323], *qumulti* [▶ 324], *spooler* [▶ 330]

## 11.6 edau, xau, delau, xauw

**NAME**

edau - Edit an AU program

xau - Execute an AU program

delau - Delete an AU program

xauw – Execute an AU program and wait for it to finish

**SYNTAX**

edau [<name>]

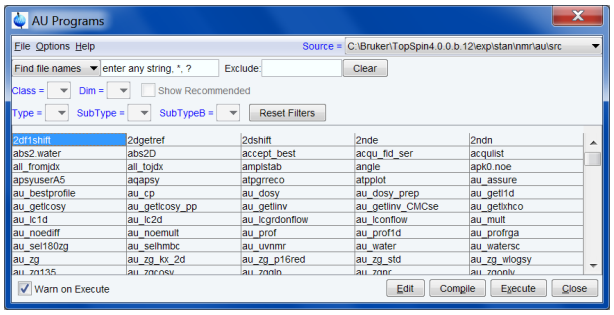
xau [<name>]

delau [<name>]

xauw [<name>]

**DESCRIPTION**

When entered without arguments, the AU program commands **edau**, **xau** and **delau** all open the AU program dialog box:




The dialog offers the following buttons:

**Edit**

Edit the selected AU program. Equivalent to double-clicking the AU program name or entering **edau <name>** on the command line.

**Compile**

Compile the selected AU program. Equivalent to entering **cplbruk <name>** on the command line.

**Execute**

Execute the selected AU program. Equivalent to entering **<name>** or **xau <name>** on the command line.

**Close**

Close the dialog.

The AU programs are selected from the **Source** directory as selected at the upper right of the dialog. Note that:

*<tshome>\exp\stan\nmr\au\src* - contains all Bruker AU programs

*<tshome>\exp\stan\nmr\au\src\user* - contains all user defined AU programs

### The File menu

The **File** menu offers the following functions:

**New...**

Create a new AU program. Note that new AU programs can only be stored in user defined directories.

**Save as...**

Save the selected AU program under a new name. A dialog will appear where you can specify the AU program name and destination directory.

**Delete...**

Delete the selected AU program. Note that both the source and binary AU program are deleted.

**Rename...**

Rename the selected AU program. Note that both the source and binary AU program are deleted.Note that only user defined AU programs can be renamed.

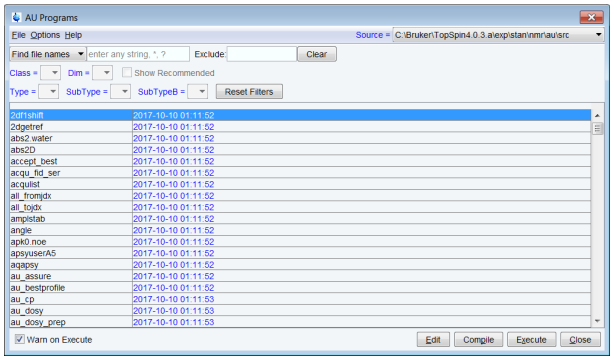
**Export...**

Export the selected AU program to an arbitrary directory. A file dialog will appear where you can select/specify the destination directory.

**Import...**

Import an AU program from an arbitrary directory. A file dialog will appear where you can select/specify the AU program.

### The Options menu




The **Options** menu offers the following functions:

**Show Comment**

Toggles between displaying AU programs with/without comments.

**Show Date**

Toggles between displaying AU programs with/without date.

**Sort by Date**

Sort AU programs by date when selected.

**Manage Source Directories**

Add/modify AU programs source directories. AU programs will be searched for in the order of the directories specified.

Detailed information about *Manage Source Directories* is described in Chapter User specific handling of Source Directories.

**Export Sources...**

Opens a dialog to export an entire AU program library to a user defined directory. Note the difference to the *Export* function under the *File* menu (see below).

When you edit a Bruker AU program, it is shown in view mode which means it cannot be modified. However, if you click **Save as..** and store it under a different name, the stored file is automatically opened in edit mode. When you edit a User defined AU program, it is opened in edit mode and can be modified.

When **edau** is entered on the command line with an argument, the corresponding AU program will be opened. If it does not exist it will be created. If the argument contains wildcards, the AU dialog box is opened showing the matching AU programs. For example, **edau a*** displays all AU programs which start with *a*.

Bruker AU programs must be installed once with **expinstall** before they can be opened with **edau**. The installation must be repeated when a new version of TopSpin is installed.

**edau** uses the editor which is defined in the TopSpin User Preferences. To change it, enter **set**, click **Miscellaneous** and select or change the editor.

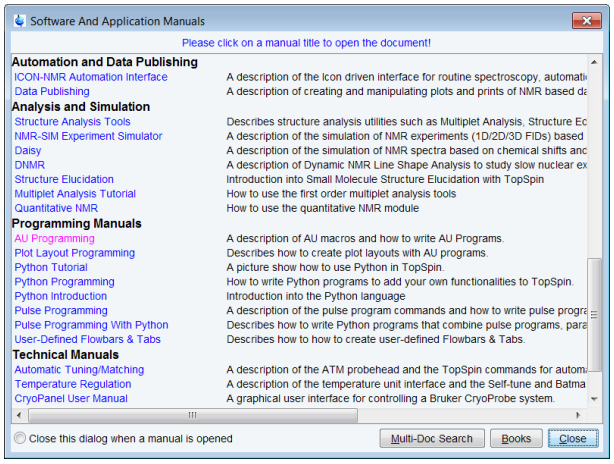
AU programs are usually executed simply by entering their names. The command **xau** is only needed in three cases:

- The AU program has not been compiled yet.
- A TopSpin command with the same name exists.
- To call an Au program from another AU program (using the macro XAU).

AU programs run in background and several of them can run simultaneously. The command **kill** can be used to stop a running (or hanging) AU program.

For details on writing, compiling, and executing AU programs please refer to the AU reference manual:

In the menu click **Help** | **Manuals** | **Programming Manuals** | **AU programming**




### INPUT/OUTPUT FILES

*<tshome>/exp/stan/nmr/au/src/**

AU program source files.

*<tshome>/prog/au/bin/**

AU program executable binary files

### SEE ALSO

*cplbruk, cpluser* [▸ 315], *compileall* [▸ 314], (expinstall)

## 11.7 intser

### NAME

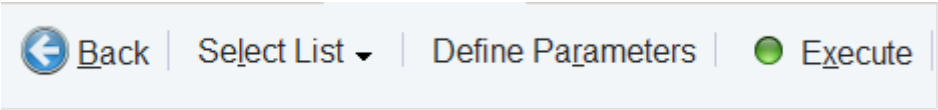
intser - integrate a list of spectra (1D, 2D)

### DESCRIPTION

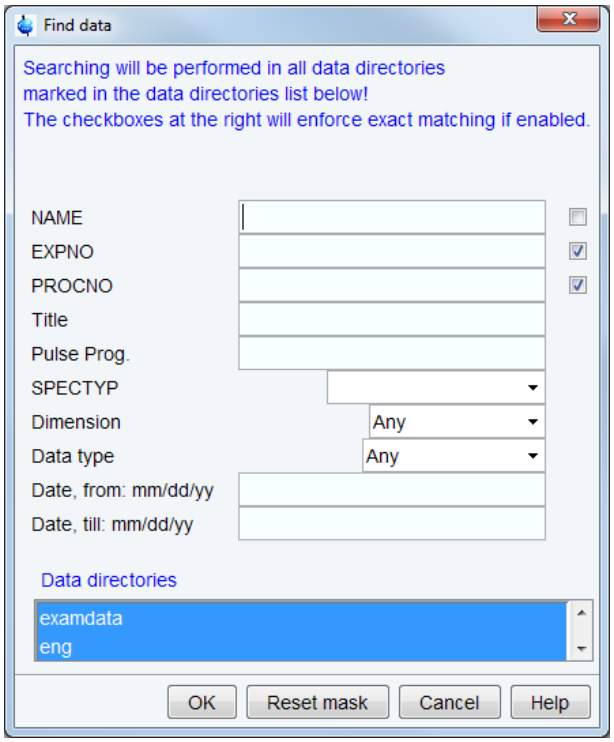
The command **intser** integrates a series of 1D or 2D data.

- Click **Process | Advanced | Integrate Spectra List**.

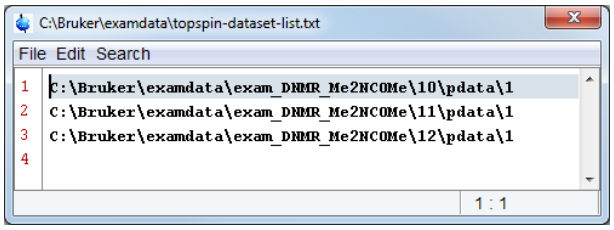
This will open the following workflow button bar:



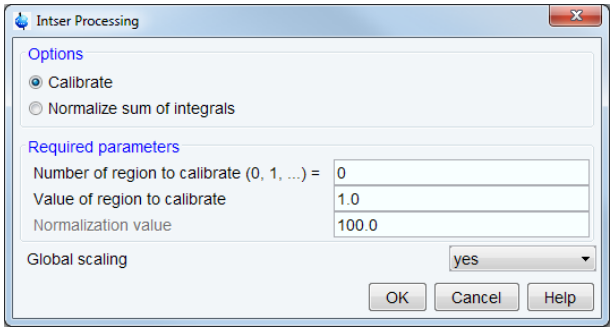

Click **Select List** to define the list of data sets on which you want to perform the series of integrations. This list must have been previously created manually or can be created by clicking on the arrow key on the **Select List** button and selecting the command **Build dataset list using find**. The latter will open a dialogue window as shown below.




Enter appropriate values for the various list items to find the data sets you want to work with. A completed list may look like the one shown below. Click on **Define List** button and select **Edit Dataset List**.




The first data set in the list serves as reference data set. Its PROCNO directory must contain an **intrng** file with the spectral regions to be integrated. This file is created by automatic integration (command **abs**) or by interactive integration (command **.int**). The next step is to set up the parameters for the serial integration. Clicking on **Define Parameters** will open the following dialogue box.



There are two options:

1. Calibrate the integrals in the series of spectra to a certain reference value. In the first (reference) spectrum, the indicated **Number of region to calibrate** is calibrated to **the Value of region to calibrate**. All integrals in the series of experiments will then be scaled with the same scaling factor. This allows to immediately compare the integrals within the series of experiments.
2. Normalize the sum of integrals. Works like the calibration, but instead of scaling the reference region to a certain value, the sum of all integrals in the reference spectrum is scaled to the **Normalization value**. All integrals in the series of experiments will then be scaled with the same scaling factor. This allows to immediately compare the integrals within the series of experiments.

**Global scaling**

Takes the value **yes** or **no**. For **yes**, all integrals of all spectra in the list will be scaled relative to the normalization region of the reference spectrum. For **no**, all integrals of one spectrum will be scaled relative to the normalization region of the same spectrum. The normalization region number and value are same for each spectrum (the specified values).

To start the calculation, click on **Execute**.

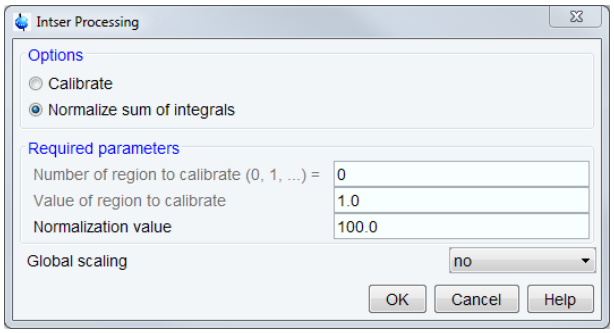
The integration result is stored in a text file whose contents are shown on the screen. Its format is demonstrated by the following example. Lines beginning with a # are comment lines. The format is suitable to be imported into a spreadsheet program such as Excel for further processing. The example is the result of integrating the 3 defined regions of 3 data sets. The first region is the reference region and all integrals in all spectra were integrated with the same scaling factor.

```
# Result of 'intser'
# Date/time = Wed Feb 21 11:42:55 CET 2018
# Data set list (full path) = C:\Bruker\examdata\topspin-dataset-list.txt
# Region to calibrate = 0
# Value of region to calibrate = 1.0
# Global scaling = yes

# --- Integral info ---
# A 1.0 #regions in PPM
# # low field high field bias slope
# 2.999042477377031 2.9053223999589988 -0.0 -0.0 # for region 1
# 2.824990905029257 2.747337126597173 -0.0 -0.0 # for region 2
# 2.01899823923418 1.895823280341909 -0.0 -0.0 # for region 3

# Spectrum#; Integral 0; Integral 1; Integral 2;
0;1.0;0.9944740153680266;1.012183456123523;
1;0.774737126457184;0.7625343796353649;0.7993500292763215;
2;0.6126474881645066;0.4877583034349917;0.6854909593010602;
```

With the parameters set as below the result of the integration will look like this.




```
# Result of 'intser'
# Date/time = Wed Feb 21 11:52:40 CET 2018
# Data set list (full path) = C:\Bruker\examdata\topspin-dataset-list.txt
# Normalization value = 100.0
# Global scaling = no

# --- Integral info ---
# A 1.0 #regions in PPM
# # low field high field bias slope
# 2.999042477377031 2.9053223999589988 -0.0 -0.0 # for region 1
# 2.824990905029257 2.747337126597173 -0.0 -0.0 # for region 2
# 2.01899823923418 1.895823280341909 -0.0 -0.0 # for region 3

# Spectrum#; Integral 0; Integral 1; Integral 2;
0;33.259525219675844;33.07573359444518;33.664741185878974;
1;33.15629487831799;32.63405596897349;34.20964915270852;
2;34.30475406014218;27.311674271708828;38.383571668148996;
```

Particularly, in this example, the last three lines with the integration results are important.

The command **intser** can also be used to integrate a series of 2D data. Note that in this case the file containing the integral regions is **int2drng**.

### SEE ALSO

*serial [▶ 327]*

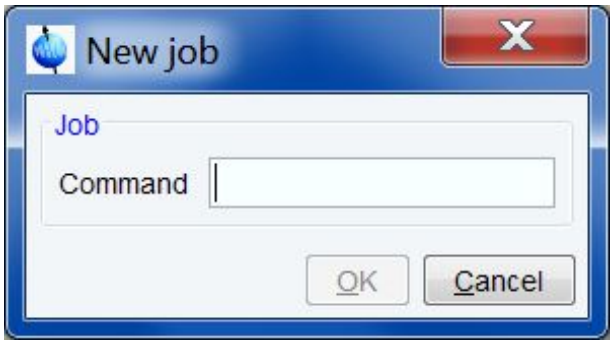
## 11.8 qu

### NAME

qu - queue a TopSpin command for execution

### DESCRIPTION

The command **qu** queues a command for execution. It requires one argument, the command to be queued.




For example, the command:

**qu xfb**

Queues the command **xfb** for execution. This means that **xfb** is executed as soon as the currently running command and previously queued commands have finished.

Command queuing can, for example be used, to process a 2D data set immediately after acquisition. This is done with the command sequence:

**zg**

**qu xfb**

Acquisition commands like **zg**, **go**, **rga** and **atma** are automatically queued, if *auto-spooling* is enabled in the User Preferences (command **set**).

Queued commands can be viewed in the command spooler, which can be started with the command **spooler** and is available in the spectrometer status bar.

### SEE ALSO

*cron* [▷ 316], *at* [▷ 312], *atmulti* [▷ 313], *qumulti* [▷ 324], *spooler* [▷ 330]

## 11.9 qumulti

### NAME

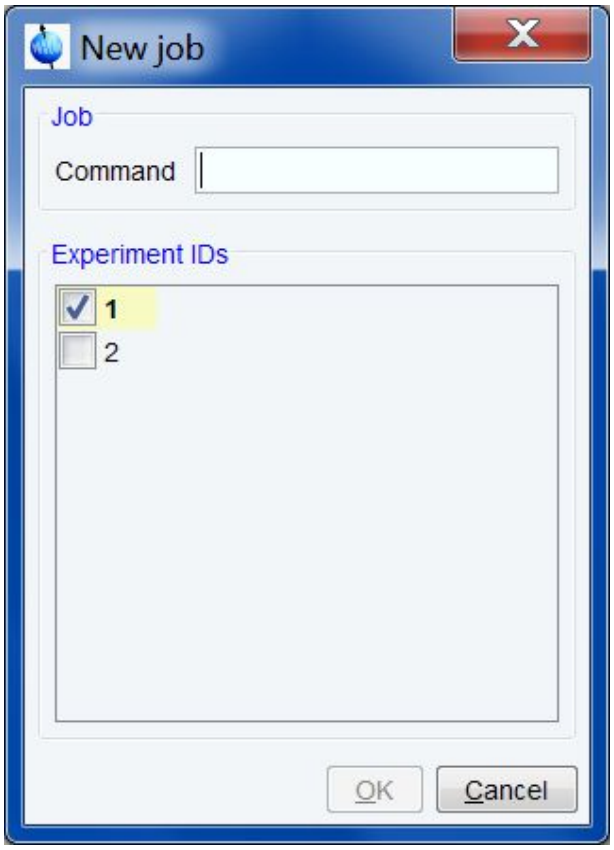
qumulti - queue a TopSpin command for execution on multiple expnos

### SYNTAX

qumulti [{*|1,2,3|1..7|1-7|1-7,20,21}}]

### DESCRIPTION

The command **qumulti** queues a command for execution on multiple expnos of the current dataset. When entered without arguments, **qumulti** opens the dialog shown:




Here you can enter the command to be executed and select the experiments numbers on which the specified command should work. The dialog shows all available expnos, with the active data set selected.

Clicking **OK** queues the command for execution.

The command **qumulti** takes two arguments, the command to be executed and the target experiment number(s). The dialog will open with the specified arguments pselected. Expnos can be specified in one of the following ways:

**n** : a single experiment number

***** : all expnos under the current data name

**n-m** : expno n through m

**n..m** : equivalent to `n-m`

**n,m** : expno n and m

**n m** : equivalent to n,m

The command to be executed can be specified before or after the expno(s).

Examples of argument strings:

The argument **efp 1,3,4-6 8 11** will preselect the command **efp** and the expnos: 1, 3, 4, 5, 6, 8 and 11.

The argument:

**1..8,10 15-20** will preselect the expnos: 1, 2, 3, 4, 5, 6, 7, 8, 10, 15, 16, 17, 18, 19 and 20, and leave the command field empty.

Specified expnos which do not exist are ignored. The preselected command and expnos can be modified/extended in the dialog.

To select or deselect all expnos in the opened dialog:

Right-click in the dialog and choose **Select all** or **Deselect all**, respectively.

If **qumulti** is entered without argument, only the current expno is preselected.

On clicking **OK**, a priority job is created for each selected expno, starting with the lowest expno, and sent to the queue.

Queued commands can be viewed in the command spooler, which can be started with the command **spooler** and is available in the spectrometer status bar.

Note that if you try to exit TopSpin while a priority job is still active, you will be warned about this and requested to confirm exiting.

### SEE ALSO

*cron* [▷ 316], *qu* [▷ 323], *at* [▷ 312], *atmulti* [▷ 313], *spooler* [▷ 330]

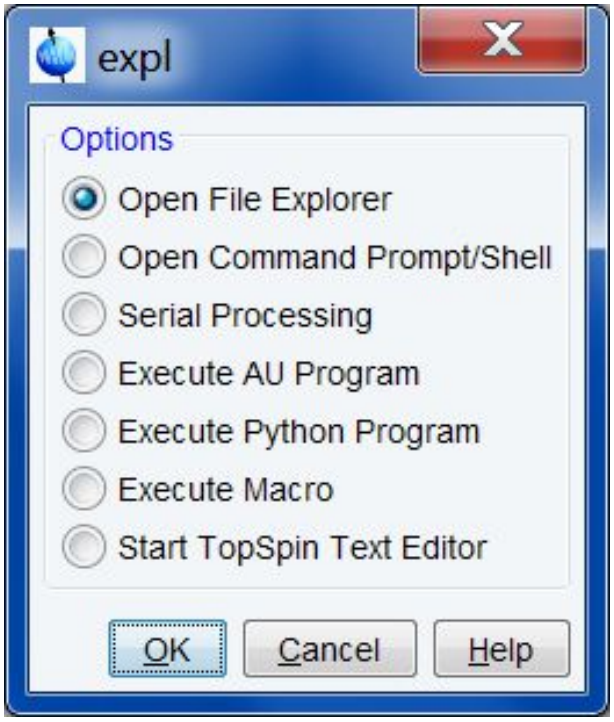
## 11.10 run

### NAME

run - Open dialog for starting macro, AU, Jython or serial.

### DESCRIPTION

The command **run** opens the run dialog window:




This dialog box has various options, each of which selects a certain command for execution.

#### Open the file explorer

This option selects the command **expl** for execution. It opens the File Explorer showing the processed data files (the files in the *procno* directory) of the active data set. Under Linux the KDE konqueror will be opened. If no data set is open in the TopSpin data area, the Explorer will show the users home directory. **expl** allows you access to the current data files as well as the entire data directory tree.

An alternative way to access data files is to right-click inside the data window and select *Files* in the appearing popup menu.

### Open Command Prompt/Shell

This option selects the command **shell** for execution. It opens a Windows Command Prompt or Linux Shell, depending on your operating system.

### Serial Processing

This option selects the command **serial** for execution. It opens a dialog window where you can set up and start data processing of a series of data sets using TopSpin commands, macros or Jython programs.

### Execute an AU program

This option selects the command **xau** for execution. It opens the AU dialog box showing a list of available AU program. Here you can select an AU program and click **Execute** to execute it. **xau** can also be entered on the command line in which case you can specify the AU program as an argument.

### Execute a Jython program

This option selects the command **xpy** for execution. It prompts you for the path name of a Jython program. Enter this path name and click **OK** to execute the Jython program.

### Execute a Macro

This option selects the command **xmac** for execution. It opens the Macro dialog box showing a list of available macros. Here you can select macro and click **Execute** to execute it. **xmac** can also be entered on the command line in which case you can specify the macro as an argument.

### Open a text editor

This option selects the command **edtext** for execution. It opens an empty text file with the TopSpin editor. The file can be stored in any directory.

### SEE ALSO

*expl* [▷ 363], *shell* [▷ 373], *edau, xau* [▷ 317], *xpy* [▷ 309], *xmac* [▷ 309], *edtext* [▷ 361]

## 11.11 serial

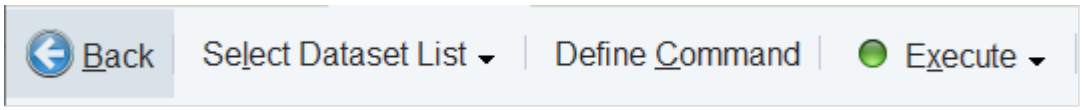
### NAME

serial - Serial processing with macro or Jython script

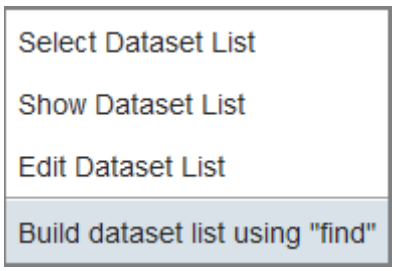
### DESCRIPTION

To start serial, click **Process | Advanced | Process Dataset List**.

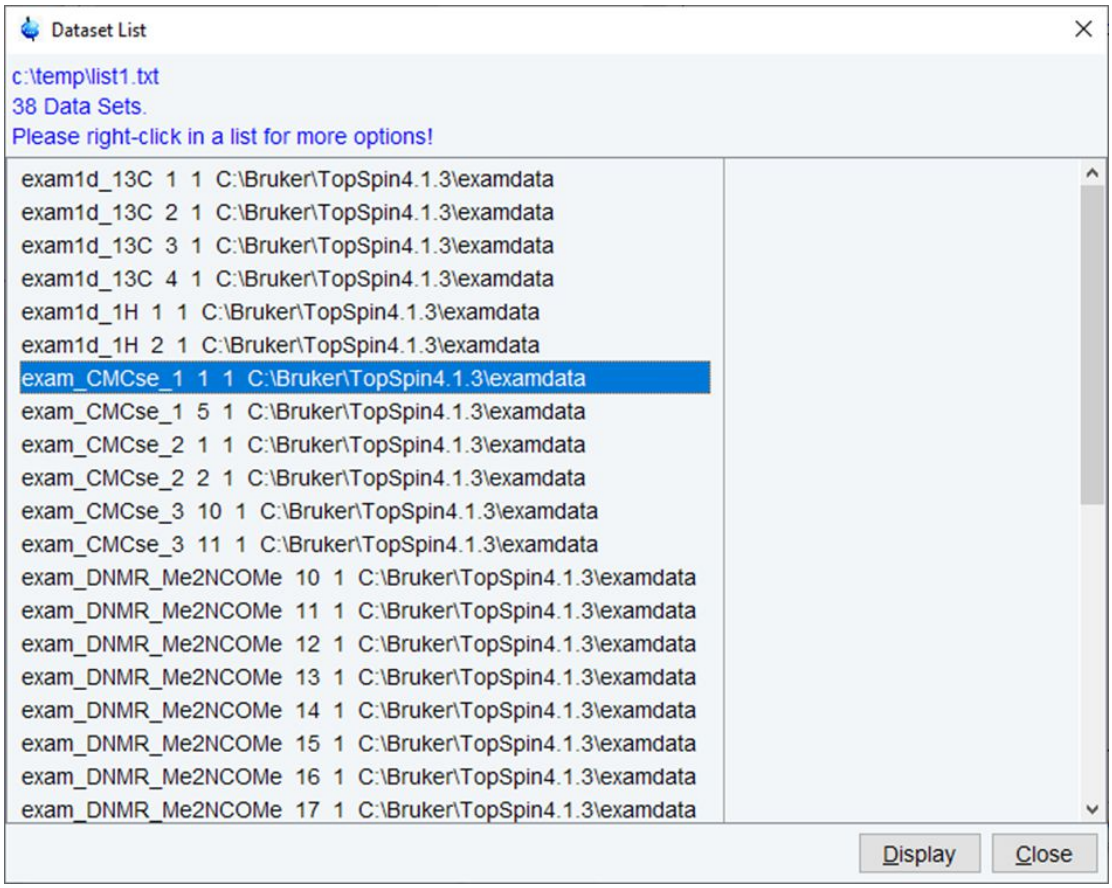
This will open the following workflow button bar.



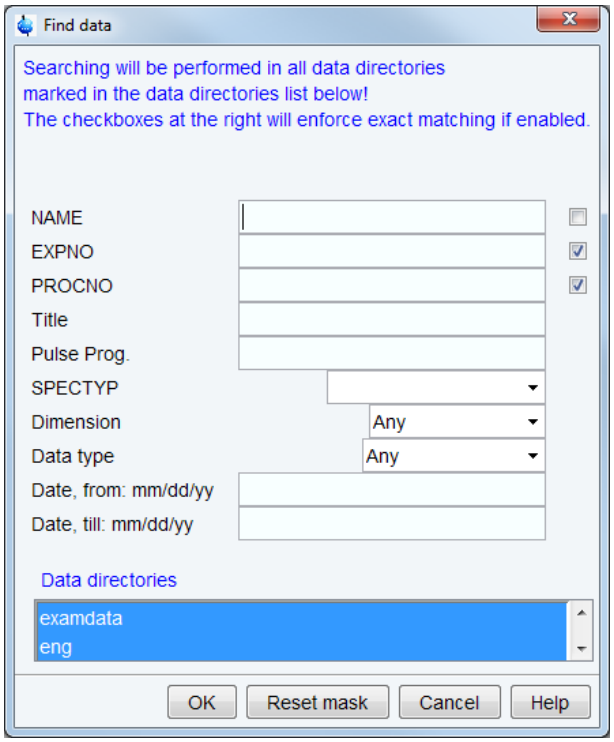
Click **Select Dataset List** to choose the list of data sets on which you want to execute the series of commands. This list must have been previously created manually or can be created by clicking on the arrow key on the **Select Dataset List** and selecting the command **Build dataset list using find**.



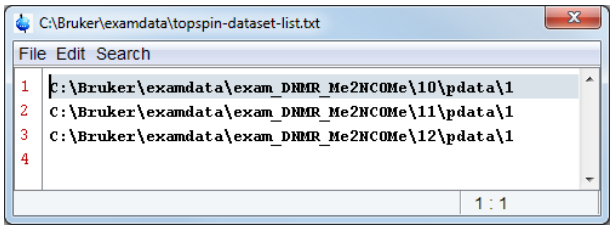

The data set lists may be now reopened from the data list menu (**Show Dataset List**). The list is shown in the same window as a result of the data search.



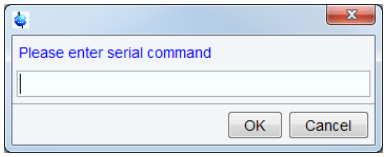

The **find** command will open a dialogue window as shown below.




Enter appropriate values for the various list items to find the data sets you want to work with. A completed list may look like the one shown below. Click **Define List** and select **Edit Dataset List**.




The next step is to set up the commands for the serial command execution. Clicking **Define Command** will open the following dialogue box.




Enter TopSpin commands, macros, AU programs or Jython scripts here. If you want to execute several commands, they must be separated with a semicolon. Examples are:

**efp**

**xmac <your macro name>**

**xpy <your jython program>**

**em; ft; apk; abs**

Note that Jython programs are much more versatile than macros. Details on Jython programming can be found under:

**Help | Manuals | Programming Manuals | Python programming**

Note that serial processing can also be started as follows:

- Click **File | Run A Program**, then select **Serial Processing** and click **OK**.

### INPUT/OUTPUT FILES

*<tshome>/exp/stan/nmr/py*

*<tshome>/exp/stan/nmr/py/user*

*ser_*.py* - Jython programs for serial processing

*<tshome>/exp/stan/nmr/lists/mac/*

*<tshome>/exp/stan/nmr/lists/mac/user*

*ser_** - Macros for serial processing

### SEE ALSO

*edpul, edcpde* [▶ 298], *intser* [▶ 320]

## 11.12 spooler

### NAME

spooler - display queued, scheduled and cron jobs.

### DESCRIPTION

The command **spooler** displays the spooler jobs. It opens a dialog showing:

- Queued jobs (jobs started with the command **qu** or **qumulti**).
- Scheduled jobs (jobs started with the command **at** or **atmulti**).
- Cron jobs (jobs started with the command **cron**).

For each job the dialog shows the command to be executed, the target data object, the owner and, depending on the job's various other information.

The Spooler dialog offer the following menus:

**Spooler**

Allows you to suspend or remove all queued, scheduled or cron jobs.

**Queue**

Allows you to:

- Create new jobs.
- Suspend all jobs.
- Remove all jobs.

For priority, delayed and cron jobs, separately.

**Job**

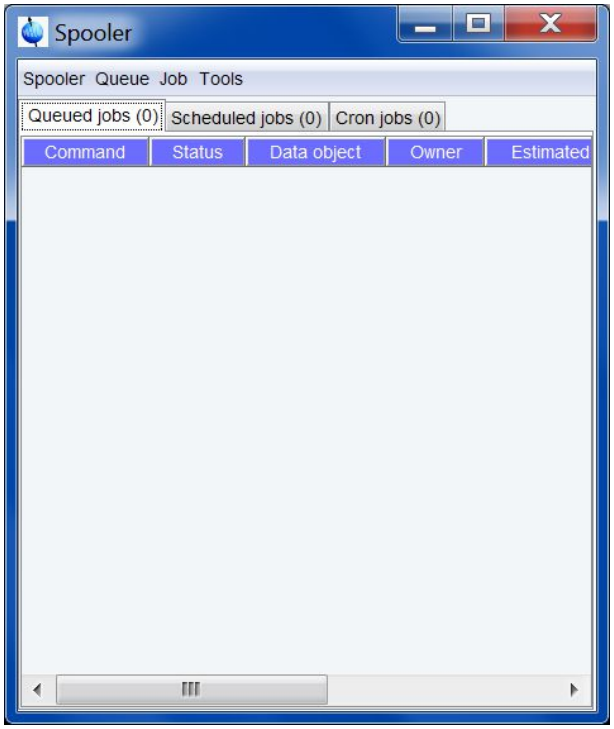
Allows you to:

- Create new jobs.
- Stop, restart or delete selected jobs.
- Open the job properties dialog from here (also available by double click on the job entry).

For the selected job type.

**Tools**

Allows you show the spooler log file and spooler report.





**Spooler Report**

To show the spooler report:

Click **Tools | Show spooler report**

To delete entries from the spooler report:

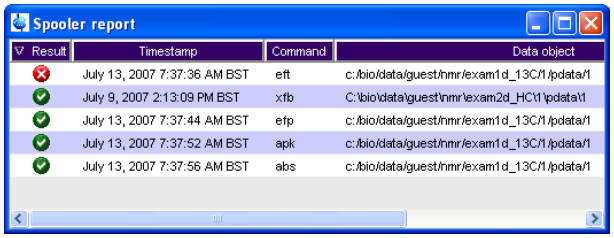
1. Mark the entries to be deleted.
2. Right-click in the dialog and select **Delete**.

To open datasets from the spooler report:

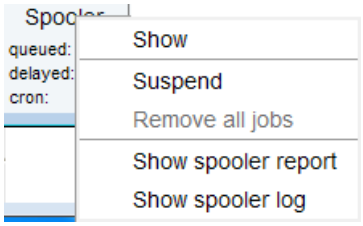
Double-click the respective entry

or

Right-click the respective entry and select **Display**.



Note that the spooler report can also be opened from Spooler field (if enabled) in the Acquisition Status Bar by right-clicking the word **Spooler** and selecting **Show spooler report**.




## INPUT/OUTPUT FILES

*<tshome>/conf/globals*

*spoolerprotocol.xml* - system spooler report

*<userhome>/.topspin-<hostname>/prop/*

*spoolerprotocol.xml* - user spooler report

## SEE ALSO

*cron* [▷ 316], *qu* [▷ 323], *qumulti* [▷ 324], *at* [▷ 312], *atmulti* [▷ 313]

# 12 Conversion Commands

This chapter describes all TopSpin conversion commands. These are commands which convert one data format to another. Described are the conversion of Bruker Aspect 2000/3000, WINNMR, Varian, Jeol and Felix data to TopSpin. Furthermore, the conversion to and from JCAMP-DX, ZIP and TXT format.

## 12.1 conv

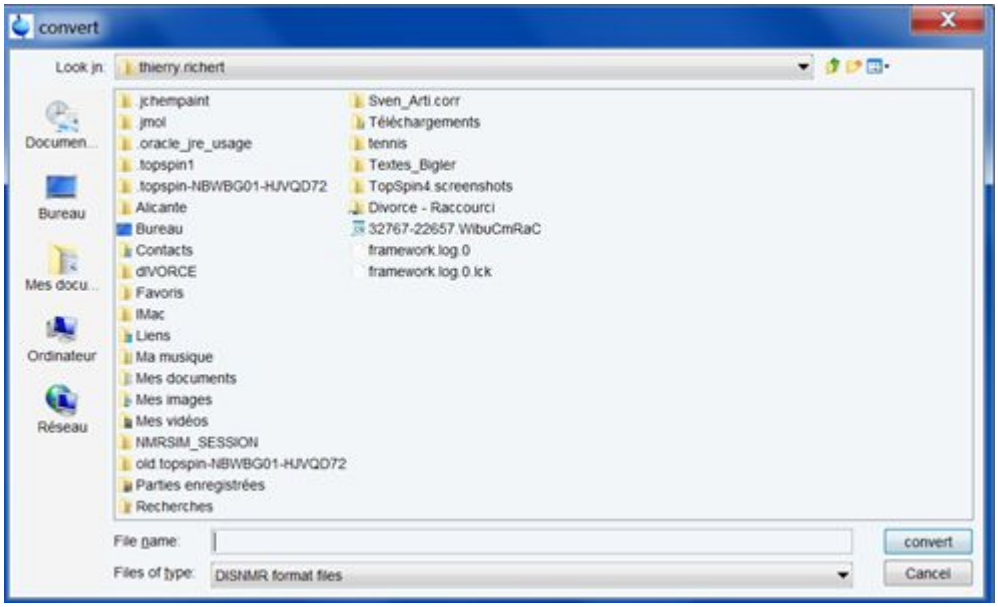
### NAME

conv - Convert Aspect 2000/3000 data to TopSpin format (1D, 2D, 3D)

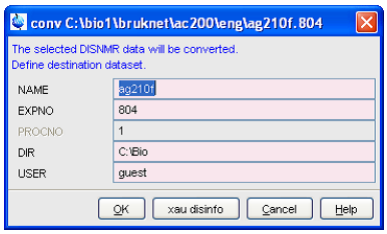
### DESCRIPTION

The command **conv** converts DISNMR/DISMSL data (data from an Aspect 2000/3000) to the TopSpin format. It opens a file browser where you can:

1. Navigate to the input directory where the DISNMR/DISMSL data reside.
2. Select the data file to be converted and click **convert**.




3. In the next dialog box specify the output TopSpin data set. Note that the data path variables are initialized as follows:
   - NAME is the file name of the DISNMR input data
   - EXPNO is the extension of the DISNMR input data set. If the extension is not numeric or if it is missing, EXPNO is initialized with 1.
   - PROCNO is set to 1 and cannot be changed.
   - DIR is the <DIR> value of the current TopSpin data path.
   - USER is the <USER> value of the current TopSpin data path.




The command **conv** executes the AU program **disconv**. This means the command **expinstall** must have been executed once, installing the Bruker AU programs, before you can use **conv**.

The dialog box shown above shows the button **xau disinfo**. Clicking this button executes the corresponding AU program showing the relevant data set parameters.

### INPUT FILES

*<input directory>/** - A2000/3000 data

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - Avance type 1D raw data

*ser* - Avance type 2D or 3D raw data

*acqu* - acquisition parameters

*acqus* - acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*1r*, *1i* - converted processed 1D data

*2rr*, *2ir*, *2ri*, *2ii* - converted processed 2D data

*proc* - processing parameters

*procs* - processing status parameters

For 2D data, the additional parameter files *acqu2*, *acqu2s*, *proc2* and *proc2s* will be created. For 3D data, the additional parameter files *acqu2*, *acqu2s*, *proc2* and *proc2s* and *acqu3*, *acqu3s*, *proc3* and *proc3s* will be created.

### SEE ALSO

*winconv* [▶ 353], *convdta* [▶ 334], *vconv* [▶ 350], *jconv* [▶ 342], *fconv* [▶ 336]

## 12.2 convdta

### NAME

convdta - Convert Avance type raw data to AMX type (1D, 2D, 3D)

### DESCRIPTION

The command **convdta** converts Avance type raw data to AMX type raw data. It can handle 1D, 2D and 3D data. This is useful if you want to process data that have been acquired on an Avance spectrometer on an AMX or ARX spectrometer.


**convdta** takes up to six arguments and can be used as follows:

1. **convdta**
2. You will be prompted for an *expno* under which the raw data must be stored.
3. **convdta <expno>**
4. The raw data will be stored under the specified *expno*.
5. **convdta <expno> <name> y**
6. The output will be stored under the specified *name* and *expno*. The last argument (*y*) causes **convdta** to overwrite existing data without a warning.
7. **convdta <expno> <name> <user> <dir> y n**
8. The output will be stored under the specified *expno*, *name*, *user* and *dir*. The second last argument (*y*) causes **convdta** to overwrite existing data without a warning. The last argument (*n*) causes the display to remain on the current data set rather than change to the output data set.

You can use any other combination of arguments as long they are entered in the correct order. The processed data number (*procno*) of the new data set cannot be chosen, it is always set to 1.

**INPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - Avance type 1D raw data

*ser* - Avance type 2D or 3D raw data

*acqu* - acquisition parameters

*acqus* - acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*proc* - processing parameters

*procs* - processing status parameters

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - AMX type 1D raw data

*ser* - AMX type 2D or 3D raw data

*acqu* - acquisition parameters

*acqus* - acquisition status parameters

*audita.txt* - acquisition audit trail

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*proc* - processing parameters

*procs* - processing status parameters

For 2D data, the additional parameter files *acqu2*, *acqu2s*, *proc2* and *proc2s* will be used. For 3D data, the additional parameter files *acqu2*, *acqu2s*, *proc2* and *proc2s* and *acqu3*, *acqu3s*, *proc3* and *proc3s* will be used.

#### USAGE IN AU PROGRAMS

CONVDTA(expno)

#### SEE ALSO

*conv* [▷ 333], *fconv* [▷ 336], *jconv* [▷ 342], *vconv* [▷ 350]

## 12.3 convertpeaklist

#### NAME

convertpeaklist - Convert XML-format peak list to TXT-format peak list

#### DESCRIPTION

The command **convertpeaklist** converts an XML-format peak list to various other formats. The output format can be controlled with the argument:

**txt** - text format, file *peak.txt*

**peaklist** - Mixed Shape Deconvolution format, file *peaklist*

**ml** - AUREMOL format, file *1r.ml* (1D), *masterlist.ml* (2D)

**peaks** - XEASY format, file xeasy.peaks)

#### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*peaklist.xml* - peak list for the Plot Editor in XML format

#### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*peak.txt* - peak list for the Plot Editor in TXT format

#### SEE ALSO

*gdcon, ldcon, mdcon, ppp, dconpl, dcon* [▷ 215] , *pps, ppf, ppl, pph, ppj, pp* [▷ 228]

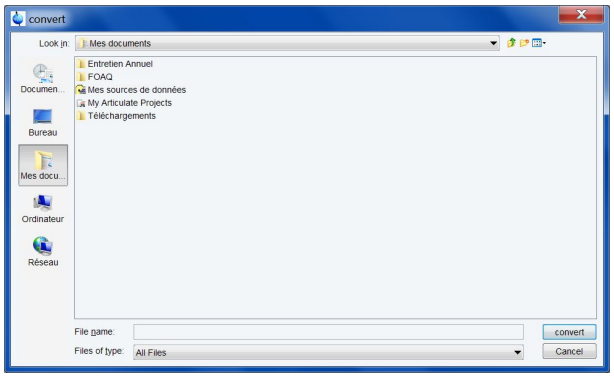
## 12.4 fconv

#### NAME

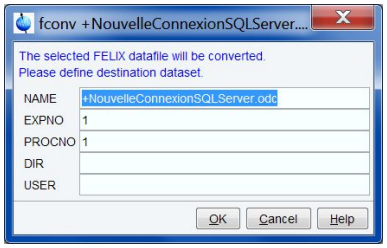
fconv - Convert Felix type data to Bruker TopSpin type data (1D)

#### DESCRIPTION

The command **fconv** converts Felix data to TopSpin format. It opens a dialog window where you can navigate to the Felix input data file. Just select the desired file and click **convert**.




This will open the dialog box shown:




Here you can specify the TopSpin destination data set and click **OK** to start the conversion.

The **fconv** source and destination data can also be entered on the command line. Here are some examples:

**fconv <path>/fdata**

When the specified input data are found, the dialog window shown above will appear. Here, you can specify the output data set.

**vconv fdata <name> <expno> <dir> <user>**

Here, the destination data set is specified as command line arguments. The *procno* is automatically set to 1. If the data set specification is incomplete, the dialog window shown above will appear.

**fconv** can convert raw and processed Felix data.

Note that **fconv** converts 1D data only.

## INPUT FILES

*<fdata_name>* - Felix data file

## OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - TopSpin 1D raw data

*acqu* - TopSpin acquisition parameters

*acqus* - TopSpin acquisition status parameters

*audita.txt* - acquisition audit trail

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/1/*

*proc* - TopSpin processing parameters

*procs* - TopSpin processing status parameters

### SEE ALSO

*vconv* [▸ 350], *jconv* [▸ 342], *conv* [▸ 333], *winconv* [▸ 353], *convdta* [▸ 334]

## 12.5 fromjdx

### NAME

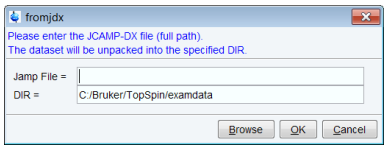
fromjdx - Convert a JCAMP-DX data file to TopSpin format (1D, 2D)

### SYNTAX

fromjdx [<pathname> [<path-variable>] [y]]

### DESCRIPTION

The command **fromdjx** converts a JCAMP-DX data file to a TopSpin data set. JCAMP-DX is a standard ascii exchange format for spectroscopic data.




- **fromdjx** supports the conversion of 1D data (raw or processed) and 2D data (raw or processed-real).
- **fromjdx** takes up to three arguments and can be used as follows:
  - **fromjdx**
  - prompts for the path name of the JCAMP-DX input file, converts it and stores it under the lowest empty *expno* and *procno* 1.
  - **fromjdx <pathname>**
  - converts the JCAMP-DX file specified by the path name and stores it under the lowest empty *expno* and *procno* 1.
  - **fromjdx <pathname> y**
  - converts the JCAMP-DX file specified by the path name and stores it under *expno* 1 and *procno* 1. Possibly existing data are overwritten (y).

In the examples above, **fromjdx** stores the output data set in the directory:

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

Where

*<dir>* - the data directory of the current data set

*<user>* - the user of the currently current data set

*<name>* - the name of the JCAMP-DX file but without the extension *.dx*

Further examples:

- **fromjdx <pathname> du**
- Converts the JCAMP-DX file specified by the path name and stores it under the *dir* (=*du*), *user*, *name*, *expno* and procno as specified in the input JCAMP-DX file.
- **fromjdx <pathname> user**
- Converts the JCAMP-DX file specified by the path name and stores it under the *dir* of the current data set and the *user*, *name*, *expno* and procno as specified in the input JCAMP-DX file.
- **fromjdx <pathname> name**

- Converts the JCAMP-DX file specified by the path name and stores it under the *dir* and user of the active data set and the name, *expno* and procno as specified in the input JCAMP-DX file.
- **fromjdx <pathname> expno**
- Converts the JCAMP-DX file specified by the path name and stores it under the *dir*, *user* and *name* of the active data set and the expno and procno as specified in the input JCAMP-DX file.
- **fromjdx <pathname> procno**
- Converts the JCAMP-DX file specified by the path name and stores it under the *dir*, *user* and *name* of the active data set, *expno* 1 and the procno as specified in the input JCAMP-DX file.

All the above examples can be used with the **y** option to overwrite possibly existing data.

**INPUT FILES**

<path name>/<mydata.dx> - TopSpin data in JCAMP-DX format

**OUTPUT FILES**

**For 1D and 2D data:**

*<tshome>/prog/curdir/<user>/*

*curdat* - current data definition

*<dir>/data/<user>/nmr/<name>/<expno>/*

*audita.txt* - acquisition audit trail (if input file contains raw data)

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*auditp.txt* - processing audit trail (if input file contains processed data)

*outd* - output device parameters

*title* - title file (see **edti**)

**For 1D data:**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - 1D raw data (if input file contains 1D raw data)

*acqu* - acquisition parameters

*acqus* - acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*1r* - real processed 1D data (if input file contains 1D real processed data)

*1i* - imaginary processed 1D data (if input file contains 1D imaginary data)

*proc* - processing parameters

*procs* - processing status parameters

**For 2D data:**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*ser* - 2D raw data (input if Output Data = raw)

*acqu* - F2 acquisition parameters

*acqu2* - F1 acquisition parameters

*acqus* - F2 acquisition status parameters

*acqu2s* - F1 acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*2rr* - real processed 2D data (if input file contains 2D real processed data)

*proc* - F2 processing parameters

*proc2* - F1 processing parameters

*procs* - F2 processing status parameters

*proc2s* - F1 processing status parameters

*clevels* - 2D contour levels

### USAGE IN AU PROGRAMS

FROMJDX(name)

For example FROMJDX("/tmp/mydata.dx")

### SEE ALSO

*tojdx* [▶ 344], *totxt* [▶ 346], *tozip* [▶ 347], *fromzip* [▶ 340]

## 12.6 fromzip

### NAME

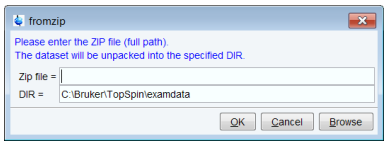
fromzip - Unzip/display a zipped TopSpin data set (nD)

### SYNTAX

fromzip [<path name> <dir> <user> ]

### DESCRIPTION

The command **fromzip** opens a dialog box to unzip a ZIP TopSpin data set.




Here you can enter the ZIP file (pathname) and the DIR and USER part of the output data path.

**fromzip** takes up to three arguments and can be used as follows:

- **fromzip**
- opens the above dialog box.
- **fromzip <pathname> <dir> <user>**
- converts the ZIP file specified by the path name and stores it under the specified <dir> and <user> and the *name*, *expno* and *procno* as stored in the ZIP archive.

In the examples above, **fromzip** stores the output data set in the directory:

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

The TopSpin data set created by **fromzip** becomes the active data set.

**INPUT FILES**

*<path name>/<mydata.bnmr.zip>* - TopSpin data as stored by **tozip**

**OUTPUT FILES**

**For 1D and 2D data:**

*<tshome>/prog/curdir/<user>/*
*curdat* - current data definition
*<dir>/data/<user>/nmr/<name>/<expno>/*
*audita.txt* - acquisition audit trail (if input file contains raw data)

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*
*auditp.txt* - processing audit trail (if input file contains processed data)
*outd* - output device parameters
*title* - title file (see **edti**)

**For 1D data:**

*<dir>/data/<user>/nmr/<name>/<expno>/*
*fid* - 1D raw data (if input file contains 1D raw data)
*acqu* - acquisition parameters
*acqus* - acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*
*1r* - real processed 1D data (if input file contains 1D real processed data)
*1i* - imaginary processed 1D data (if input file contains 1D imaginary data)
*proc* - processing parameters
*procs* - processing status parameters

**For 2D data:**

*<dir>/data/<user>/nmr/<name>/<expno>/*
*ser* - 2D raw data (input if Output Data = raw)
*acqu* - F2 acquisition parameters
*acqu2* - F1 acquisition parameters
*acqus* - F2 acquisition status parameters
*acqu2s* - F1 acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*
*2rr* - real processed 2D data (if input file contains 2D real processed data)
*proc* - F2 processing parameters
*proc2* - F1 processing parameters
*procs* - F2 processing status parameters
*proc2s* - F1 processing status parameters
*clevels* - 2D contour levels
For 3D data, the additional parameter files *acqu3*, *acqu3s*, *proc3* and *proc3s* will be created.

### SEE ALSO

*tozip* [▶ 347], *tojdx* [▶ 344], *totxt* [▶ 346], *fromzip* [▶ 340]

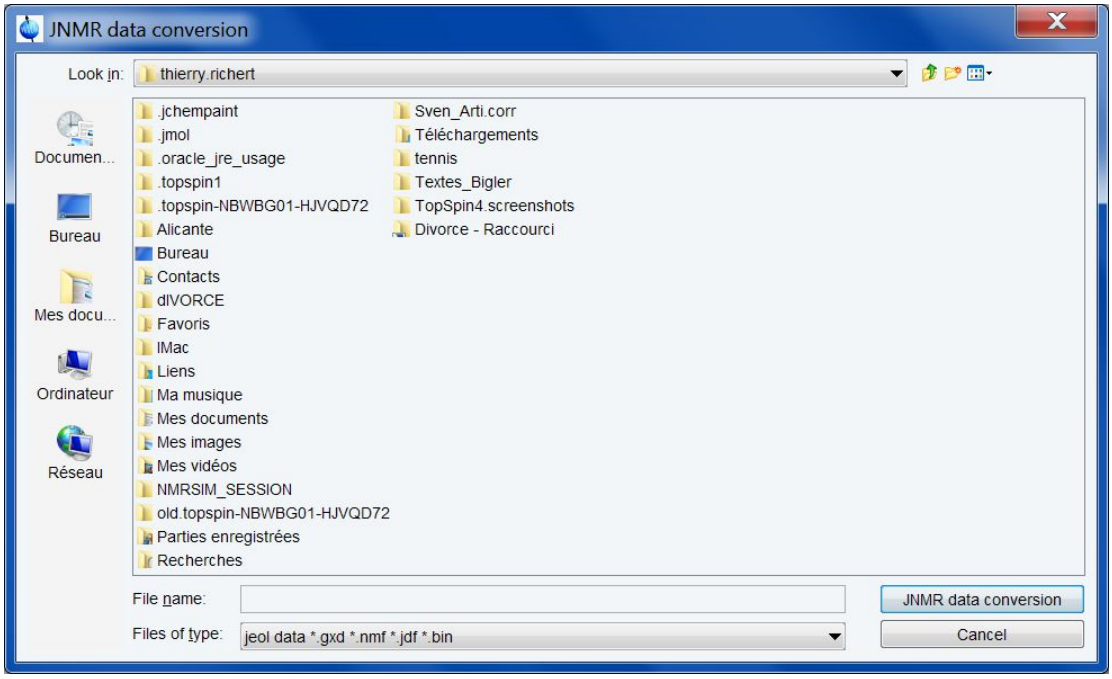
## 12.7 jconv

### NAME

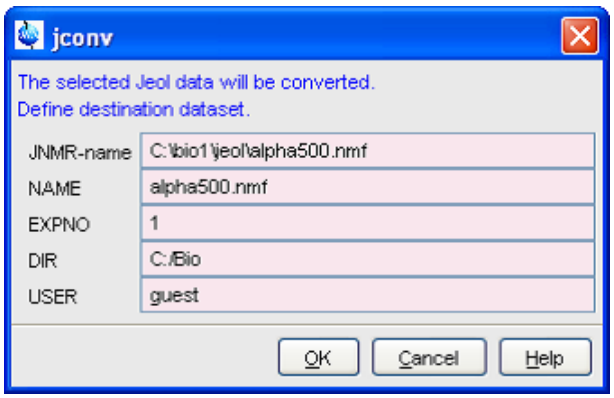
jconv - Convert Jeol type data to Bruker TopSpin data (1D, 2D, 3D)

### DESCRIPTION

The command **jconv** converts Jeol raw data to TopSpin format. It opens a dialog window where you can navigate to the Jeol input data file.




Just select the desired file and click **JNMR data conversion**. This will open the dialog box shown:




Here you can specify the TopSpin destination data set and click **OK** to start the conversion.

The **jconv** source and destination data can also be entered on the command line. Here are some examples:

**jconv jdata.<ext>**

Searches for *jdata.<ext>* in the directory defined by the environment variable JNMR (can be set with the TopSpin command env set JNMR=<path>). When the specified input data are found, the dialog window shown in the figure above will appear. Here, you can specify the output data set.

**jconv <path>/jdata.<ext>**

As above, except that the source data are searched for in the directory *<path>*

**jconv jdata.<ext> <name> <expno> <dir> <user>**

Here, the destination dataset is specified as command line arguments. The *procno* is automatically set to 1. If the data set specification is incomplete, the dialog window shown in the figure above will appear.

**jconv** can handle Jeol EX, GX and ALPHA raw data and works on 1D, 2D and 3D data. Processed data cannot be converted. The conversion of FX FID data has been implemented. FX data must have a numerical extension (like in proton.1) and the name must be specified on the command line, e.g. **jconv proton.1**. No parameter file is needed for the conversion, the most relevant parameters are extracted from the header of the data file.

| Data type | Extension of data file | Extension of parameter file |
|---|---|---|
| EX | .gxd | .gxp |
| GX | .gxd | .gxp |
| ALPHA | .nmf | .txt |
| DELTA(new) | .jdf | .jdf |
| DELTA(old) | .bin | .hdr |
| FX | .num (an integer number) | no parameter file |

**jconv** converts all Jnmr parameters which have a TopSpin equivalent. First, the Jnmr parameter EXMOD is interpreted. If it is set to a certain name, **jconv** checks the existence of a TopSpin parameter set with that name. If it exists, it is copied to the destination data set. If it does not exist, a standard parameter set (*standard1D* for 1D data) is copied. Then **jconv** converts all Jnmr parameters which have a TopSpin equivalent and overwrites the values of the parameter set which was previously copied. The parameters of the TopSpin parameter set which do not have a Jnmr equivalent keep their original values. If you frequently convert Jnmr data, with typical values of EXMOD, you might want to create the TopSpin parameter sets with the corresponding names. This can be done by reading a standard parameter set with **rpar**, modify it with **eda** and **edp** and then store it with **wpar**.

## INPUT FILES

*<jdata.ext>* - Jeol raw data

## OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - TopSpin 1D raw data

*acqu* - TopSpin acquisition parameters

*acqus* - TopSpin acquisition status parameters

*audita.txt* - acquisition audit trail

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/1/*

*proc* - TopSpin processing parameters

*procs* - TopSpin processing status parameters

*jnm* original Jeol parameter file

For 2D and 3D data, the raw data are stored in the file *ser* and the additional parameter files *acqu2(s)*, *acqu3(s)*, *proc2(s)* and *proc3(s)* are created.

### USAGE IN AU PROGRAMS

JCONV(jname, uxname, uxexp, uxdisk, uxuser)

### SEE ALSO

*vconv* [▸ 350], *fconv* [▸ 336], *conv* [▸ 333], *winconv* [▸ 353], *convdta* [▸ 334]

## 12.8 tojdx

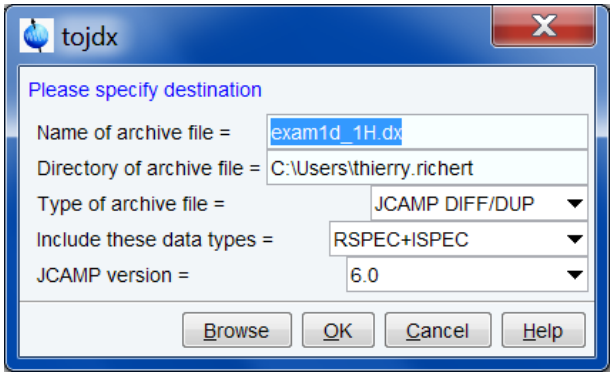
### NAME

tojdx - Convert dataset to JCAMP-DX format (1D, 2D)

### DESCRIPTION

The command **todjx** converts a TopSpin data set to JCAMP-DX format. JCAMP-DX is a standard ascii exchange format for spectroscopic data.

When **tojdx** is entered without argument, it will open a dialog box in which you can enter the required information.




This dialog box includes:

#### Name of the archive file

The file name should have the extension *.dx*. This allows you to open it in TopSpin with drop & drag. Default is the data set name with the extension *.dx*.

#### Directory of the archive file

Any directory. Default is the users home directory.

#### Type of archive file

For JCAMP format, you can choose between the following archive types:

- *FIX* (=0): Table format.
- *PACKED* (=1): No spaces between the intensity values.
- *SQUEEZED* (=2): The sign of the intensity values is encoded in the first digit.

- *DIFF/DUP* (=3): The difference between successive values is encoded, suppressing repetition of successive equal values.

The default value is *DIFF/DUP*.

**Include these data types**

For the included data types, you have the following choices:

- FID (=0): Raw data.
- RSPEC (=1): Real processed data.
- RSPEC+ISPEC (=2): Real and imaginary processed data.
- PARAMS (=3): Parameter files.
- FID+RSPEC+ISPEC (=4): Raw data + real and imaginary processed data.
- FID+ALL_PROCNOS (=5): Raw data +real and imaginary processed data of all PROCNO's under the current EXPNO.
- ALL_EXPNOS_DIM_1_2 (=6): Raw data +real and imaginary processed data of all EXPNO's under the current NAME.
- FID+RSPEC+ISPEC (=4): Raw and real + imaginary processed data.
- ALL PROCNOS (=5): All procnos under current expno.
- ALL EXPNOS (=6): All expnos under current name.

The default value is RSPEC+ISPEC (=2)

The above information can be entered as arguments of **tojdx** as follows:

**tojdx <path> <data> <file> <title> <origin> <owner>**

Note that in this case three extra arguments are required. The arguments have the following meaning:

*<path>*: Name and directory of the archive file.

*<data>*: Data types included.

*<file>*: Type of archive file.

*<title>*: The title as it appears in the output file: enter a character string.

*<origin>:* The origin as it appears in the output file: enter a character string.

*<owner>*: The owner as it appears in the output file: enter a character string.

The default *title* is the plot title as defined with **edti**. If no plot title is defined the data name is taken as default. The default *origin* and *owner* are taken from the acquisition status parameter files (*acqus*). If you enter an * character as argument, the default value will be used.

Here are some examples:

**tojdx C:\temp\mydata.dx 0 2 mytitle BRUKER guest**

**tojdx D:\nmr\mydata.dx 0 2 mytitle * ***

**tojdx * 1 * mytitle MYORIGIN joe**

**tojdx F:\users\guest\mydata.dx * * * * ***

**INPUT FILES**

**For 1D and 2D data:**

*<tshome>/prog/curdir/<user>/*

*curdat* - current data definition

**For 1D data:**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - 1D raw data

*acqus* - acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*1r* - real processed 1D data

*1i* - imaginary processed 1D data

*proc* - processing status parameters

*procs* - processing status parameters

**For 2D data:**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*ser* - 2D raw data

*acqus* - F2 acquisition status parameters

*acqu2s* - F1 acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*2rr* - real processed 2D data

*proc* - F2 processing parameters

*proc2* - F1 processing parameters

*procs* - F2 processing status parameters

*proc2s* - F1 processing status parameters

**OUTPUT FILES**

*<path name>/<mydata.dx>* - TopSpin data in JCAMP-DX format

**USAGE IN AU PROGRAMS**

TOJDX(name, data, mode, title, origin, owner)

For example **TOJDX("/tmp/mydata.dx", 0, 2, "mytitle", "BRUKER", "joe")**

**SEE ALSO**

*fromjdx* [▸ 338], *tozip* [▸ 347], *totxt* [▸ 346]

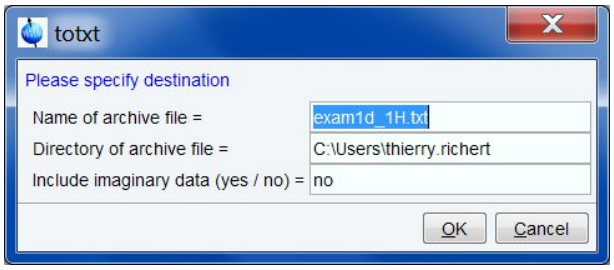
## 12.9 totxt

**NAME**

totxt - Save the currently displayed region as a text file (1D, 2D)

**DESCRIPTION**

The command **totxt** saves the currently displayed spectral region as text file. It will open the following dialog box in which you can enter the text file name and directory:




**totxt** works on 1D and 2D data sets and only stores the real processed data. The 1D file format is:

# File created = Wednesday, March 3, 2004 11:52:01 AM CET

# Data set = exam1d_13C 1 1 C:\bio guest

# Spectral Region:

# LEFT = 145.2549493926 ppm. RIGHT = 116.58206350384 ppm.

# SIZE = 3940 (= number of points)

# In the following ordering is from the 'left' to the 'right' limits!

# Lines beginning with '#' must be considered as comment..

# 1.4612096E7

```
3084512.0
4615664.0
1.6594048E7
4898192.0
-4555792.0 ...
```

**INPUT FILES**

**For 1D data:**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*1r* - real processed 1D data

**For 2D data:**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*2rr* - real processed 2D data

**OUTPUT FILES**

*<pathname>/<mydata.txt>* - text file containing displayed region

**SEE ALSO**

*tojdx* [▶ 344], *tozip* [▶ 347]

## 12.10 tozip

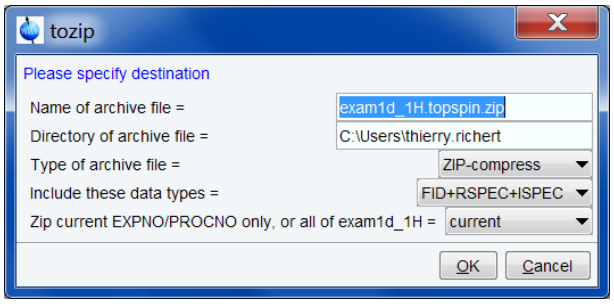
**NAME**

tozip - Store current dataset in ZIP file (nD)

## DESCRIPTION

The command **tozip** converts a TopSpin dataset to ZIP format.

It opens a dialog box where you can enter the required information:




This information includes:

Name of archive file: output file name and extension *(<datasetname> .topspin.zip*)

Directory of archive file: directory where output file is stored.

**Type of archive**:

- ZIP-compress - Compressed nmr data in zip format.
- ZIP-no compress - Uncompressed nmr data in zip format.

**Data types included**:

- FID+RSPEC+ISPEC: Raw, real and imaginary processed data.
- FID+RSPEC: Raw + real processed data.
- FID: Raw data.
- RSPEC+ISPEC: Real and imaginary processed data.
- RSPEC: Real processed data.

**Zip current EXPNO/PROCNO only, or all of** ...: Archive current expno/procno or all expnos/procnos in current data set.

**Options for tozip dialog window:**

Without argument, tozip will open it's dialog showing the default destination file *<dataname>.topspin.zip.* You can change this default as follows:

1. Enter **expl prop** in TopSpin command line to open the file explorer in the user properties directory.
2. Edit the file *globals.prop*.
3. Add the line:

   **TOZIP_CONFIG=option1|option2**

   Where the options must be separated by the character "|" and
   - option1= N, NE or NEP, for name, name-expno or name-expno-procno, respectively.
   - option2 = any string, e.g. "-mycompany.zip"

**Example:**

Dataset: "exam1d_13C 102 1 c:\bruker\topspin guest"

option2=.bruker.zip

- If option1=N:

  - the default name is: exam1d_13C.bruker.zip.
- If option1= NE:
  - the default name is exam1d_13C-102.bruker.zip
- If option1 was NEP:
  - the default name is exam1d_13C-102-1.bruker.zip

**Options for the command tozip**

- Arguments for the command tozip:
- The command **tozip** takes four arguments, "tozip optionA, optionB, optionC, optionD":
  - optionA = nmr-data which should be transferred to zip file.
  - optionB = name and directory of archive data.
  - optionC = FID_RE_IM, FID_RE, FID, RE_IM, RE, PARAMS.
  - optionD = COMPRESS, NO_COMPRESS.

**Zipfile from command line:**

The command **tozip** can be executed on the command line with the option '- d' and only the path name of the new zip file:

**tozip -d <path>/<filename>.zip**

This command transfers the raw and processed data in uncompressed zip-format. If the graphical user interface should be used, simply enter the command **tozip** as described above.

**Zip file from within an AU Program:**

In AU Programs both commands **tozip** and **tozip -d** can be used with the command **sendgui**.

The following two examples show the entering-procedure:

**XCMD("sendgui tozip -d C:/mydata.zip")**

**QUIT**

**XCMD("sendgui tozip C:/Bruker/ts21pl1/data/guest/nmr/exam1d_1H/1/pdata/1, C:/testdata.zip, FID_RE_IM, NO_COMPRESS")**

**QUIT**

**INPUT FILES**

**If Data type includes FID**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - 1D raw data

*ser* - 2D or 3D raw data

**If Data type includes RSPEC**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*1r* - real processed 1D data

*2rr* - real processed 2D data

*3rrr* - real processed 3D data

**If Data type includes ISPEC**

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>*

*1i* - imaginary processed 1D data

*2ir*, *2ri*, *2ii* - imaginary processed 2D data

*3irr*, *3rir*, *3iii* - imaginary processed 3D data

The parameter files *acqu** and *proc** are stored for all data types.

**OUTPUT FILES**

*<pathname>/<mydata.topspin.zip>* - TopSpin data in ZIP format

**SEE ALSO**

*fromzip* [▶ 340], *tojdx* [▶ 344], *totxt* [▶ 346]

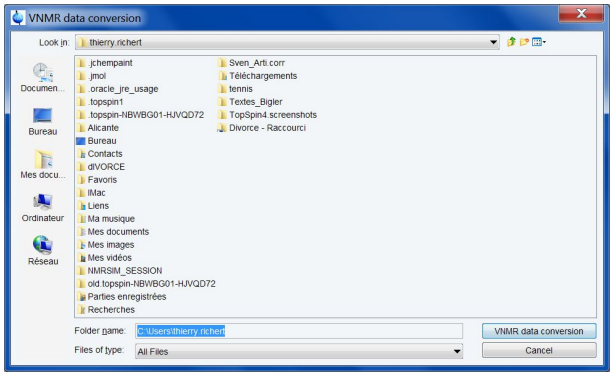
## 12.11 vconv

**NAME**

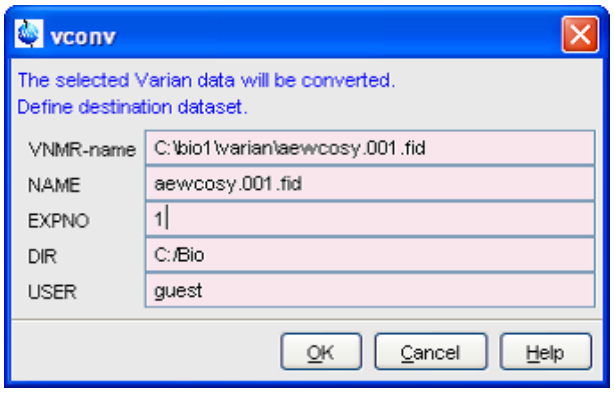
vconv - Convert Varian type data to TopSpin data (1D, 2D, 3D)

**DESCRIPTION**

The command **vconv** converts Varian data, which were measured with the Vnmr program, to TopSpin format.




It opens a browser where you can navigate to the Varian input data file. Just select the desired file and click **VNMR data conversion**. This will open the dialog box shown:




Here you can specify the TopSpin destination dataset and click **OK** to start the conversion.

The **vconv** source and destination data can also be entered on the command line. Here are some examples:

**vconv vdata.fid**

searches for *vdata.fid* in the directory defined by the environment variable VNMR (can be set with the TopSpin command env set VNMR=<path>). When the specified input data are found, the dialog window shown in the figure above will appear. Here, you can specify the output data set.

**vconv <path>/vdata.fid**

as above, except that the source data are searched for in the directory *<path>*

**vconv vdata.fid <name> <expno> <dir> <user>**

Here, the destination data set is specified as command line arguments. The *procno* is automatically set to 1. If the data set specification is incomplete, the dialog window shown in the figure above will appear.

**Note** that the extension *.fid* of the Vnmr dataset is not obligatory.

**vconv** converts all Vnmr parameters which have a TopSpin equivalent. First, the Vnmr parameter SEQFIL is interpreted. If it is set to a certain name, **vconv** checks the existence of a TopSpin parameter set with that name. If it exists, it is copied to the destination dataset. If it does not exist, a standard parameter set (*standard1D* for 1D data) is copied. Then **vconv** converts all Vnmr parameters which have a TopSpin equivalent and overwrites the values of the parameter set which was previously copied. The parameters of the TopSpin parameter set which do not have a Vnmr equivalent keep their original values. If you frequently convert Vnmr data, with typical values of SEQFIL, you might want to create the TopSpin parameter sets with the corresponding names. This can be done by reading a standard parameter set with **rpar**, modify it with **eda** and **edp** and then store it with **wpar**.

**NMR parameter equivalence between Bruker and Varian software**

| VNMR | XWIN-NMR / TopSpin | VNMR | XWIN-NMR / TopSpin |
|---|---|---|---|
| ct | NS(status) | rfl/rfp | OFFSET |
| d1 | D1 | rfl1/rfp1 | OFFSET(2D) |
| date | DATE | rfl2/rfp2 | OFFSET(3D) |
| dfrq | BF2 | rp | PHC0 |
| dfrq2 | BF3 | rp/lp | PHC0/PHC1 |
| dmf | P31 | rp1/lp1 | PHC0/PHC1(2D) |
| dn | DECNUC | rp2/lp2 | PHC0/PHC1(3D) |
| dn2 | DECBNUC | seqfil | PULPROG |
| dof | O2 | sfrq | BF1 |
| dof2 | O3 | solvent | SOLVENT |
| fb | FW | spin | RO |
| fn | SI | ss | DS |
| lp | PHC1 | sw | SW_h |
| np | TD | sw1 | SW_h(2D) |
| nt | NS(foreground) | sw2 | SW_h(3D) |

| VNMR | XWIN-NMR / TopSpin | VNMR | XWIN-NMR / TopSpin |
|---|---|---|---|
| pp | P3 | temp | TE |
| pslabel | AUNM | tn | NUCLEUS |
| pw | P0 | tof | O1 |
| pw90 | P1 | | |

The original Vnmr parameter file *procpar* is stored in the TopSpin processed data directory. You can check this ascii file for possible parameters which could not be converted.

The table above shows the Varian parameters and there TopSpin equivalent.

**vconv** can handle Unity and Gemini data acquired with Vnmr 4.1 or newer. Data from older Varian spectrometers or acquired with older software versions might also work, but have not been tested by Bruker.

### INPUT FILES

*<dir>/data/<user>/nmr/<vdata>.fid*

or

*<VNMR>/<vdata>.fid/*

*fid* - the Vnmr raw data

*procpar* - the parameters

*text* - title file

### OUTPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - TopSpin 1D raw data

*acqu* - TopSpin acquisition parameters

*acqus* - TopSpin acquisition status parameters

*audita.txt* - acquisition audit trail

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/1*

*proc* - TopSpin processing parameters

*procs* - TopSpin processing status parameters

*procpar* - Vnmr parameter file

For 2D and 3D data, the raw data are stored in the file *ser* and the additional parameter files *acqu2(s)*, *acqu3(s)*, *proc2(s)* and *proc3(s)*are created.

### USAGE IN AU PROGRAMS

VCONV(vname, xwname, xwexpno, xwdisk, xwuser)

### SEE ALSO

*jconv* [▸ 342], *fconv* [▸ 336], *conv* [▸ 333]

## 12.12 winconv

### NAME

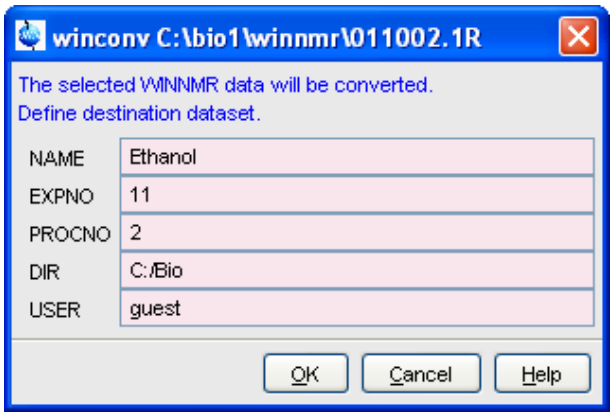
winconv - Convert WINNMR type data to TopSpin data (1D)

### DESCRIPTION

The command **winconv** converts Bruker Win-nmr data to TopSpin format. It opens a browser where you can navigate to the Win-NMR input datasets. A Win-nmr dataset is a directory with several files. Each file has:

- A number as file name.
- The extension *.FID*, *.1R, .1I*, *.AQS* or *.FQS* for raw data, processed real data, processed imaginary data, acquisition parameters and processing parameters, respectively.

Just select any of these files and click **convert**. This will open the dialog box shown:




Here you can specify the TopSpin destination dataset. The data path fields are initialized as follows:

NAME - the Win-nmr data directory

EXPNO - the first three digits of the Win-nmr data name

PROCNO - the second three digits of the Win-nmr data name

DIR - DIR of the active TopSpin data set

USER - USER of the active TopSpin data set

Specify a data path or accept the initial values and click **OK** to start the conversion. To display the data set, open it from the TopSpin browser or use the command **re**.

### INPUT FILES

*<name>/*

*num.FID* - Win-nmr raw data

*num.1R* - Win-nmr real processed data

*num.1I* - Win-nmr imaginary processed data

*num.1I* - Win-nmr imaginary processed data

*num.AQS* - Win-nmr acquisition parameters

*num.FQS* - Win-nmr processing parameters

*num.TIT* - Win-nmr title

**OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*fid* - TopSpin 1D raw data

*acqu* - TopSpin acquisition parameters

*acqus* - TopSpin acquisition status parameters

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/1*

*1r* - real processed data

*1i* - imaginary processed data

*proc* - TopSpin processing parameters

*procs* - TopSpin processing status parameters

**SEE ALSO**

*conv* [▷ 333], *fconv* [▷ 336], *jconv* [▷ 342], *vconv* [▷ 350], *convdta* [▷ 334]

# 13 TopSpin Interface/Processes

This chapter describes commands which are related to the User interface and TopSpin processes. Each user can set up his/her own interface including the TopSpin menu, colours, printer usage etc. Commands are described for following processes on the screen, storing them in the history file or killing them. Online help is described as far as it can be started from the command line.

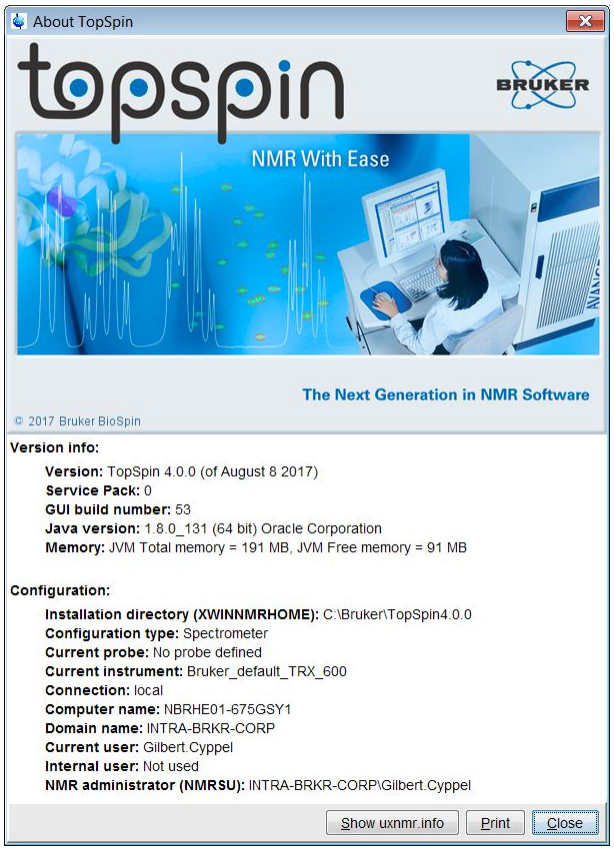
## 13.1 about

### NAME

about – Displays the TopSpin version and configuration information. .

### DESCRIPTION

The command **about** displays various TopSpin version and configuration informations:




Alternatively click **Help** | **Version Info**.

## 13.2 bpan

### NAME

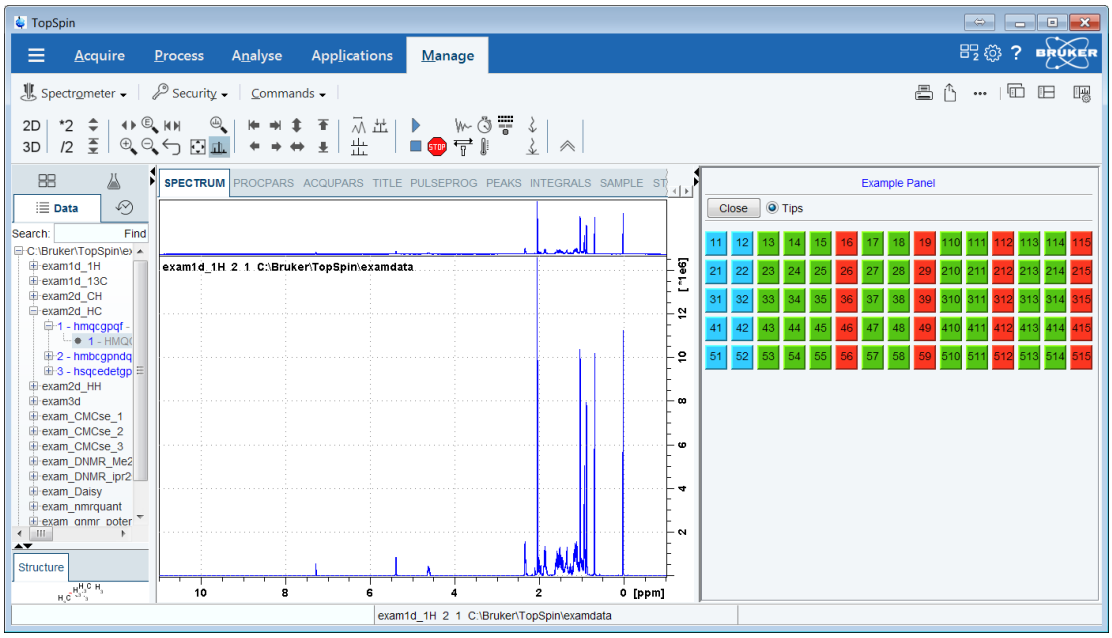
bpan - Opens a user defined button panel (nD)

## DESCRIPTION

The command **bpan** opens a user defined button panel. It prompts for the name of the desired panel.


A button panel is a window with user-defined buttons for executing TopSpin commands, AU programs, Jython programs or macros. It appears as an integral part of the active data window and act on that. Bruker delivers a few standard button panels available with the command **bnmr**. To create your own button panels, you can modify one of these or write them from scratch.

In this description we will create a very simple button panel with some 1D processing commands and **print/export** buttons:




To write this button panel, take the following steps:

1. Open the File Explorer and navigate to the subdirectory *userdefined* of the users properties directory (to locate this, enter hist and look for the entry "User properties directory=").
2. Create a text file with the name *buttonpanel_<name>.prop*, where *<name>* is the name of the button panel.
3. Enter the button definitions including **Panel title**, **Colors**, **Toggle buttons**, **Top buttons**, **Panel layout**, **Panel buttons** and **Tooltips**.
4. Save the file. Make sure the extension of the file is *.prop* and not *.txt*, *.prop.txt* or anything else.
5. Enter **bpan <name>** on the command line to open the button panel.

Here is the content of the properties file for the button panel above:

```
# Color definitions used in this file (RGB)

BLUE1=51$ 204$ 255
```

```
YELLOW1=255$ 255$ 0
GREEN1=84$ 196$ 20
# Title definition
TITLE=1D Processing Panel
TITLE_COLOR=0$ 0$ 255
# Toggle button definition
TOGGLE_BUTTON=To 2D
TOGGLE_CMD=bpan bproc2d
TOGGLE_TIP=Switch to 2D processing
# Top row button definition
TOP_BUTTONS=EM$ $FT$ $PK$ $
TOP_COLORS=YELLOW1$ YELLOW1$ YELLOW1
TOP_CMDS=em$ ft$ pk
TOP_TIPS=Exponential multiplication $\
Fourier transform$\
Phase correction
# Panel button definitions
# LAYOUT format: rows columns hgap vgap
PAN_LAYOUT=1$ 3$ 8$ 8
PAN_BUTTONS=Print$ $ EXPORT$ $SEND TO$ $
PAN_COLORS=BLUE1$ BLUE1$ BLUE1
PAN_CMDS=prnt$ exportfile$ smail
PAN_TIPS=Print the spectrum<br>\
as it appears on the screen$\
Export the dataset<br>\
to png, jpg, bmp etc.$\
Send the dataset by email
```

Note that:

- The **Close** button and the **Tips** checkbox are automatically created. You don’t need to specify them.
- The **TOGGLE** button is typically, but not necessarily, used to call another button panel. In this example it calls the panel **bproc2d**.
- Items must be separated with the "$" character, button items with "$ $".
- A "\" followed by "end of line" continues an item on the next line.
- Tool tips may use html tags for text formatting.
- Commands may be specified as single commands like "em" or as composite commands like "em\nft\npk". Note that in the latter case, the commands must be separated by "\n".

**INPUT FILES**

*<userhome>/<.topspin-hostname>/prop/userdefined/cmdpanel_<name>.prop*

### SEE ALSO

(bnmr)

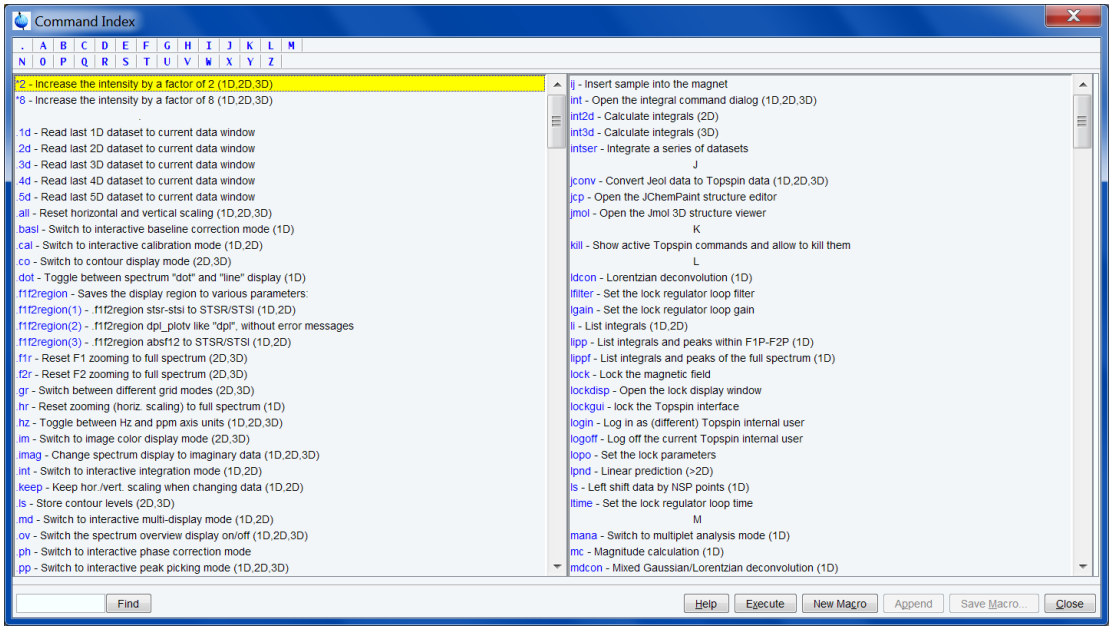
## 13.3 cmdindex

### NAME

cmdindex - Open the command index

### DESCRIPTION

The command **cmdindex** opens a command index dialog box:




It displays all TopSpin commands which can be entered from the command line with a one-line description for each command. Select one or more commands for further actions. The following actions are available:

#### Help

Open the HTML Help page of the selected command. This is equivalent to double-clicking the command.

#### Execute

Execute the selected command or commands.

#### New Macro

Create a new macro and append commands from the list or enter commands manually.

#### Append

Append the (first) selected command to the command line. The appended command can be edited and executed. Useful for commands with many arguments such as **re**.

#### Save Macro

The selected command(s) are stored as a macro. You will be prompted for the macro name. To edit this macro, enter **edmac <macro-name>**. To execute it, just enter the name on the command line.

**Find**

Find a character string in the command index.

**INPUT FILES**

*<tshome>/classes/prop*

*cmdindex_main.prop* - command index properties file

*<tshome>/prog/docu>/english/xwinproc/html*

**.html* - TopSpin command help files

**OUTPUT FILES**

*<tshome>/exp/stan/nmr/lists/mac/*

* - Macros (created by **cmdindex** and **Save Macro.**.)

**SEE ALSO**

*cmdhist [ 359]*

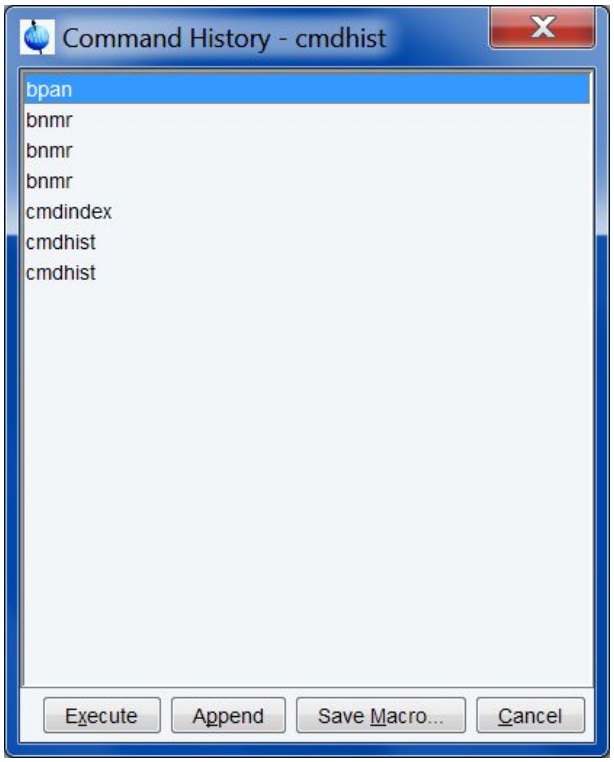
## 13.4 cmdhist

**NAME**

cmdhist - Open command history.

**DESCRIPTION**

The command **cmdhist** opens a command history control window:




It displays all commands that have been entered from the command line since TopSpin was started. You can select one or more commands. Furthermore, the following buttons are available:

**Execute**

Execute the selected command or commands.

**Append**

Append the (first) selected command to the command line. The appended command can be edited and executed. Useful for commands with many arguments such as **re**.

**Save Macro...**

The selected command(s) are stored as a macro. You will be prompted for the macro name. To edit this macro, enter **edmac <macro-name>**. To execute it, just enter the name on the command line.

To open the command history control window right-click in the command line and select **Command Line History**.

### OUTPUT FILES

*<tshome>/exp/stan/nmr/lists/mac/*

* - Macros (created by **cmdhist** and **Save Macro..**)

### SEE ALSO

*hist* [▶ 365], *edpul, edcpde* [▶ 298]

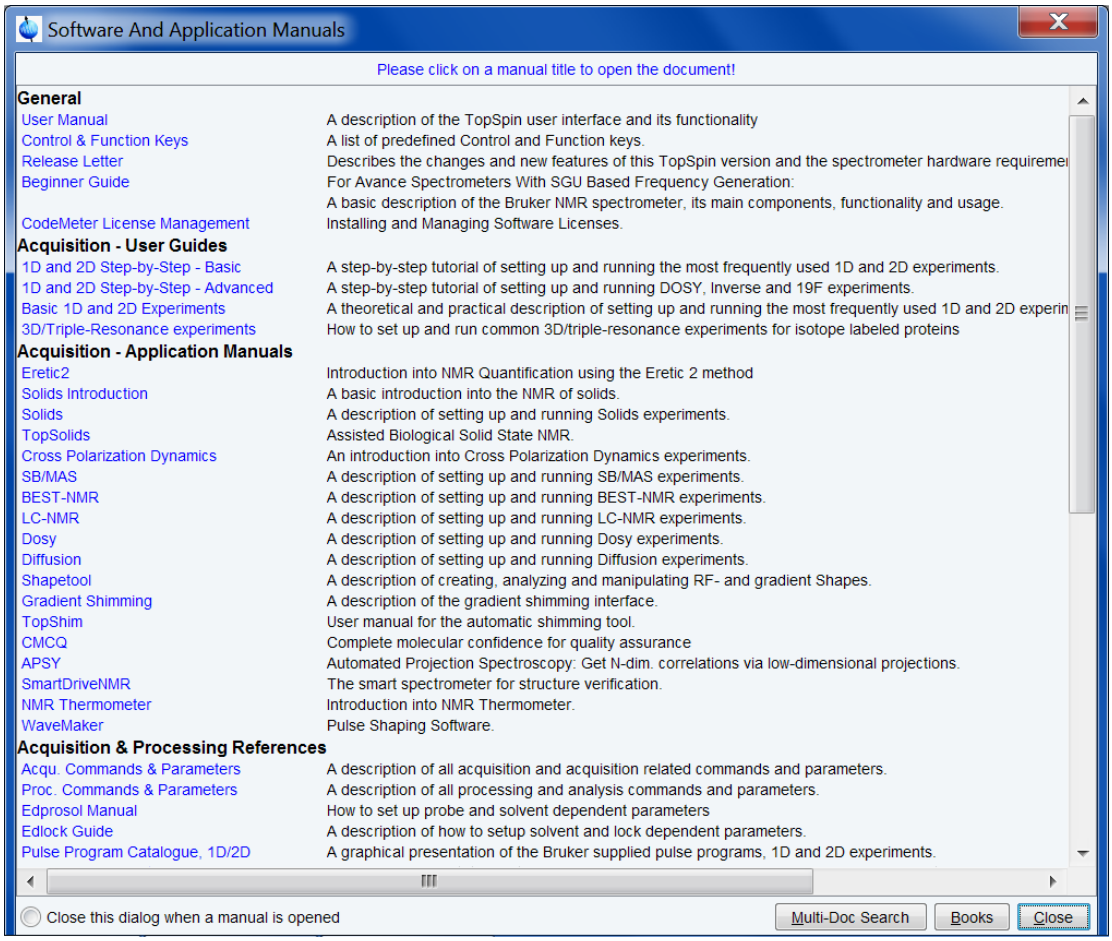
## 13.5 docs

### NAME

docs - Open Manual list.

### DESCRIPTION

The command **docs** opens a list of available documents. This list displays all Bruker manuals delivered on the TopSpin DVD:




The manuals are divided in topic groups as **General**, **Acquisition - User Guides**, etc...

Just click the manual name to open it. Furthermore, the Manual dialog offer the following buttons or check box:

- **Check box**: Close this dialog when a manual is opened.
- **Multi-Doc Search**: A small help for the advanced search in Adobe to find a search string in several manuals.
- **Books** : A list of available hardcopy (printed) manuals.
- **Close**: Close this dialog.

### SEE ALSO

*help, ghelpg [▶ 366]*

## 13.6 edtext

### NAME

edtext - Open an empty text file with an editor.

### DESCRIPTION

The command edtext opens an empty text file with the TopSpin editor. The file can be stored in any directory.

### SEE ALSO

*nbook [▷ 367]*

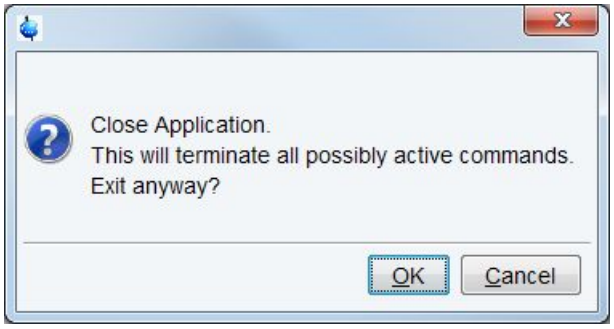
## 13.7 exit

### NAME

exit - Exit TopSpin

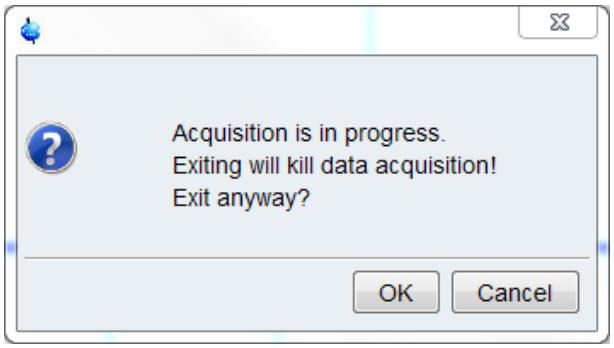
### DESCRIPTION

The command **exit** exits TopSpin and terminates all running processes. Before this happens, the following warning is displayed:

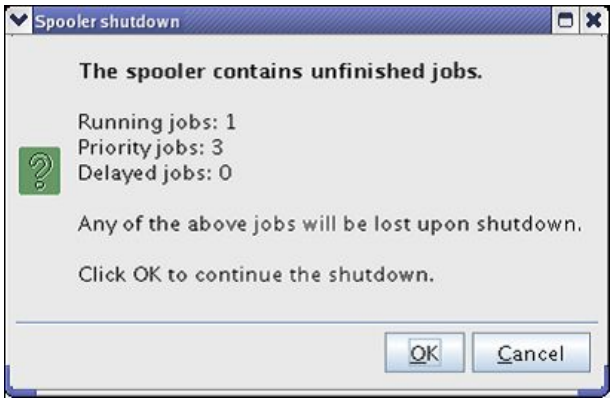



TopSpin displays different warnings and error messages, depending on the actual TopSpin use, before exiting the program:

- If Acquisition is running:




- If the spooler contains unfinished jobs click **OK** in the respectively of the three dialogs above to exit TopSpin.




- If IconNMR runs actively at the exit-moment, TopSpin cannot be closed:


Entering **exit** on the command line is equivalent to clicking **File | Exit**.

**SEE ALSO**

*newwin, nextwin, close, closeall [*▸ 368]

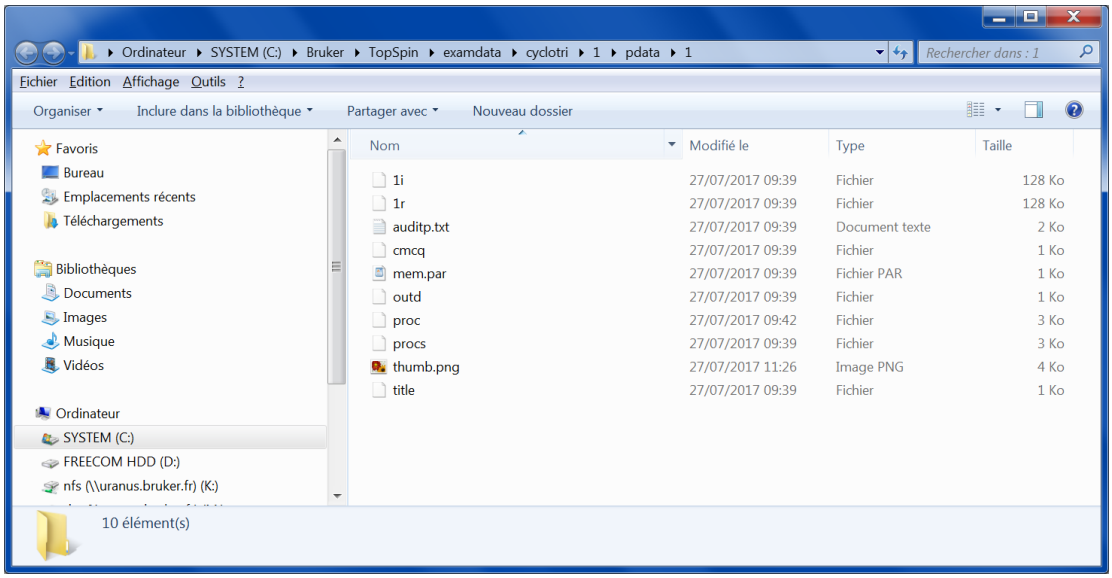
## 13.8 expl

**NAME**

expl - Open File Explorer, show current processing folder.

**DESCRIPTION**

The command **expl** opens the Explorer (Windows) or Konqueror/Nautilus (Linux) showing the processed data files (the files in the *procno* directory) of the active dataset:




If no data set is open in the TopSpin data area, the users home directory will be shown.

**expl** allows you to access to the current data files as well as the entire data directory tree. An alternative way to access the processed data files is to right-click in the data window and select *Files...*

The command can also be used with one argument:

**expl top**

Opens the TopSpin home directory

**expl home**

Opens the User home directory

**expl spect**

Opens the directory *<tshome>/conf/instr/<curinst>*

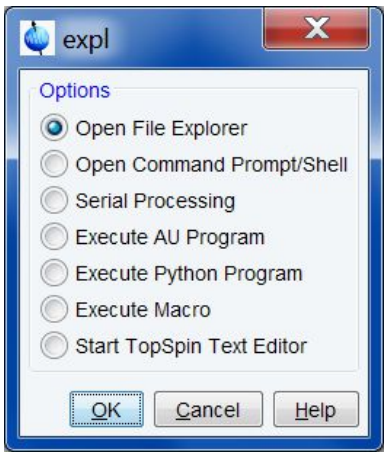
**expl prop**

Opens the User properties directory

**expl <absolute_path>**

Opens the directory *<absolute_path>*

**expl** can also be started from **File | Run a Program**.



### SEE ALSO

*run* [▶ 326]

## 13.9 hist

### NAME

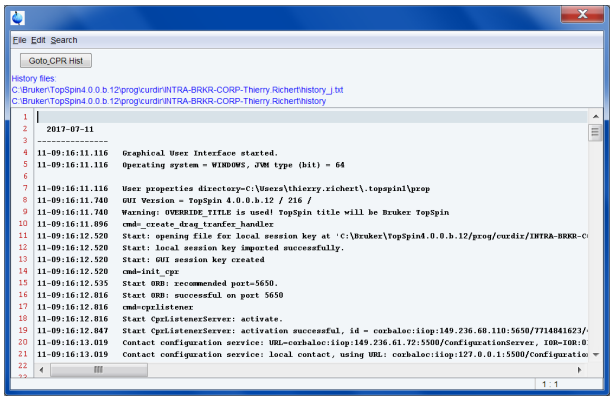
hist - Show the TopSpin history and protocol.

### DESCRIPTION

The command **hist** displays the TopSpin protocol and history files. These files only contain information if the protocol function is active. You can switch on this function as follows:

1. Click **Setup preferences**  or enter **set**.
2. Click the **Miscellaneous** group in the left part of the dialog box.
3. Check the item **Record commands in protocol file**.

The protocol file contains TopSpin startup information and command information on interface level. The history file contains command information on the level of the command interpreter and application modules. It also contains error messages.




Note that the files *history* and *protocol* are emptied when you restart TopSpin which means the history of the previous TopSpin session is lost. In case of problems, you should first make a copy of these files before you restart TopSpin. Note that a long TopSpin session, especially with automation can create very large *history* and protocol files. Therefore, it is useful to regularly check the size of the files or simply restart TopSpin after each (automation) session.

### OUTPUT FILES

*<tshome>/prog/curdir/<user>/*

*history* - TopSpin history file

*history_i.txt* - TopSpin protocol file

### SEE ALSO

*ptrace* [▶ 369]

## 13.10 help, ghelp

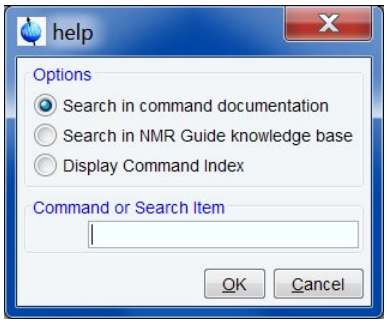
### NAME

help - Search for keywords in command help.

ghelp - Search for keywords in command in NMR Guide.

### DESCRIPTION

The command **help** opens a search dialog:




This dialog box has several options, each of which selects a certain command for execution.

**Search in command documentation**

This option activates the command **help**. It allows you to search for the specified item in the command help documents.

**Search in NMR Guide knowledge base**

This option activates the command **ghelp**. It allows you to search for the specified item in the NMR Guide knowledge base.

**Search in NMR Guide knowledge base**

This option activates the command **cmdindex**. It opens the command index dialog, irrespective of the specified command.

Entering **help** on the command line is equivalent to clicking **Help | Advanced Search**  or clicking **F1**.

### INPUT FILES

*<tshome>/prog/docu>/english/xwinproc/html*

**.html* - TopSpin command help files

*<tshome>/guide/*

* - NMR Guide files and directories

### SEE ALSO

*docs* [▶ 360]

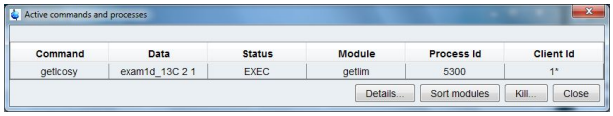
## 13.11 kill, show

### NAME

kill, show - Show active TopSpin commands and allow to kill them.

### DESCRIPTION

The command **kill** displays a list of all active TopSpin commands.




To kill a command:

- In the list select a command entry.
- Click **Kill...**

**Note:** a running acquisition should not be stopped with **kill** because this would leave an inconsistent data set. Instead, the commands **halt** or **stop** should be used for this purpose.

**Note:** the command **show** is equivalent to **kill**.

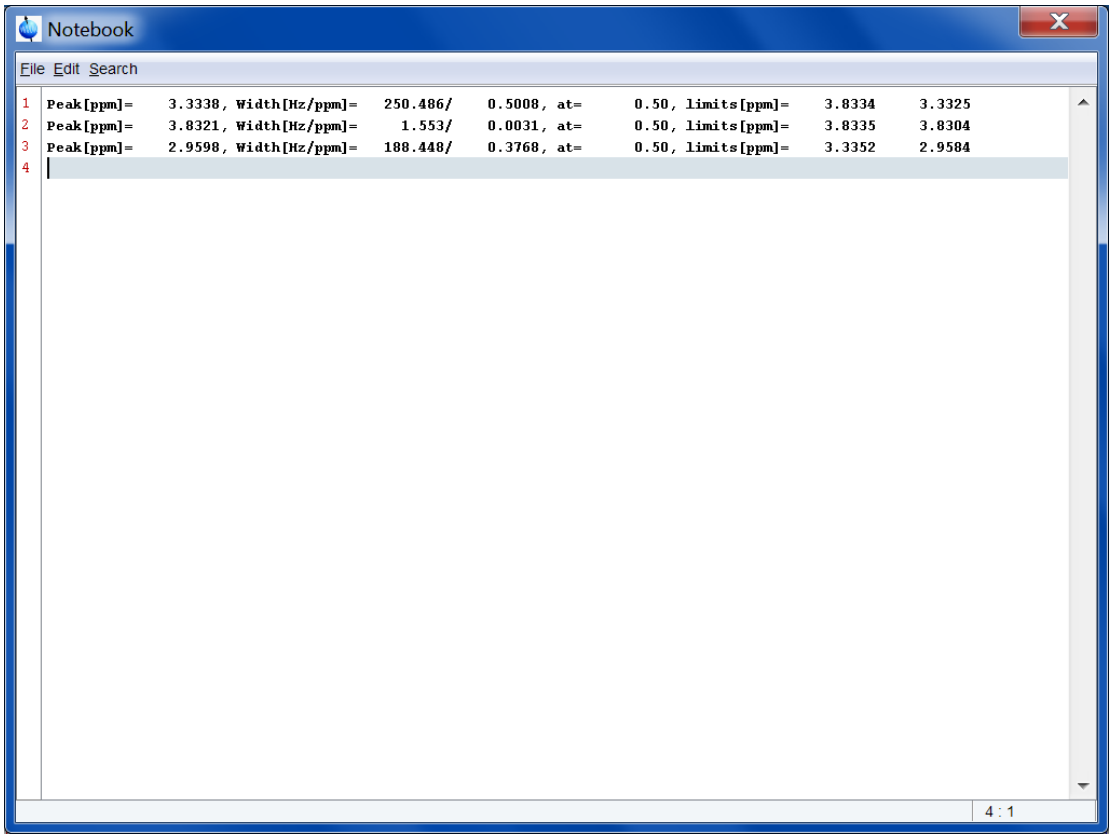
## 13.12 nbook

### NAME

nbook - Open the user notebook

### DESCRIPTION

The command **nbook** opens a user specific notebook. Each user can create and keep their own notebook for individual notes, information, settings etc.



### INPUT AND OUTPUT FILES

*<userhome>/<.topspin-hostname/prop/*

*notebook.txt* - notebook text file

### SEE ALSO

*peakw* [▸ 228]

## 13.13 newtop

### NAME

newtop - Open a new TopSpin interface.

### DESCRIPTION

The command **newtop** opens a new additional TopSpin interface. The additional interface is completely equivalent to the one it was started from. Entering **newtop** in the second or in the initial TopSpin interface opens another interface etc. The number of TopSpin interfaces is only limited by the available computer memory.


When single data set is displayed in multiple TopSpin interfaces, the display in each interface is completely independent from the others. As such, you can display different regions, scaling and data objects. When the data set is (re)processed from one interface, its display is automatically updated in all TopSpin interfaces.

The command **exit** closes the current TopSpin interface. Interfaces that were opened from that interface remain open. Entering **exit** in the last open TopSpin interface, finishes the entire TopSpin session.

The position and geometry of each TopSpin interface is saved and restored after restart.

### SEE ALSO

*exit* [▸ 362], *hist* [▸ 365], *newwin, nextwin, close, closeall* [▸ 368]

## 13.14 newwin, nextwin, close, closeall

### NAME

newwin - Open a new (empty) data window.

nextwin - Select the next data window.

close - Close the current data window.

closeall - Close all data windows.

### DESCRIPTION

The command **newwin** opens a new empty data window. The command **nextwin** activates the next open data window. It is equivalent to clicking the Window Switcher 
or clicking **F6**.

The command **close** closes the current data window. It is equivalent to clicking **File | Close Active Window** or hitting **Ctrl-w**.

The command **closeall** closes all current data windows. It is equivalent to clicking **File | Close All Windows**.



### SEE ALSO

*newtop [▷ 368]*

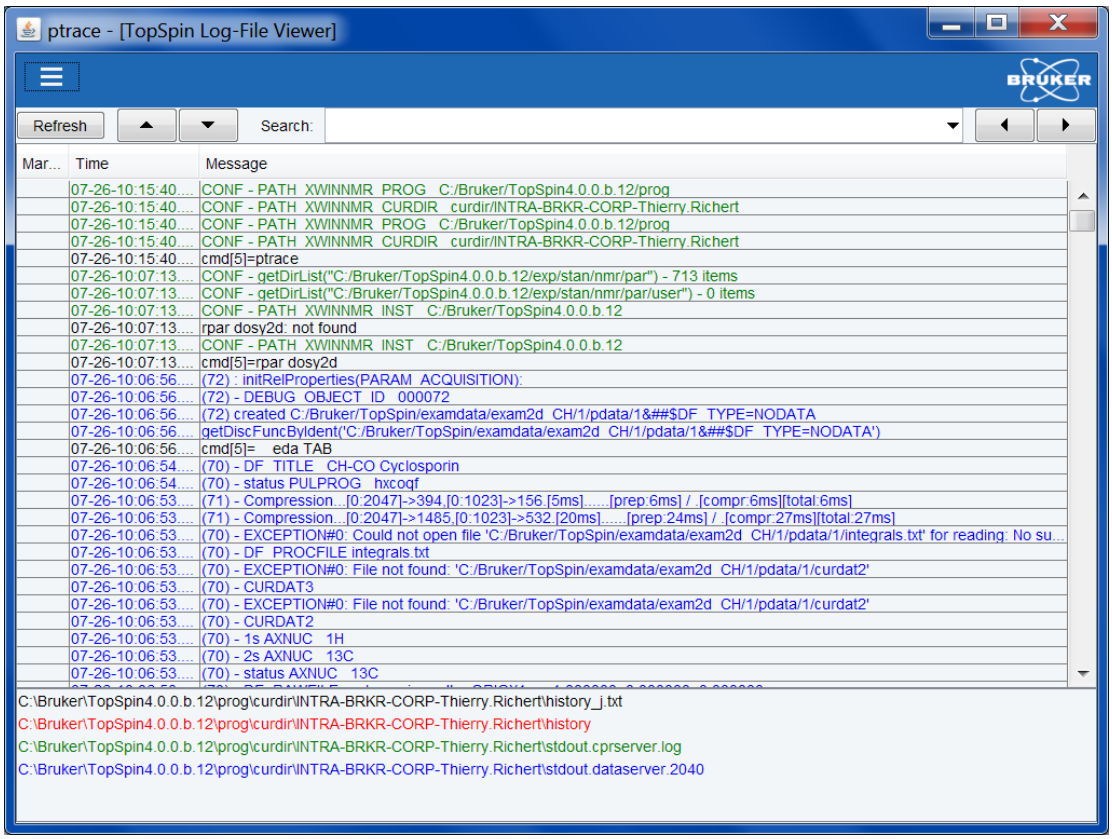
## 13.15 ptrace

### NAME

ptrace - Display messages from various log files time sorted.

### DESCRIPTION

The command **ptrace** displays the TopSpin protocol and history files time sorted:



These files only contain valuable information if the protocol function is active. You can switch on this function as follows:

1. Click **Setup preferences**  or enter **set**.
2. Click the **Miscellaneous** group in the left part of the dialog box.
3. Check the item **Record commands in protocol file**.

The protocol file contains TopSpin startup information and command information on interface level. The history file contains command information on the level of the command interpreter and application modules. It also contains error messages.

Note that the files *history* and *protocol* are emptied when you restart TopSpin which means the history of the previous TopSpin session is lost. In case of problems, you should first make a copy of these files before you restart TopSpin. Note that a long TopSpin session, especially with automation can create very large *history* and protocol files. Therefore, it is useful to regularly check the size of the files or simply restart TopSpin after each (automation) session.

### OUTPUT FILES

*<tshome>/prog/curdir/<user>/*

*history* - TopSpin history file

*history_i.txt* - TopSpin protocol file

*history.traffic.txt* - network traffic log

*stdout.dataserver.<number>.txt* – data server output file

*<userhome>/<.topspin-hostname>/prop/*

*protocol.txt* - TopSpin protocol file (if TopSpin was started as **topspin -client**)

### SEE ALSO

*hist* [▶ 365]

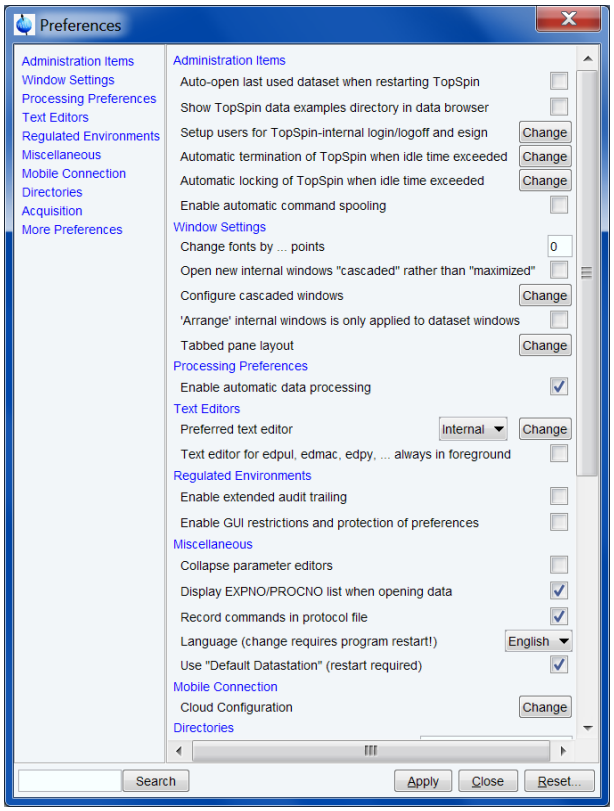
## 13.16 set

### NAME

set - Open the user preferences window.

### DESCRIPTION

The command **set** allows to set user preferences. It opens the dialog box shown:




In the left part of the dialog window, you find various categories of objects. Click the category of which you want to view/change certain objects. It will become highlighted and the corresponding objects will be displayed at the right part of the dialog box. Some objects can be changed by entering a value, others can be changed by clicking **Change** to the right of the object entry.

### INPUT AND OUTPUT FILE

*<home>/.topspin-<hostname>/prop*

*globals.prop* - ascii file containing User Interface settings

*view.prop* - colors fonts etc.

Where:

*<home>* is the users home directory

*<hostname>* is the hostname of the computer

## 13.17 setdef

### NAME

**setdef** - Configure acknowledgment of dialog windows.

### DESCRIPTION

With the command **setdef** the settings for automatic dialog acknowledgement can be changed.

### USAGE

**Display a help message**

- `xcpr setdef`
  Display a dialog with the available sub commands.
- `xcpr setdef <sub command>`
  Display a dialog with the available options for the specified sub command, where <sub command> is one of **ackn**, **beep**, **quest**, **stderr**, **stdout**.

**Sub command ackn**

This defines how standard dialog windows with a simple OK button are handled.

- `setdef ackn no`
  Program execution continues without acknowledgment.
- `setdef ackn ok`
  Program execution continues only after acknowledgment.
- `setdef ackn hide`
  Program execution continues without acknowledgment. In addition, the dialog window is not displayed.

**Sub command beep**

Obsolete. There is no action related to this sub command.

**Sub command quest**

This defines how question windows with an OK and a CANCEL button are handled.

- `setdef quest ok`
  Program execution continues without acknowledgment assuming OK.
- `setdef quest can`
  Program execution continues without acknowledgment assuming CANCEL.
- `setdef quest no`
  Program execution continues only after acknowledgment.

**Sub command stderr**

This defines whether stderr (error messages from program execution) are written to a file.

- `setdef stderr off`
  Program error messages are not written to a file.
- `setdef stderr on`
  Program error messages are written to a file.

**Sub command stdout**

This defines whether **stdout** (standard messages from program execution) are written to a file.

- `setdef stdout off`
  Program standard messages are not written to a file.
- `setdef stdout on`
  Program standard messages are written to a file.

## DEFAULTS

At TopSpin start all of these settings are reset to their defaults, namely

- `setdef ackn ok`
- `setdef quest no`
- `setdef stderr on`
- `setdef stdout on`

## OUTPUT FILES

If program standard and/or error messages are written to a file, the files are in the directory

*<tshome>/prog/curdir/<user>*

and have the names

*stdout.<pid>* - standard TopSpin output file

*stderr.<pid>* - standard TopSpin error file

**PROGRAMMING GUIDE**

In AU programs the current state of each sub command option can be queried with

int result = CPR_exec("setdef <sub command> ?");

The result is

- sub command ackn: 'n' for no, 'o' for ok
- sub command quest: 0 for ok, 1 for can, -1 for no
- sub command stdout and stderr: 'y' for on, 'n' for off
- sub command beep: 'y' for yes, 'n' for no.

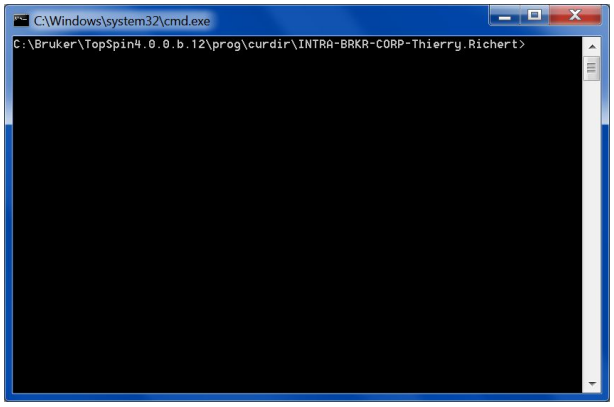
## 13.18 shell

**NAME**

shell - Open a Windows Command Prompt or Linux Shell

**DESCRIPTION**

The command **shell** opens a Command Prompt (under Windows) or a shell (under Linux).




## 13.19 start_rest_interface, stop_rest_interface

**NAME**

start_rest_interface, stop_rest_interface

**DESCRIPTION**

The command **start_rest_interface** starts a RESTful service, which allows to access Topspin from any other application. For the sake of security, an administrator password is required.

The command allows to specify the port number on which the service is running **start_rest_interface [-p portNumber]**

The default port is 3080. The service is limited to the local machine (localhost, 127.0.0.1).

Following features are available:

- Send command to Topspin
- Read NMR datasets (currently limited to 1D und 2D)
- Read peak lists
- Read integration regions
- Read and write data set parameters

Command **stop_rest_interface** stops the RESTful service immediately.

## 13.20 swin

### NAME

swin - Swap the position and geometry of two data windows.

### DESCRIPTION

The command **swin** swaps the position of two data windows. If the layout contains exactly two data windows, **swin** simple swaps their position and geometry. If the layout contains more than two data windows, **swin** allows you to swap the currently selected (active) data window with any of the other data windows. The latter can be selected from a list.

**swin** is typically used after reading a window layout with more than one data window.

### SEE ALSO

*Newwin, nextwin, close, closeall [▶ 368]*

# 14 TopSpin Audit Trails

This chapter describes commands which are related to TopSpin audit trail. The audit trail contains a record of all acquisition and processing activities, data checksums and electronic signatures.

Please note, that the user management and electronic signatures are available only in the Topspin GxP version.

## 14.1 audit, auditcheck

### NAME

audit - Open audit trail dialog box (nD)

auditcheck - Check data consistency (nD)

### DESCRIPTION

The command **audit** opens the audit trail dialog box:


This dialog box has several options, each of which selects a certain command for execution.

#### View audit trail of the processed data

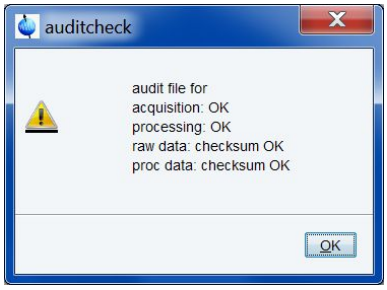
This option selects the command **audit proc** for execution. It shows the processing audit trail file *auditp.txt*. This file is created by the processing command that creates the processed data, e.g. **em**. Any processing command that modifies/updates the processed data, e.g. **ft**, makes an additional entry. Furthermore, any command that changes one or more processing status parameters makes an additional entry.

### View audit trail of the acquisition data

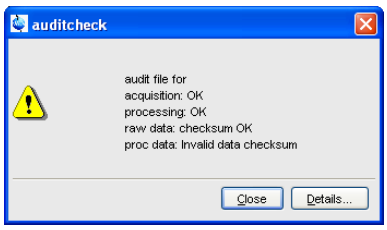
This option selects the command **audit acqu** for execution. It shows the acquisition audit trail file *audita.txt*. This file is created by the acquisition command that creates the raw data, e.g. **zg**. Any acquisition command that modifies/updates the raw data, e.g. **go**, makes an additional entry. Furthermore, any command that changes one or more acquisition status parameters makes an additional entry.

### Verify audit trails

This option selects the command **audit check** for execution. It performs an audit trail check, i.e. a data consistency check. If both raw and processed data are consistent, you will get the following message:




If the data have been manipulated, e.g. with third party software or by changing certain status parameters (e.g. SI), the checksum will be inconsistent. The following figure shows the message for inconsistent processed data.




### Add a comment to audit trail

This option selects the command **audit com** for execution. It allows you to add a comment to one of the audit trail files (raw or processed).

Each audit trail file entry contains the following elements:

- **Number**: The entry number (1, 2, 3,...).
- **When**: Starting date and time of the command.
- **Who**: User who starts the command (the user that started Topspin).
- **Where**: Location where the command started (the computer host name).
- **Version**: The TopSpin version which performed the acquisition or processing.
- **What**: Command and associated parameters, e.g. 

The last line of the file is a checksum which looks like:

$$ 24 EB 5D 82 76 AD F2 2B 7E D2 A1 35 7B B5 C4 D5

The command **auditcheck** uses this line for the consistency check.

### INPUT FILES

*<dir>/data/<user>/nmr/<name>/<expno>/*

*audita.txt* - acquisition audit trail

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*auditp.txt* - processing audit trail

Note that these are also the output files for **audit com**.

**SEE ALSO**

*gdcheck [▶ 377]*

## 14.2 gdcheck

**NAME**

gdcheck - Generate data checksum

**DESCRIPTION**

The command **gdcheck** generates a data checksum. It updates the audit trail files. It takes one argument and can be used as follows:

- **gdcheck**: Makes the processing audit trail consistent.
- **gdcheck raw**: Make the acquisition audit trail consistent.

**gdcheck** is, for example, required if a data set has been manipulated with third party software. In that case the audit trail would be inconsistent, i.e. the command **auditcheck** would report an inconsistency error. **gdcheck** updates the audit trail file with a new data checksum and adds the entry:

*Unknown data manipulation detected.*

After this, **auditcheck** would report:

*Unknown data manipulation.*

For 2D and 3D data, **gdcheck** adds a data checksum. For 1D data, a data checksum is automatically created by processing commands. In 2D and 3D, however, processing commands do not create a data checksum because this would be too time consuming. If it is required **gdcheck** allows you to create it.

**INPUT AND OUTPUT FILES**

*<dir>/data/<user>/nmr/<name>/<expno>/*

*audita.txt* - Acquisition audit trail.

*<dir>/data/<user>/nmr/<name>/<expno>/pdata/<procno>/*

*auditp.txt* - Processing audit trail.

**USAGE IN AU PROGRAMS**

GDCHECK

GDCHECK_RAW: Executes the command **gdcheck raw**.

AUDITCOMMENTA("user comment"): Adds a user comment to the *audita.txt* file.

AUDITCOMMENTP("user comment"): Adds a user comment to the *auditp.txt* file.

**SEE ALSO**

*audit, auditcheck commanda [▶ 375]*

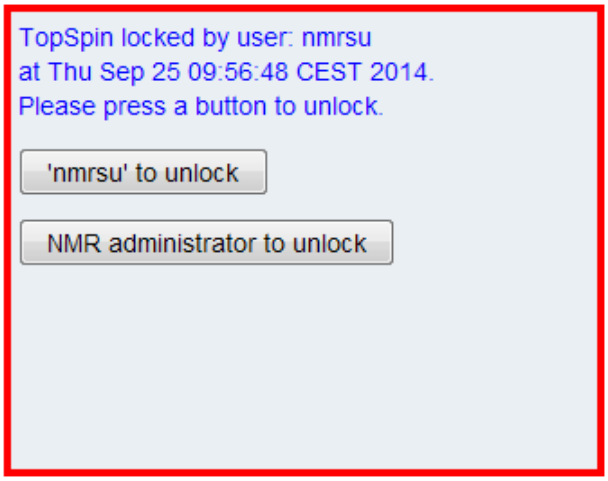
## 14.3 lockgui

### NAME

lockgui - Lock the TopSpin interface.

### DESCRIPTION

The command **lockgui** allows to logoff the internal user. It opens the dialog shown:




This indicates the locked status and offers buttons to unlock. Note that only the current internal user and the NMR Administrator can unlock the interface.

The command can also be started as follows:

Click **Manage | Security | Lock TopSpin for Other Users**.

### INPUT FILES

*<tshome>/conf/*

*topspin-users.prop* - TopSpin users properties file.

### SEE ALSO

uadmin, esign, chpwd, login, logoff

# 15 Contact

**Manufacturer**

Bruker BioSpin GmbH
Rudolf-Plank-Str. 23
D-76275 Ettlingen
Germany

Contact form: *https://www.bruker.com/en/services/service/magnetic-resonance.html*
Website: *http://www.bruker.com*
WEEE DE43181702

**Bruker BioSpin Hotlines**

Contact our Bruker BioSpin service centers.

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

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

*https://www.bruker.com/en/services/support.html*

# Index

## Symbols



## A



## B

## C



## D

## E



## F



## G



## H

## I



## J



## K



## L



## M

## Q



## R



## S

## T



## U



## V



## W

## X



## Z

**Bruker Corporation**

nmr-support@bruker.de
www.bruker.com

Part No: H9776SA4