

VNMR Command and Parameter Reference

*Varian NMR Spectrometer Systems
With VNMR 6.1C Software
Pub. No. 01-999164-00, Rev. A0800*



VARIAN

VNMR Command and Parameter Reference
Varian NMR Spectrometer Systems
With VNMR 6.1C Software
Pub. No. 01-999164-00, Rev. A0800

Revision history:
A0800 – Initial release for VNMR 6.1C software

Applicability of manual:
UNITYINOVA, *MERCURY VxWorks Powered (MERCURY-VX)* throughout this manual),
MERCURY, *UNITYplus*, *GEMINI 2000*, *UNITY*, and *VXR-S*
NMR spectrometer systems with VNMR 6.1C software installed

Technical contributors: Frits Vosman, Dan Iverson, Matt Howitt, Hung Lin,
Bayard Fetler, Phil Hornung, He Liu, Steve Cheatham, Rolf Kyburz, Chris Price,
Peter Sandor, Sukumar Subramaniam
Technical writers: Michael Carlisle, Mike Miller, Everett Schreiber
Technical editor: Dan Steele

Copyright ©2000 by Varian, Inc.
3120 Hansen Way, Palo Alto, California 94304
<http://www.varianinc.com>
All rights reserved. Printed in the United States.

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

UNITYINOVA, *MERCURY*, Gemini, *GEMINI 2000*, *UNITYplus*, *UNITY*, *VXR*, *XL*, *VNMR*, *VnmrS*, *VnmrX*, *VnmrI*, *VnmrV*, *VnmrSGI*, *MAGICAL II*, *AutoLock*, *AutoShim*, *AutoPhase*, *limNET*, *ASM*, and *SMS* are registered trademarks or trademarks of Varian, Inc. *Sun*, *Solaris*, *CDE*, *Suninstall*, *Ultra*, *SPARC*, *SPARCstation*, *SunCD*, and *NFS* are registered trademarks or trademarks of Sun Microsystems, Inc. and *SPARC International*. *Oxford* is a registered trademark of Oxford Instruments LTD. *Ethernet* is a registered trademark of Xerox Corporation. *VxWORKS* and *VxWORKS POWERED* are registered trademarks of WindRiver Inc. Other product names in this document are registered trademarks or trademarks of their respective holders.

Table of Contents

Introduction	33
---------------------------	-----------

A

aa	Abort acquisition with error (C).....	35
abort	Terminate action of calling macro and all higher macros (C).....	35
abortallacqs	Reset acquisition computer in a drastic situation (C).....	35
abortoff	Terminate normal functioning of abort in a macro (C).....	35
aborton	Restore normal functioning of abort in a macro (C).....	36
abs	Find absolute value of a number (C).....	36
AC1-AC9	Automated calibration (obsolete).....	36
AC1S-AC11S	Autocalibration macros (M).....	36
ACbackup	Make backup copy of current probe file (M).....	36
ACreport	Print copy of probe file after autocalibration (M).....	37
acos	Find arc cosine of number (C).....	37
acosy	Automatic analysis of COSY data (C).....	37
acosyold	Automatic analysis of COSY data, old algorithm (C).....	37
acqdisp	Display message on the acquisition status line (C).....	38
acqi	Interactive acquisition display process (C).....	38
acqmeter	Open Acqmeter window (M).....	39
Acqmeter	Open Acqmeter window (U).....	40
acqstat	Open Acquisition Status window (M).....	40
Acqstat	Open Acquisition Status window (U).....	41
acqstatus	Acquisition status (P).....	42
add	Add current FID to add/subtract experiment (C).....	44
addfids	Add a series of FIDs together (M).....	45
addi	Start interactive add/subtract mode (C).....	45
addnucleus	Add new nucleus to existing probe file (M).....	46
addpar	Add selected parameters to current experiment (M).....	46
addparams	Add parameter to current probe file (M).....	48
addprobe	Create new probe directory and probe file (M).....	48
adept	Automatic DEPT analysis and spectrum editing (C).....	49
addrcvrs	Combine data from multiple receivers (M).....	49
aexmpl	Automatic plot of spectral expansion (M).....	49
ai	Select absolute-intensity mode (C).....	50
aig	Absolute-intensity group (P).....	50
alfa	Set alfa delay before acquisition (P).....	50
alock	Automatic lock control (P).....	51
ampmode	Independent control of amplifier mode (P).....	51
amptype	Amplifier type (P).....	52
analyze	Generalized curve fitting (C).....	53
ap	Print out "all" parameters (C).....	55
ap	"All" parameters display control (P).....	55
apa	Plot parameters automatically (M).....	55
aph	Automatic phase adjustment of spectra (C).....	55
aph0	Automatic phase of zero-order term (C).....	56
aphb	Auto phasing for Bruker data (C).....	56
aphx	Perform optimized automatic phasing (M).....	56
apinterface	AP Interface board type (P).....	57
appmode	Application mode (P).....	57
apt	Set up parameters for APT pulse sequence (M).....	58

APT	Change parameters for APT experiment (M).....	58
aptaph	Automatic processing for APT spectra (M)	58
arccos	Calculate arc cosine of real number (M).....	58
arcsin	Calculate arc sine of real number (M).....	59
arctan	Calculate arc tangent of real number (M)	59
array	Easy entry of linearly spaced array values (M).....	60
array	Parameter order and precedence (P)	60
arraydim	Dimension of experiment (P)	61
asin	Find arc sine of number (C)	61
asize	Make plot resolution along f_1 and f_2 the same (M).....	61
assign	Assign transitions to experimental lines (M)	61
at	Acquisition time (P)	62
atan	Find arc tangent of a number (C).....	62
atan2	Find arc tangent of two numbers (C)	63
atext	Append string to current experiment text file (M)	63
attens	Fast attenuators present (P)	63
au	Submit experiment to acquisition and process data (M).....	64
AuC	Get parameters for carbon 1D experiment in <i>GLIDE</i> (M).....	65
AuCALch3i	Set up autocalibration with CH ₃ I sample (M)	65
AuCALch3i1	Get autocalibration with CH ₃ I sample (M).....	65
AuCALch3i2	Autocalibration macros with CH ₃ I sample (M).....	65
AuCALch3i3	Autocalibration macros with CH ₃ I sample (M).....	65
AuCALch3oh	Set up autocalibration with Autotest sample (M).....	65
AuCALch3oh1	Get autocalibration with Autotest sample (M).....	65
AuCALch3oh2	Autocalibration macros with Autotest sample (M)	66
Aucalibz0	Automatic Hz to DAC calibration for Z0 (M)	66
AuCdec_n	Carbon decoupler calibration macros (M)	66
AuCdec	Carbon decoupler calibration macro (M).....	66
AuCDEPT	Get parameters for carbon and DEPT experiments in <i>GLIDE</i> (M)	66
AuCexp	Get parameters for ¹³ C & ¹³ C-detected experiments in <i>GLIDE</i> (M).....	66
AuCgrad	Carbon/proton gradient ratio calibration macro (M).....	67
AuCobs	Carbon observe calibration macro (M)	67
audiofilter	Audio filter board type (P)	67
AuF	Get parameters for fluorine 1D experiment in <i>GLIDE</i> (M)	67
Aufindz0	Automatic adjustment of Z0 (M)	67
Augcal	Probe gcal calibration macro (M)	68
Augmap	Automated gradient map generation (M).....	68
Augmapz0	Automatic lock gradient map generation and z0 calibration (M)	68
AuH	Get parameters for proton 1D experiment in <i>GLIDE</i> (M)	68
AuH4nuc	Set up parameters for selectable 4nuc (HCPF) experiment (M).....	68
AuHCOSY	Set up parameters for ¹ H and COSY experiments in <i>GLIDE</i> (M).....	68
AuHdec	Proton decoupler calibration (M).....	69
AuHexp	Get parameters for ¹ H & ¹ H-detected experiments in <i>GLIDE</i> (M)	69
AuHobs	Proton observe calibration macro (M)	69
AuHsel	Get parameters for ¹ H and ¹ H-detected experiments in <i>GLIDE</i> (M).....	69
Aumakegmap	Auto lock gradient map generation (M).....	69
AuNuc	Get parameters for a given nucleus (M).....	69
AuP	Get parameters for phosphorus 1D experiment in <i>GLIDE</i> (M)	70
auto	Prepare for an automation run (C)	70
auto	Automation mode active (P)	70
AutoAddEXP	Add selected experiment to <i>GLIDE</i> chain (M)	70
auto_au	Controlling macro for automation (M)	71
Autobackup	Back up current probe file (M).....	71
Autocalnext	Run next item in calibration chain (M).....	71
Autocalpar	Add <i>GLIDE</i> calibration parameters (M)	72
Autocalsave	Save current item in the calibration directory (M).....	72
AutoCheck	Check for FID file (M)	72
Autoclrexp	Clean up current experiment (M).....	72
AutoDelCAL	Delete selected calibration routine from <i>GLIDE</i> chain (M).....	72

AutoDelEXP	Delete selected experiment from <i>GLIDE</i> chain (M)	72
autodept	Automated complete analysis of DEPT data (M)	73
autodir	Automation directory absolute path (P)	73
Autoexplist	Display current <i>GLIDE</i> selection (M).....	73
autogo	Start automation run (C)	73
AutoLIST	Run chained experiments (M).....	74
autolist	Set up and start chained acquisition (M).....	74
Automacrodir	Create directory to save macros in <i>GLIDE</i> run (M).....	75
Automkdir	Create directory to save data in <i>GLIDE</i> run (M).....	75
autoname	Create path for data storage (C)	75
autoname	Prefix for automation data file (P).....	75
Autoplot2D	Check for <i>GLIDE</i> -selected plot options (M).....	77
Autopsgset	Set up parameters for various experiments (M)	77
autora	Resume suspended automation run (C)	77
Autormmac	Delete macro set from curexp+ ' /macdir ' (M)	77
autosa	Suspend current automation run (C)	78
autoscale	Resume autoscaling after limits set by scalelimits macro (M)	78
Autosetgpar	Add <i>GLIDE</i> parameters to current parameter sets (M).....	78
Autosetwexp	Create AutoLIST from experiments list (M).....	78
autostack	Automatic stacking for processing and plotting arrays (M).....	78
AutoStrtfid	Recall stored FID (M)	79
AutoStrtpar	Recall stored parameters (M).....	79
autotest	Open Auto Test Window (C).....	79
autotime	Displays approximate time for automation (M).....	79
Autowrmac	Write a string to a macro in curexp+ ' /macdir ' (M)	79
av	Set abs. value mode in directly detected dimension (C)	79
av1	Set abs. value mode in 1st indirectly detected dimension (C)	80
av2	Set abs. value mode in 2nd indirectly detected dimension (C)	80
averag	Calculate average and standard deviation of input (C)	81
awc	Additive weighting const. in directly detected dimension (P)	81
awc1	Additive weighting const. in 1st indirectly detected dimension (P)	82
awc2	Additive weighting const. in 2nd indirectly detected dimension (P)	82
axis	Provide axis labels and scaling factors (C)	82
axis	Axis label for displays and plots (P)	82
axisf	Axis label for FID displays and plots (P).....	83
B		
B0	Magnet main static field (P)	85
bandinfo	Shaped pulse information for calibration (M).....	85
banner	Display message with large characters (C)	85
bc	1D and 2D baseline correction (C).....	86
bc2d	2D baseline correction (obsolete).....	87
beepoff	Turn beeper off (C).....	87
beepon	Turn beeper on (C)	87
binom	Set up parameters for BINOM pulse sequence (M).....	87
bootup	Macro executed automatically when VNMR activated (M)	88
boresize	Magnet bore size (P)	88
box	Draw a box on a plotter or graphics display (C)	88
boxes	Draw boxes selected by the mark command (M).....	89
bpa	Plot boxed parameters (M).....	89
br24	Set up parameters for BR24 pulse sequence (M).....	90
browser	Start Image Browser application (U)	90
bs	Block size (P)	91
btune	Tune broadband channel on MERCURY-Series and GEMINI 2000 (M).....	91
C		
c13	Automated carbon acquisition (M)	92
c13p	Process 1D carbon spectra (M)	92

calcdim	Calculate dimension of experiment (C)	92
calfa	Recalculate alfa so that first-order phase is zero (M).....	93
calibflag	Correct systematic errors in DOSY experiments (P)	93
calibrate	Start a dialog for autocalibration routines (M).....	93
capt	Automated carbon and APT acquisition (M)	93
CARBON	Set up parameters for carbon spectrum (M).....	94
cat	Display one or more text files in text window (C)	94
cattn	Coarse attenuator type (P).....	94
cd	Change working directory (C)	94
cdc	Cancel drift correction (C)	95
cdept	Automated carbon and DEPT acquisition (M).....	95
celem	Completed FID elements (P).....	95
center	Set display limits for center of screen (C).....	96
centersw	Move cursor to center of spectrum (M)	96
centersw1	Move cursor to center of spectrum in 1st indirect dimension (M).....	96
centersw2	Move cursor to center of spectrum in 2nd indirect dimension (M)	96
cexp	Create a VNMR experiment (M).....	97
cf	Current FID (P)	97
cfpmult	Calculate first-point multiplier for 2D experiments (M).....	97
change	Submit a change sample experiment to acquisition (M).....	98
cla	Clear all line assignments (M)	98
cla	Calculated transition number (P)	98
clamp	Calculated transition amplitude (P).....	99
cleanexp	Remove old files and directories from an experiment (M)	99
clear	Clear a window (C)	99
cleardosy	Delete temporarily saved data in current subexperiment (M).....	100
clfreq	Calculated transition frequency (P).....	100
clindex	Index of experimental frequency of a transition (P)	100
clradd	Clear add/subtract experiment (C)	100
color	Select plotting colors from a graphical interface (M)	100
combiplate	View a color map for visual analysis of VAST microtiter plate (U)	101
combishow	Display regions as red, green, and blue in CombiPlate window (M).....	101
compressfid	Compress double-precision VNMR FID data (M,U).....	101
config	Display current configuration and possibly change it (M).....	102
confirm	Confirm message using the mouse (C)	107
Console	System console type (P).....	107
contact_time	MAS cross-polarization spin-lock contact time (M).....	108
conv2ta	Convert imaging 3D transform to absolute value (U).....	108
convert	Convert data set from a VXR-style system (M,U)	109
convertbru	Convert Bruker data (M,U)	109
copy	Copy a file (C).....	112
cos	Find cosine value of an angle (C)	112
cosy	Set up parameters to a COSY pulse sequence (M)	112
COSY	Change parameters for COSY experiment (M)	112
cosyps	Set up parameters for phase-sensitive COSY pulse sequence (M)	113
cp	Copy a file (C).....	113
cp	Cycle phase (P)	113
cpmgt2	Set up parameters for CPMGT2 pulse sequence (M)	113
cpos_cvt	Convert data set from a VXR-style system (M,U)	114
cptmp	Copy experiment data into experiment subfile (M)	114
cpx	Create pbox shape file (M).....	114
cr	Cursor position in directly detected dimension (P).....	115
cr1	Cursor position in 1st indirectly detected dimension (P).....	115
cr2	Cursor position in 2nd indirectly detected dimension (P).....	115
crcom	Create user macro without using text editor (M)	115
create	Create new parameter in a parameter tree (C).....	115
creategtable	Generate system gradient table (M)	117
crf	Current time-domain cursor position (P)	117
crl	Clear reference line in directly detected dimension (M).....	117

cr11	Clear reference line in 1st indirectly detected dimension (M).....	118
cr12	Clear reference line in 2nd indirectly detected dimension (M).....	118
crmode	Current state of the cursors in df, ds, or dconi programs (P)	118
crof2	Recalculate rof2 so that $lp = 0$ (M)	118
ct	Completed transients (P)	119
ctext	Clear the text of the current experiment (C)	119
ctune	Tune carbon channel on $^1\text{H}/^{13}\text{C}$ GEMINI 2000 (M)	119
curecc	Name of eddy current compensation file (P).....	120
curexp	Current experiment directory (P)	120
curscan	Scan currently in progress (P).....	120
curwin	Current window (P).....	120
CustomQ	Set up a custom queue in automation (M)	120
cutoff	Data truncation limit (P)	121
cyclenoe	Set up parameters for CYCLENOE pulse sequence (M).....	121
cylbr24	Set up parameters for cycled BR24 pulse sequence (M)	121
cylmrev	Set up parameters for cycled MREV8 pulse sequence (M)	121
cz	Clear integral reset points (C)	121
D		
d0	Overhead delay between FIDs (P)	123
d1	First delay (P).....	123
d2	Incremented delay in 1st indirectly detected dimension (P).....	124
d2pul	Set up parameters for D2PUL pulse sequence (M).....	124
d3	Incremented delay for 2nd indirectly detected dimension (P)	124
d4	Incremented delay for 3rd indirectly detected dimension (P).....	125
DAC_to_G	Store gradient calibration value in DOSY sequences (P)	125
da	Display acquisition parameter arrays (C).....	126
daslp	Increment for t_1 dependent first-order phase correction (P).....	126
date	Date (P)	126
daxis	Display horizontal LC axis (M)	126
Dbppste	Set up parameters for Dbppste pulse sequence (M).....	127
Dbppsteinept	Set up parameters for Dbppsteinept pulse sequence (M).....	127
dc	Calculate spectral drift correction (C).....	127
dc2d	Apply drift correction to 2D spectra (C).....	127
dcg	Drift correction group (P)	128
dcon	Display noninteractive color intensity map (C)	128
dconi	Interactive 2D data display (C)	129
dconi	Control display selection for the dconi program (P).....	131
dconn	Display color intensity map without screen erase (C)	131
dcrmv	Remove dc offsets from FIDs in special cases (P).....	132
ddf	Display data file in current experiment (C).....	132
ddff	Display FID file in current experiment (C)	132
ddfp	Display phase file in current experiment (C)	132
ddif	Synthesize and show DOSY plot (C).....	133
dds	Default display (M)	133
dds_seqfil	Sequence-specific default display (M)	133
debug	Trace order of macro and command execution (C).....	134
decfrq	Interrogate or set first decoupler frequency (obsolete)	134
dec2frq	Interrogate or set second decoupler frequency (obsolete).....	134
dec3frq	Interrogate or set third decoupler frequency (obsolete)	134
decomp	Decompose a VXR-style directory (M)	134
def_osfilt	Default value of osfilt parameter (P).....	135
defaultdir	Default directory for Files menu system (P).....	135
delcom	Delete a user macro (M).....	135
delete	Delete a file, parameter directory, or FID directory (C).....	135
delexp	Delete an experiment (M)	136
dels	Delete spectra from T_1 or T_2 analysis (C)	136
delta	Cursor difference in directly detected dimension (P).....	137

delta1	Cursor difference in 1st indirectly detected dimension (P).....	137
delta2	Cursor difference in 2nd indirectly detected dimension (P)	137
deltaf	Difference of two time-domain cursors (P).....	137
dept	Set up parameters for DEPT pulse sequence (M)	137
DEPT	Change parameters for DEPT experiment (M)	138
deptgl	Set up parameters for DEPTGL pulse sequence (M).....	138
deptproc	Process array of DEPT spectra (M)	138
destroy	Destroy a parameter (C)	138
destroygroup	Destroy parameters of a group in a tree (C).....	139
df	Display a single FID (C)	139
df2d	Display FIDs of 2D experiment (C).....	140
df2dn	Display FIDs of 2D experiment without screen erase (obsolete)	140
dfid	Display a single FID (C)	140
dfmode	Current state of display of imaginary part of a FID (P)	140
dfrq	Transmitter frequency of first decoupler (P).....	141
dfrq2	Transmitter frequency of second decoupler (P)	141
dfrq3	Transmitter frequency of third decoupler (P).....	141
dfrq4	Transmitter frequency of fourth decoupler (P)	141
dfs	Display stacked FIDs (C).....	142
dfsa	Display stacked FIDs automatically (C)	142
dfsan	Display stacked FIDs automatically without screen erase (C).....	143
dfsh	Display stacked FIDs horizontally (C).....	143
dfshn	Display stacked FIDs horizontally without screen erase (C).....	143
dfsn	Display stacked FIDs without screen erase (C)	144
dfww	Display FIDs in whitewash mode (C).....	144
dg	Display group of acquisition/processing parameters (C).....	144
dg	Control dg parameter group display (P).....	145
dg1	Display group of display parameters (M)	145
dg1	Control dg1 parameter group display (P).....	145
dg2	Display group of 3rd and 4th rf channel/3D parameters (M).....	145
dg2	Control dg2 parameter group display (P).....	145
dga	Display group of spin simulation parameters (M)	146
DgcsteSL	Set up parameters for DgcsteSL pulse sequence (M)	146
Dgcstecosy	Set up parameters for Dgcstecosy pulse sequence (M).....	146
Dgcstehmqc	Set up parameters for Dgcstehmqc pulse sequence (M)	146
dglc	Display group of LC-NMR parameters (M)	147
dglc	Control dglc parameter group display (P).....	147
dglp	Display group of linear prediction parameters (M).....	147
dgm	Display menu to view parameter screens (C).....	147
dgs	Display group of shims and automation parameters (M).....	147
dgs	Control dgs parameter group display (P)	147
dhp	Decoupler high-power control with class C amplifier (P).....	148
dialog	Display a dialog box from a macro (C).....	148
diffshims	Compare two sets of shims (M,U)	149
digfilt	Write digitally filtered FIDs to another experiment (M).....	149
dir	List files in directory (C)	149
disp3d	Display 3D data (U)	149
display	Display parameters and their attributes (C)	150
dla	Display spin simulation parameter arrays (M).....	151
dlalong	Long display of spin simulation parameter arrays (C).....	151
dli	Display list of integrals (C)	151
dlivast	Produce text file and process wells (M)	152
dll	Display listed line frequencies and intensities (C).....	152
dlni	Display list of normalized integrals (M)	153
dlp	Decoupler low-power control with class C amplifier (P).....	153
dm	Decoupler mode for first decoupler (P).....	154
dm2	Decoupler mode for second decoupler (P).....	154
dm3	Decoupler mode for third decoupler (P)	154
dm4	Decoupler mode for fourth decoupler (P)	155

dmf	Decoupler modulation frequency for first decoupler (P)	155
dmf2	Decoupler modulation frequency for second decoupler (P)	156
dmf3	Decoupler modulation frequency for third decoupler (P)	156
dmf4	Decoupler modulation frequency for fourth decoupler (P)	156
dmfadj	Adjust tip-angle resolution time for first decoupler (M)	156
dmf2adj	Adjust tip-angle resolution time for second decoupler (M)	157
dmf3adj	Adjust tip-angle resolution time for third decoupler (M)	157
dmf4adj	Adjust tip-angle resolution time for fourth decoupler (M)	158
dmg	Data display mode in directly detected dimension (P)	158
dmg1	Data display mode in 1st indirectly detected dimension (P)	159
dmg2	Data display mode in 2nd indirectly detected dimension (P)	159
dmgf	Absolute-value display of FID data or spectrum in acqi (P)	159
dmi	Display multiple images (M)	160
dmm	Decoupler modulation mode for first decoupler (P)	160
dmm2	Decoupler modulation mode for second decoupler (P)	161
dmm3	Decoupler modulation mode for third decoupler (P)	161
dmm4	Decoupler modulation mode for fourth decoupler (P)	161
dn	Nucleus for first decoupler (P)	162
dn2	Nucleus for second decoupler (P)	162
dn3	Nucleus for third decoupler (P)	162
dn4	Nucleus for fourth decoupler (P)	163
dnode	Display list of valid limNET nodes (M,U)	163
dnuc	Retrieve nucleus table parameters for first decoupler (obsolete)	163
dnuc2	Retrieve nucleus table parameters for second decoupler (obsolete)	163
dnuc3	Retrieve nucleus table parameters for third decoupler (obsolete)	163
doautodialog	Start a dialog window using def file (M)	163
dodialog	Start a dialog window with dialoglib file (M)	164
doexpdialog	Start a dialog window with glide/exp/experiment def file (M)	164
dof	Frequency offset for first decoupler (P)	164
dof2	Frequency offset for second decoupler (P)	164
dof3	Frequency offset for third decoupler (P)	164
dof4	Frequency offset for fourth decoupler (P)	165
Doneshot	Set up parameters for Doneshot pulse sequence (M)	165
dopardialog	Start a dialog with dialoglib/experiment def file (M)	165
do_pcsm	Calculate proton chemical shifts spectrum (C)	165
dosy	Process DOSY experiments (M)	166
dosyfrq	Larmor frequency of phase encoded nucleus in DOSY (P)	166
dosygamma	Gyromagnetic constant of phase encoded nucleus in DOSY (P)	166
dosytimecubed	Gyromagnetic constant of phase encoded nucleus in DOSY (P)	167
dot1	Set up a T_1 experiment (M)	167
dotflag	Display FID as connected dots (P)	167
downsamp	Downsampling factor applied after digital filtering (P)	168
dp	Double precision (P)	168
dpcon	Display plotted contours (C)	168
dpconn	Display plotted contours without screen erase (C)	169
dpf	Display peak frequencies over spectrum (C)	169
dpir	Display integral amplitudes below spectrum (C)	170
dpirn	Display normalized integral amplitudes below spectrum (M)	170
dpl	Default plot (M)	170
dpl_seqfil	Sequence-specific default plot (M)	170
dplane	Display a 3D plane (M)	171
dpr	Default process (M)	171
dpr_seqfil	Sequence-specific default process (M)	171
dprofile	Display pulse excitation profile (M)	172
dproj	Display a 3D plane projection (M)	172
dps	Display pulse sequence (C)	172
dpwr	Power level for first decoupler with linear amplifier (P)	173
dpwr2	Power level for second decoupler with linear amplifier (P)	174
dpwr3	Power level for third decoupler with linear amplifier (P)	174

dpwr4	Power level for fourth decoupler amplifier (P)	175
dpwrf	First decoupler fine power (P).....	175
dpwrf2	Second decoupler fine power (P)	176
dpwrf3	Third decoupler fine power (P)	176
dpwrm	First decoupler linear modulator power (P)	176
dpwrm2	Second decoupler linear modulator power (P).....	176
dpwrm3	Third decoupler linear modulator power (P).....	176
dqcosy	Set up parameters for double-quantum filtered COSY (M)	177
DQCOSY	Change parameters for DQCOSY experiment (M).....	177
draw	Draw line from current location to another location (C).....	177
drawslice	Display target slices (M)	178
drawvox	Display target voxels (M).....	178
dres	Measure linewidth and digital resolution (C).....	178
dres	Tip-angle resolution for first decoupler (P).....	179
dres2	Tip-angle resolution for second decoupler (P).....	179
dres3	Tip-angle resolution for third decoupler (P)	179
dres4	Tip-angle resolution for fourth decoupler (P)	179
ds	Display a spectrum (C)	180
ds2d	Display 2D spectra in whitewash mode (C).....	181
ds2dn	Display 2D spectra in whitewash mode without screen erase (C)	182
dscale	Display scale below spectrum or FID (C).....	182
dscoef	Digital filter coefficients for downsampling (P).....	182
dseq	Decoupler sequence for first decoupler (P).....	183
dseq2	Decoupler sequence for second decoupler (P)	183
dseq3	Decoupler sequence for third decoupler (P).....	183
dsfb	Digital filter bandwidth for downsampling (P)	183
dshape	Display pulse shape or modulation pattern (M).....	184
dshapef	Display last generated pulse shape (M)	184
dshapei	Display pulse shape or modulation pattern interactively (M)	184
dshim	Display a shim “method” string (M).....	185
dslice	Display multiple images from multislice data set (M).....	185
dsfsfrq	Bandpass filter offset for downsampling (P).....	185
dsn	Measure signal-to-noise (C)	186
dsnmax	Calculate maximum signal-to-noise (M)	186
dsp	Display calculated spectrum (C)	186
dsp	Type of DSP for data acquisition (P)	187
dsplanes	Display a series of 3D planes (M).....	188
dss	Display stacked spectra (C).....	189
dssa	Display stacked spectra automatically (C).....	190
dssan	Display stacked spectra automatically without erasing (C)	192
dssh	Display stacked spectra horizontally (C)	192
dsshn	Display stacked spectra horizontally without erasing (C).....	193
dssl	Label a display of stacked spectra (M)	193
dssn	Display stacked spectra without screen erase (C)	194
dsvast	Display VAST data in a stacked 1D-NMR matrix format (M)	194
dsvast2d	Display VAST data in a pseudo-2D format (M).....	195
dsww	Display spectra in whitewash mode (C).....	195
dtext	Display a text file in graphics window (M).....	195
dtrig	Delay to wait for another trigger or acquire a spectrum (P)	196
dtune	Tune lock channel on GEMINI 2000 (M).....	196
E		
e	Eject sample (M)	197
eaddr	Display Ethernet address (M,U).....	197
ecc	Set up parameters to get eddy current compensation data (M)	197
ecctabl	Put gcal value and ecc file into table (M).....	197
ecctool	Open eccTool window (M)	198
echo	Display strings and parameter values in text window (C).....	198

echo	Current echo index for transformed image (P)	198
eddyout	Data analysis of eddy current compensation (M)	198
eddysend	Update acquisition eddy current settings (M)	199
edit	Edit a file with user-selectable editor (M).....	199
eff_echo	Effective echo position in EPI experiments (P)	199
eject	Eject sample (M)	200
elist	Display directory on remote VXR-style system (M,U)	200
element	Current array index for transformed image (P).....	200
enter	Enter sample information for automation run (M,U).....	200
enterdialog	Start a dialog window using enterexp file (M).....	201
epift	Process and display image in EPI experiments (M).....	201
epiph	Generate phasemap file in EPI experiments (M)	201
epirs	Reverse spectral data in EPI experiments (C)	202
epirun	Collect, process, and display EPI data (M)	202
episet	Set up parameters for EPI experiments (M).....	202
episvib	Save EPI images in FDF for ImageBrowser (M).....	202
eread	Transfer file from remote source (M,U).....	203
ernst	Calculate the Ernst angle pulse (C).....	203
errlog	Display recent VNMR error messages (C)	203
errloglen	Number of lines in VNMR error message display (P)	204
ewrite	Transfer file to remote destination (M,U)	204
exec	Execute a VNMR command (C)	204
exists	Checks if parameter, file, or macro exists and file type (C)	204
exit	Call the vnmexit command (M).....	206
exp	Find exponential value of a number (C).....	206
expactive	Determine if experiment has active acquisition (C).....	206
expfit	Make least-squares fit to polynomial or exponential curve (U)	207
expl	Display exponential or polynomial curves (C)	209
expladd	Add another diffusion analysis to current display (M)	210
explib	Display experiment library (M)	210
explist	Display current experiment chain and approx. time for each (M)	210
explog	Display log file for experiment (M)	210
exptime	Display experiment time (C).....	211
F		
f	Set display parameters to full spectrum (C).....	212
f19	Automated fluorine acquisition (M).....	212
f19p	Process 1D fluorine spectra (M).....	212
f1coef	Coefficient to construct F1 interferogram (P)	213
f2coef	Coefficient to construct F2 interferogram (P)	213
fattn	Fine attenuator (P).....	213
fb	Filter bandwidth (P)	214
fbc	Apply baseline correction for each spectrum in an array (M)	215
fdfgluer	Make FDF file from header and data parts (U)	215
fdfsplit	Divide FDF file into header and data parts (U).....	216
fdm1	Set, write 1D FDM parameters, run FDM (M)	216
fiddc3d	3D time-domain dc correction (P).....	217
fiddle	Perform reference deconvolution (M).....	217
fiddled	Perform reference deconvolution subtracting alternate FIDs (C)	219
fiddleu	Perform reference deconvolution subtracting successive FIDs (C)	219
fiddle2d	Perform 2D reference deconvolution (C).....	219
fiddle2D	Perform 2D reference deconvolution (C).....	219
fiddle2dd	Perform 2D reference deconvolution subtracting alternate FIDs (C)	219
fiddle2Dd	Perform 2D reference deconvolution subtracting alternate FIDs (C)	219
fidpar	Add parameters for FID display in current experiment (M)	219
fifolpsize	FIFO loop size (P).....	220
fixgrd	Convert gauss/cm value to DAC (M)	220
file	File name of parameter set (P)	220

files	Interactively handle files (C).....	221
filesinfo	Return file information for files display (C).....	221
filter	Gaussian low-pass filter for image processing (M).....	221
filtfile	File of FIR digital filter coefficients (P).....	222
fitplot	Adjust plot parameters (M).....	222
fitspec	Perform spectrum deconvolution (C, U).....	223
fixpar	Correct parameter characteristics in experiment (M).....	224
fixparflow	224	
fixpar3rf	Create parameters for third rf channel (M).....	224
fixpar4rf	Create parameters for fourth rf channel (M).....	225
fixpar5rf	Create parameters for fifth rf channel (M).....	225
fixup	Adjust parameter values selected by setup macros (M).....	225
flashc	Convert compressed 2D data to standard 2D format (C).....	225
flip	Flip between graphics and text windows (C).....	227
flipflop	Set up parameters for FLIPFLOP pulse sequence (M).....	227
fliplist	Standard flip angle list (P).....	227
FLUORINE	Set up parameters for fluorine spectrum (M).....	228
flush	Write out data in VNMR memory (C).....	228
fn	Fourier number in directly detected dimension (P).....	228
fn1	Fourier number in 1st indirectly detected dimension (P).....	228
fn2	Fourier number in 2nd indirectly detected dimension (P).....	229
fn2D	Fourier number to build up 2D DOSY display in frequency domain (P).....	229
focus	Send keyboard focus to VNMR input window (C).....	229
foldcc	Fold INADEQUATE data about two-quantum axis (C).....	229
foldj	Fold J-resolved 2D spectrum about $f_1=0$ axis (C).....	230
foldt	Fold COSY-like spectrum along diagonal axis (C).....	230
fontselect	Open FontSelect window (C).....	230
format	Format a real number or convert a string for output (C).....	231
fp	Find peak heights or phases (C).....	231
fpmult	First point multiplier for np FID data (P).....	232
fpmult1	First point multiplier for ni interferogram data (P).....	232
fpmult2	First point multiplier for ni2 interferogram data (P).....	232
fr	Full recall of a display parameter set (M).....	233
fread	Read parameters from file and load them into a tree (C).....	233
fsave	Save parameters from a tree to a file (C).....	234
fsq	Frequency-shifted quadrature detection (P).....	234
ft	Fourier transform 1D data (C).....	234
ft1d	Fourier transform along f_2 dimension (C).....	236
ft1da	Fourier transform phase-sensitive data (M).....	237
ft1dac	Combine arrayed 2D FID matrices (M).....	237
ft2d	Fourier transform 2D data (C).....	238
ft2da	Fourier transform phase-sensitive data (M).....	240
ft2dac	Combine arrayed 2D FID matrices (M).....	242
ft3d	Perform a 3D Fourier transform on a 3D FID data set (M,U).....	242
full	Set display limits for a full screen (C).....	246
fullsq	Display largest square 2D display (M).....	246
fullt	Set display limits for a full screen with room for traces (C).....	246
G		
g2pul	Set up pulse sequence for gradient evaluation (M).....	248
ga	Submit experiment to acquisition and FT the result (M).....	248
gadm	Display <i>GLIDE</i> administration tool (C).....	249
gain	Receiver gain (P).....	249
gap	Find gap in the current spectrum (M).....	250
gaussian	Set up unshifted Gaussian window function (M).....	250
gcal	Gradient calibration constant (P).....	250
gcoil	Current gradient coil (P).....	251
gCOSY	Change parameters for gCOSY experiment (M).....	252

gcosy	Set up pulse sequence for gradient COSY (M).....	252
gcrush	Crusher gradient level (P).....	252
gdifff	Diffusion gradient level (P).....	252
get1d	Select a 1D experiment for processing (M).....	252
get2d	Select a 2D experiment for processing (M).....	253
getdim	Return dimensionality of experiment (M).....	253
getfile	Get information about directories and files (C).....	254
getgcal	Get gcal value from table (M).....	254
getl1	Get intensity and line frequency of line (C).....	255
getparam	Retrieve parameter from probe file (M).....	255
getplane	Extract planes from a 3D spectral data set (M).....	255
getreg	Get frequency limits of a specified region (C).....	256
getsn	Get signal-to-noise estimate of a spectrum (M).....	257
gettxt	Get text file from VNMR data file (C).....	257
getvalue	Get value of parameter in a tree (C).....	257
gf	Prepare parameters for FID/spectrum display in acqi (M).....	258
gf	Gaussian function in directly detected dimension (P).....	258
gf1	Gaussian function in 1st indirectly detected dimension (P).....	259
gf2	Gaussian function in 2nd indirectly detected dimension (P).....	259
gflow	Flow encoding gradient level (P).....	259
gfs	Gaussian shift const. in directly detected dimension (P).....	259
gfs1	Gaussian shift const. in 1st indirectly detected dimension (P).....	259
gfs2	Gaussian shift const. in 2nd indirectly detected dimension (P).....	260
gHMBC	Change parameters for gHMBC experiment (M).....	260
ghmqc	Set up a PFG HMQC pulse sequence (M).....	260
gHMQC	Set up parameters for gHMQC experiment (M).....	260
gHMQC15	Set up parameters for ¹⁵ N gHMQC experiment (M).....	260
gHMQC_d2	Set up parameters for ¹⁵ N gHMQC experiment using decoupler 2 (M).....	261
gHMQC_d213	Set up parameters for ¹³ C gHMQC experiment using decoupler 2 (M).....	261
ghmqcps	Set up a PFG HMQC phase-sensitive pulse sequence (M).....	261
ghmqctoxy	Change parameters for gHMQCCTOXY experiment (M).....	261
ghsqc	Set up a PFG HSQC pulse sequence (M).....	261
gHSQC	Set up parameters for gHSQC experiment (M).....	262
gHSQC15	Set up parameters for ¹⁵ N gHSQC experiment (M).....	262
gHSQC_d2	Set up parameters for ¹⁵ N gHSQC experiment using decoupler 2 (M).....	262
gHSQC_d213	Set up parameters for ¹³ C gHSQC experiment using decoupler 2 (M).....	262
gHSQCTOXY	Set up parameters for gHSQCTOXY experiment (M).....	262
gilson	Open the Gilson Control window (C).....	263
gin	Return current mouse position and button values (C).....	263
glide	Interactive windows for data acquisition and processing (C).....	263
globalauto	Automation directory name (P).....	264
glue	Create a pseudo-2D dataset (M).....	264
gmapshim	Start gradient autoshimming (M).....	264
gmapshim_au	Start acquisition with gradient shimming (M).....	265
gmapsys	Run gradient autoshimming, set parameters, map shims (M).....	265
gmapuser	Run gradient autoshimming and set parameters (obsolete).....	266
gmapz	Get parameters and files for gmapz pulse sequence (M).....	266
gmap_findtof	Gradient shimming flag to first find tof (P).....	266
gmap_z1z4	Gradient shimming flag to first shim z1-z4 (P).....	267
gmax	Maximum gradient strength (P).....	267
gmqcosy	Set up PFG absolute-value MQF COSY parameter set (M).....	267
gnoesy	Set up a PFG NOESY parameter set (M).....	267
go	Submit experiment to acquisition (M).....	268
go_	Pulse sequence setup macro called by go, ga, and au (M).....	269
gpat-gpat3	Gradient shape (P).....	269
gpe	Phase encoding gradient increment (P).....	269
gped	Phase encode dephasing gradient in the EPI sequence (P).....	270
gpemult	Phase encode gradient increment multiplier (P).....	270
gradaxis	Gradient axis (P).....	270

gradstepsz	Gradient step size (P)	270
gradtype	Gradients for X, Y, and Z axes (P)	270
graphis	Return the current graphics display status (C)	271
grayctr	Gray level window adjustment (P)	271
graysl	Gray level slope (contrast) adjustment (P)	272
grecovery	Eddy current testing (M)	272
grid	Draw a grid on a 2D display (M)	272
griserate	Gradient rise rate (P)	273
gro	Readout gradient strength (P)	273
groa	Readout gradient adjuster in EPI experiment (P)	273
gropat	Readout gradient shape (P)	273
gror	Read out compensation gradient (P)	273
grora	Readout dephasing gradient adjuster in EPI experiment (P)	274
groupcopy	Copy parameters of group from one tree to another (C)	274
gsh2pul	Set up parameters for shaped gradients tests (M)	274
gspoil	Spoiler gradient level (P)	275
gss	Slice selection gradient strength (P)	275
gssf	Slice selection fractional refocusing (P)	275
gsspat	Slice-select gradient shape (P)	275
gssr	Slice selection refocusing gradient (P)	275
gss2,gss3	Slice selection gradient level (P)	276
gtnnoesy	Set up a PFG TNNNOESY parameter set (M)	276
gtnroesy	Set up a PFG absolute-value ROESY parameter set (M)	276
gtotlimit	Gradient total limit (P)	276
gtrim	Trim gradient level (P)	276
gvx1-gvox3	Gradient strength for voxel selection (P)	276
gx, gy, gz	Gradient strength for X, Y, and Z gradients (P)	277
gxcal,gycal,gzcal	Gradient calibration constants (P)	277
gxmax,gymax,gzmax	Maximum gradient strength for each axis (P)	277
gzlvl	Number of z-axis shims used by gradient shimming (P)	278
gzsize	Number of z-axis shims used by gradient shimming (P)	278
gzwin	Spectral width percentage used for gradient shimming (P)	278

H

h1	Automated proton acquisition (M)	279
h1freq	Proton frequency of spectrometer (P)	279
h1p	Process 1D proton spectra (M)	279
h2cal	Calculate strength of the decoupler field (C)	280
halt	Abort acquisition with no error (C)	280
hc	Automated proton and carbon acquisition (M)	281
hcapt	Automated proton, carbon, and APT acquisition (M)	281
hcchtocsy	Set up parameters for HCCHTOCSY pulse sequence (M)	281
hccorr	Automated proton, carbon, and HETCOR acquisition (M)	282
hcdept	Automated proton, carbon, and DEPT acquisition (M)	282
hcosy	Automated proton and COSY acquisition (M)	282
hdofst	Proton homonuclear decoupler offset (P)	283
hdwshim	Hardware shimming (P)	283
hdwshimlist	List of shims for hardware shimming (P)	283
help	Display current help file (C)	284
helppath	Path to user's help directory (P)	284
het2dj	Set up parameters for HET2DJ pulse sequence (M)	284
HETCOR	Change parameters for HETCOR experiment (M)	284
hetcor	Set up parameters for HETCOR pulse sequence (M)	284
hetcorcp1	Set up parameters for solids HETCOR pulse sequence (M)	285
hetcorps	Set up parameters for HETCORPS pulse sequence (M)	285
hidecommand	Execute macro instead of command with same name (C)	285
HMBC	Change parameters for HMBC experiment (M)	285
hmqc	Set up parameters for HMQC pulse sequence (M)	286

HMQC	Set up parameters for HMQC experiment (M)	286
HMQC15	Set up parameters for ¹⁵ N HMQC experiment (M)	286
HMQC_d2	Set up parameters for ¹⁵ N HMQC experiment using decoupler 2 (M)	286
HMQC_d213	Set up parameters for ¹³ C HMQC experiment using decoupler 2 (M)	286
hmqcr	Set up parameters for HMQCR pulse sequence (M)	287
hmqctocsy	Set up parameters for HMQCTOCSY pulse sequence (M)	287
HMQCTOXY	Set up parameters for HMQCTOXY experiment (M)	287
HMQCTOXY15	Set up parameters for ¹⁵ N HMQCTOXY experiment (M)	287
HMQCTOXY_d2	Set up parameters for ¹⁵ N HMQCTOXY using decoupler 2 (M)	287
HMQCTOXY_d213	Set up parameters for ¹³ C HMQCTOXY using decoupler 2 (M)	287
hmqctoxy3d	Set up parameters for HMQC-TOCSY 3D pulse sequence (M)	288
ho	Horizontal offset (P)	288
hold	Post-trigger delay (P)	288
hom2dj	Set up parameters for HOM2DJ pulse sequence (M)	288
HOMODEC	Change parameters for HOMODEC experiment (M)	288
homdec	Proton homonuclear decoupler present (P)	289
homo	Homodecoupling control for first decoupler (P)	289
homo2	Homodecoupling control for second decoupler (P)	290
homo3	Homodecoupling control for third decoupler (P)	290
homo4	Homodecoupling control for fourth decoupler (P)	290
hoult	Set parameters alfa and rof2 according to Hoult (M)	290
hpa	Plot parameters on special preprinted chart paper (C)	291
hregions	Select integral regions in proton spectrum (M)	291
hs	Homospoil pulses (P)	291
hsqc	Set up parameters for HSQC pulse sequence (M)	291
HSQC	Set up parameters for HSQC experiment (M)	292
HSQC15	Set up parameters for ¹⁵ N HSQC experiment (M)	292
HSQC_d2	Set up parameters for ¹⁵ N HSQC experiment using decoupler 2 (M)	292
HSQC_d213	Set up parameters for ¹³ C HSQC experiment using decoupler 2 (M)	292
HSQCTOXY	Set up parameters for HSQCTOXY experiment (M)	292
HSQCTOXY15	Set up parameters for ¹⁵ N HSQCTOXY experiment (M)	293
HSQCTOXY_d2	Set up parameters for ¹⁵ N HSQCTOXY using decoupler 2 (M)	293
HSQCTOXY_d213	Set up parameters for ¹³ C HSQCTOXY using decoupler 2 (M)	293
hsqctoxySE	Set up parameters for HSQC-TOCSY 3D pulse sequence (M)	293
hsrotor	Display rotor speed for solids operation (P)	293
hst	Homospoil time (P)	294
htune	Tune proton channel on GEMINI 2000 (M)	294
hzmm	Scaling factor for plots (P)	294
hztomm	Convert locations from Hz or ppm to plotter units (C)	294
/		
i	Insert sample (M)	296
ihwinfo	Hardware status of ^{UNITY} INOVA console (U)	296
il	Interleave arrayed and 2D experiments (P)	296
ilfid	Interleave FIDs during data processing (C)	296
image	Display noninteractive gray scale image (M)	297
image	Control phase encoding gradient in EPI experiments (P)	297
imageprint	Plot noninteractive gray scale image (M)	298
imark	Annotate an image display (M)	298
imcalc	Calculate 2D phasefiles (M,U)	298
imcalci	Format arguments for imcalc macro (M)	299
imconi	Display 2D data in interactive grayscale mode (M)	300
imfit	Fit arrayed imaging data to T_1 or T_2 exponential data (M,U)	300
imprep	Set up rf pulses, imaging and voxel selection gradients (M)	301
in	Lock and spin interlock (P)	301
inadqt	Set up parameters for INADEQUATE pulse sequence (M)	301
index2	Projection or 3D plane index selected (P)	302
inept	Set up parameters for INEPT pulse sequence (M)	302

initialize_iterate	Set iterate string to contain relevant parameters (M)	302
input	Receive input from keyboard (C)	302
ins	Integral normalization scale (P)	303
ins2	2D volume value (P)	303
insref	Fourier number scaled value of an integral (P)	303
ins2ref	Fourier number scaled volume of a peak (P)	303
insert	Insert sample (M)	304
inset	Display an inset spectrum (C)	304
integ	Find largest integral in a specified region (C)	304
integrate	Automatically integrate 1D spectrum (M)	305
intmod	Integral display mode (P)	305
intvast	Produces a text file of integral regions (M)	305
iplan	Open interactive image planning tools (M)	305
io	Integral offset (P)	306
ir	Inversion recovery mode (P)	306
is	Integral scale (P)	306
isadj	Automatic integral scale adjustment (M)	306
isadj2	Automatic integral scale adjustment by powers of two (M)	307
iterate	Parameters to be iterated (P)	307
J		
jdesign	Start Plot Designer Program (M)	308
jexp	Join existing experiment (C)	308
jexp1-jexp9999	Join existing experiment and display new parameters (M)	308
jplot	Plot from Plot Designer program (C)	309
jplotscale	Scale plot parameters (M)	309
jplotunscale	Restore current experiment parameters (M)	309
jumpret	Set up parameters for JUMPRET pulse sequence (M)	310
jwin	Activate and record activity in current window (M)	310
K		
killft3d	Terminate any ft3d process started in an experiment (M,U)	311
killplot	Stop plot jobs and remove from plot queue (M)	311
killprint	Stop print jobs and remove from print queue (M)	312
kind	Kinetics analysis, decreasing intensity (M)	312
kinds	Kinetics analysis, decreasing intensity, short form (M)	312
kini	Kinetics analysis, increasing intensity (M)	312
kinis	Kinetics analysis, increasing intensity, short form (M)	313
L		
large	Use large graphics window (C)	314
lastlk	Last lock solvent used (P)	314
lastmenu	Menu to display when Return button is selected (P)	314
latch	Frequency synthesizer latching (P)	314
lb	Line broadening in directly detected dimension (P)	315
lb1	Line broadening in 1st indirectly detected dimension (P)	315
lb2	Line broadening in 2nd indirectly detected dimension (P)	315
lc1d	Pulse sequence for LC-NMR (M)	316
lcpard2d	Create 2D LC-NMR acquisition parameters (M)	316
lcpeak	Peak number (P)	316
lcplot	Plot LC-NMR data (M)	316
lcpsgset	Set up parameters for various LC-NMR pulse sequences (M)	317
lcset2d	General setup for 2D LC-NMR experiments (M)	317
left	Set display limits to left half of screen (C)	317
legrelay	Independent control of magnet leg relay (P)	318
length	Determine length of a string (C)	318
lf	List files in directory (C)	318

liamp	Amplitudes of integral reset points (P)	318
lifrq	Frequencies of integral reset points (P).....	319
listenoff	Disable receipt of messages from send2Vnmr (M).....	319
listenon	Enable receipt of messages from send2Vnmr (M).....	319
lkof	Track changes in lock frequency (P).....	319
ll2d	Automatic and interactive 2D peak picking (C).....	320
ll2dbackup	Copy current ll2d peak file to another file (M)	322
ll2dmode	Control display of peaks picked by ll2d (P).....	323
llamp	List of line amplitudes (P).....	323
llfrq	List of line frequencies (P).....	323
ln	Find natural logarithm of a number (C)	323
load	Load status of displayed shims (P)	324
loadcolors	Load colors for graphics window and plotters (M).....	324
loc	Location of sample in tray (P).....	324
location	Get coordinate information from an image display (M)	325
lock	Submit an Autolock experiment to acquisition (C).....	325
lockacqtc	Lock loop time constant during acquisition (P)	326
lockfreq	Lock frequency (P).....	326
lockgain	Lock gain (P).....	327
lockphase	Lock phase (P)	327
lockpower	Lock power (P).....	327
locktc	Lock time constant (P)	327
logate	Transmitter local oscillator gate (P)	328
lookup	Look up words and lines from a text file (C)	328
lp	First-order phase in directly detected dimension (P)	330
lp1	First-order phase in 1st indirectly detected dimension (P).....	330
lp2	First-order phase in 2nd indirectly detected dimension (P)	330
lpalg	LP algorithm in np dimension (P).....	330
lpalg1	LP algorithm in ni dimension (P).....	331
lpalg2	LP algorithm in ni2 dimension (P).....	331
lpe	Field of view size for phase-encode axis (P).....	332
lpe2	Field of view size for 2nd phase-encode axis (P).....	332
lpext	LP data extension in np dimension (P)	332
lpext1	LP data extension in ni dimension (P)	333
lpext2	LP data extension in ni2 dimension (P)	333
lpfilt	LP coefficients to calculate in np dimension (P).....	333
lpfilt1	LP coefficients to calculate in ni dimension (P).....	333
lpfilt2	LP coefficients to calculate in ni2 dimension (P).....	334
lpnupts	LP number of data points in np dimension (P)	334
lpnupts1	LP number of data points in ni dimension (P)	334
lpnupts2	LP number of data points in ni2 dimension (P)	334
lpopt	LP algorithm data extension in np dimension (P).....	335
lpopt1	LP algorithm data extension in ni dimension (P).....	335
lpopt2	LP algorithm data extension in ni2 dimension (P).....	336
lpprint	LP print output for np dimension (P)	336
lpprint1	LP print output for ni dimension (P).....	336
lpprint2	LP print output for ni2 dimension (P).....	337
lptrace	LP output spectrum in np dimension (P)	337
lptrace1	LP output spectrum in ni dimension (P)	337
lptrace2	LP output spectrum in ni2 dimension (P)	337
lro	Field of view size for readout axis (P)	338
ls	List files in directory (C).....	338
lsfid	Number of complex points to left-shift the np FID (P).....	338
lsfid1	Number of complex points to left-shift ni interferogram (P).....	339
lsfid2	Number of complex points to left-shift ni2 interferogram (P).....	339
lsfrq	Frequency shift of the fn spectrum (P).....	340
lsfrq1	Frequency shift of the fn1 spectrum (P).....	340
lsfrq2	Frequency shift of the fn2 spectrum (P).....	341
lvl	Zero-order baseline correction (P)	341

lvl	Control sensitivity of lvl and tlt adjustments (P)	341
lvl	Control sensitivity of lvl and tlt adjustments (P)	341
M		
maclibpath	Path to user's macro directory (P)	342
macro	Macro name (P)	342
macrocat	Display a user macro file in text window (C)	342
macrocp	Copy a user macro file (C)	342
macrodir	List user macro files (C)	343
macroedit	Edit a macro with user-selectable editor (M)	343
macrold	Load a macro into memory (C)	343
macrorm	Remove a user macro (C)	344
macrosyscat	Display a system macro file in text window (C)	344
macrosyscp	Copy a system macro to become a user macro (C)	345
macrosysdir	List system macros (C)	345
macrosysld	Load a system macro into memory (obsolete)	345
macrosysrm	Remove a system macro (C)	345
macrosysvi	Edit a system macro with the vi text editor (obsolete)	346
macrovi	Edit a user macro with the vi text editor (M)	346
make3dcoef	Make a 3D coefficients file from 2D coefficients (M)	346
makedosyparams Create parameters for DOSY processing (M)	347
makefid	Make a FID element using numeric text input (C)	347
makephf	Transform and save images as phasefiles (M)	348
makeslice	Synthesize 2D projection of 3D DOSY experiment (C)	348
man	Display online description of command or macro (M)	349
manualpath	Path to user's manual directory (P)	349
manvi	Edit online description of a command or macro (M)	349
mapwin	List of experiment numbers (P)	349
mark	Determine intensity of spectrum at a point (C)	350
masvt	Type of variable temperature system (P)	352
math	Fourier transform mathematics (obsolete)	352
maxpen	Maximum number of pens to use (P)	352
maxsw_loband	Maximum spectral width of Input board (P)	352
md	Move display parameters between experiments (C)	353
menu	Change status of menu system (C)	353
menulibpath	Path to user's menu directory (P)	353
menuvi	Edit a menu with vi text editor (M)	354
method	Autoshim method (P)	354
mf	Move FIDs between experiments (C)	354
mfblk	Copy FID block (C)	355
mfclose	Close memory map FID (C)	355
mfdata	Move FID data (C)	356
mfopen	Memory map open FID file (C)	356
mftrace	Move FID trace (C)	357
minsw	Reduce spectral width to minimum required (M)	358
mkdir	Create new directory (C)	358
mlabel	Menu label (P)	358
move	Move to an absolute location to start a line (C)	359
movedssw	Set downsampling parameters for selected spectral region (M)	359
moveossw	Set oversampling parameters for selected spectral region (M)	359
movepro	Move the imaging readout position (C)	360
movesw	Move spectral window according to cursors (M)	360
movetof	Move transmitter offset (M)	361
mp	Move parameters between experiments (C)	361
mqcosy	Set up parameters for MQCOSY pulse sequence (M)	361
mrev8	Set up parameters for MREV8 pulse sequence (M)	361
mrfb	Set the filter bandwidths for multiple receivers (P)	362
mrgain	Set the gain for multiple receivers (P)	362
mstat	Display memory usage statistics (C)	362

mstring	Menu string (P)	363
mv	Move and/or rename a file (C)	363
mxconst	Maximum scaling constant (P)	363
N		
n1,n2,n3	Name storage for macros (P).....	364
nactivercvrs	Return number of receivers currently active (M)	364
nD	Application dimension (P)	364
ne	Number of echoes to be acquired (P).....	364
newmenu	Select a menu without immediate activation (C)	365
newshm	Interactively create a shim method with options (M)	365
nextpl	Display the next 3D plane (M).....	366
nf	Number of FIDs (P)	366
ni	Number of increments in 1st indirectly detected dimension (P).....	366
ni2	Number of increments in 2nd indirectly detected dimension (P)	367
ni3	Number of increments in 3rd indirectly detected dimension (P).....	367
niter	Number of iterations (P)	367
nl	Position cursor at the nearest line (C)	367
nli	Find integral values (C).....	367
nlivast	Produces a text file of integral regions without a sum region (M).....	368
nlivast2	Produces a text file with normalized integral regions (M).....	368
nlivast3	Produces a text file with normalized integral regions (M).....	368
nll	Find line frequencies and intensities (C).....	368
nlni	Find normalized integral values (obsolete)	369
369		
nm	Select normalized intensity mode (C).....	369
nm2d	Select Automatic 2D normalization (M)	369
noedif	Convert parameters for NOE difference experiment (M)	370
NOESY	Change parameters for NOESY experiment (M).....	370
noesy	Set up parameters for NOESY pulse sequence (M).....	370
NOESY1D	Change parameters for NOESY1D experiment (M).....	371
noise	Measure noise level of FID (C).....	371
noisemult	Control noise multiplier for automatic 2D processing (M).....	371
noislm	Limit noise in spectrum (M)	372
np	Number of data points (P)	372
npoint	Number of points for fp peak search (P)	373
nrecords	Determine number of lines in a file (M).....	373
ns	Number of slices to be acquired (P).....	373
nscans	Number of scout scan or real scan repetitions (P).....	373
nt	Number of transients (P)	373
ntrig	Number of trigger signals to wait before acquisition (P).....	374
ntype3d	Specify whether f_1 or f_2 display expected to be N-type (P).....	374
numrcvrs	Number of receivers in the system (P)	374
numreg	Return the number of regions in a spectrum (C).....	374
numrfch	Number of rf channels (P).....	375
nv	Number of phase encode steps (P)	375
O		
off	Make a parameter inactive (C)	376
offset	Calculate frequency offset of cursor (M)	376
on	Make a parameter active or test its state (C)	376
opx	Open shape definition file for Pbox (M)	377
orient	Slice plane orientation (P).....	377
oscoef	Digital filter coefficients for oversampling (P).....	377
osfb	Digital filter bandwidth for oversampling (P).....	378
osfilt	Oversampling filter for real-time DSP (P)	378
oslsfrq	Bandpass filter offset for oversampling (P).....	378
overrange	Frequency synthesizer overrange (P)	379

oversamp	Oversampling factor for acquisition (P).....	379
P		
p1	Enter pulse width for p1 in degrees (C)	381
p1	First pulse width (P)	381
p1pat	Shape of excitation pulse (P).....	381
p2	180° refocus pulse width (P).....	381
p2pat	RF pulse pattern of 180° refocus pulse p2 (P)	382
p2pul	Set up sequence for PFG testing (M)	382
p31	Automated phosphorus acquisition (M).....	382
p31p	Process 1D phosphorus spectra (M).....	382
pa	Set phase angle mode in directly detected dimension (C)	383
pa1	Set phase angle mode in 1st indirectly detected dimension (C).....	383
pacosy	Plot automatic COSY analysis (C).....	384
pad	Preacquisition delay (P)	384
padept	Perform adept analysis and plot resulting spectra (C)	385
page	Submit plot and change plotter page (C)	385
pap	Plot out “all” parameters (C).....	386
par2d	Create 2D acquisition, processing, and display parameters (M).....	386
par3d	Create 3D acquisition, processing, and display parameters (M).....	387
par3rf	Get display templates for 3rd rf channel parameters (M)	387
par4d	Create 4D acquisition parameters (M)	387
paramedit	Edit a parameter and its attributes with user-selected editor (C)	387
paramvi	Edit a parameter and its attributes with vi editor (M)	388
pards	Create additional parameters used by downsampling (M).....	388
parfidss	Create parameters for time-domain solvent subtraction (M)	389
parfix	Update parameter sets (M).....	390
parlc	Create parameters for LC-NMR experiments (M).....	390
parl12d	Create parameters for 2D peak picking (M)	390
parlp	Create parameters for linear prediction (M).....	391
parmax	Parameter maximum values (P)	391
parmin	Parameter minimum values (P)	392
paros	Create additional parameters used by oversampling (M).....	392
parstep	Parameter step size values (P).....	392
parstyle	Parameter style for plotting (P)	392
parversion	Version of parameter set (P).....	393
path3d	Path to currently displayed 2D planes from a 3D data set (P)	393
patlist	Active pulse template parameter list (P)	394
paxis	Plot horizontal LC axis (M)	394
Pbox	Pulse shaping software (U)	394
pbox_bw	Define excitation band (M)	395
pbox_bws	Define excitation band for solvent suppression (notch) pulses (M).....	396
pbox_dmf	Extract dmf value from pbox.cal or Pbox shape file (M).....	396
pbox_dres	Extract dres value from pbox.cal or Pbox shape file (M).....	396
pbox_name	Extract name of last shape generated by Pbox from pbox.cal (M)	396
pbox_pw	Extract pulse length from pbox.cal or Pbox shape file (M)	397
pbox_pwr	Extract power level from Pbox.cal or Pbox shape file (M)	397
pbox_pwrf	Extract fine power level from pbox.cal or Pbox shape file (M)	397
pboxget	Extract Pbox calibration data (M)	398
pboxpar	Add parameter definition to the Pbox.inp file (M).....	398
pboxrst	Reset temporary Pbox VNMR variables (M).....	398
pboxunits	Converts to Pbox default units (M)	399
pcmapapply	Apply phase correction map to data in EPI experiments (C).....	399
pcmapclose	Close phase correction map in EPI experiments (C)	399
pcmapgen	Generate phase correction map in EPI experiments (C)	399
pcmapopen	Open phase correction map in EPI experiments (C).....	400
pcon	Plot contours on a plotter (C).....	400
pcss	Calculate and show proton chemical shifts spectrum (M)	401

peak	Find tallest peak in specified region (C)	401
peak2d	Return information about maximum in 2D data (C).....	402
peccfile	Programmable eddy current compensation file (obsolete).....	402
pen	Select a pen or color for drawing (C).....	402
pexpl	Plot exponential or polynomial curves (C)	403
pexpladd	Add another diffusion analysis to current plot (M).....	403
pfgon	Pulsed field gradient amplifiers on/off control (P).....	404
pfiltr	Programmable filters (P)	404
pfww	Plot FIDs in whitewash mode (C).....	404
pge	Convert parameter set to PGE pulse sequence (M).....	405
pge_calib	Calibrate gradient strengths for PGE pulse sequence (M).....	405
pge_data	Extract data from single element of PGE pulse sequence (M)	405
pge_output	Output results from PGE pulse sequence (M).....	406
pge_process	Automated processing of data from PGE pulse sequence (M)	406
pge_results	Calculate diffusion constant for integral region (M).....	406
pge_setup	Set up gradient control parameters for PGE pulse sequence (M)	406
ph	Set phased mode in directly detected dimension (C).....	407
ph1	Set phased mode in 1st indirectly detected dimension (C)	407
ph2	Set phased mode in 2nd indirectly detected dimension (C).....	408
phase	Change frequency-independent phase rp (M)	408
phase	Phase selection (P)	409
phase1	Phase of first pulse (P)	409
phase2	Phase selection for 3D acquisition (P)	409
phase3	Phase selection for 4D acquisition (P)	409
phasing	Control update region during interactive phasing (P)	410
phfid	Zero-order phasing constant for the np FID (P).....	410
phfid1	Zero-order phasing constant for ni interferogram (P).....	410
phfid2	Zero-order phasing constant for ni2 interferogram (P).....	411
phi	Euler angle phi from magnet frame (P).....	411
PHOSPHORUS	Set up parameters for phosphorus spectrum (M)	411
pi	Inversion pulse length (P)	412
pi3ssbsq	Set up pi/3 shifted sinebell-squared window function (M)	412
pi4ssbsq	Set up pi/4 shifted sinebell-squared window function (M)	412
pilot	Automatic sequence setup (P).....	412
pintvast	Plots of integral regions (M)	413
pipat	Shape of an inversion pulse (P).....	413
pir	Plot integral amplitudes below spectrum (C).....	413
pirn	Plot normalized integral amplitudes below spectrum (M).....	413
pkpick	Peak pick (P)	414
pl	Plot spectra (C).....	414
pl2d	Plot 2D spectra in whitewash mode (C).....	415
plan	Display menu for planning a target scan (M).....	415
plane	Currently displayed 3D plane type (P).....	417
planlock	Planner lock (P).....	417
plapt	Plot APT-type spectra automatically (M).....	417
plarray	Plotting macro for arrayed 1D spectra (M).....	418
plate_glue	Define a glue order for plotting and display (U).....	418
plc	Plot a carbon spectrum (M).....	418
plcosy	Plot COSY- and NOESY-type spectra automatically (M).....	419
pldept	Plot DEPT data, edited or unedited (M)	419
plfid	Plot FIDs (C).....	420
plfit	Plot deconvolution analysis (M).....	420
plgrid	Plot a grid on a 2D plot (M)	421
plh	Plot proton spectrum (M).....	421
plhet2dj	Plot heteronuclear J-resolved 2D spectra automatically (M)	421
plhom2dj	Plot homonuclear J-resolved 2D spectra automatically (M).....	422
plhxcor	Plot X,H-correlation 2D spectrum (M)	423
plist	Active pulse length parameter list (P).....	424
pll	Plot a line list (M)	424

pl12d	Plot results of 2D peak picking (C).....	424
plot	Automatically plot spectra (M).....	425
plot1d	Plotting macro for simple (non-arrayed) 1D spectra (M).....	425
plot2D	Plot 2D spectra (M).....	426
plotside	Plot spectrum on side (M).....	426
plotter	Plotter device (P).....	426
plottop	Plot spectrum on top (M).....	427
plottopside	Plot spectrum on top and side (M).....	427
plp	Plot phosphorus spectrum (M).....	427
plplanes	Plot a series of 3D planes (M).....	427
plttext	Plot text file (M).....	428
pltmod	Plotter display mode (P).....	428
plvast	Plot VAST data in a stacked 1D-NMR matrix format (M).....	429
plvast2d	Plot VAST data in a stacked pseudo-2D format (M).....	429
plww	Plot spectra in whitewash mode (C).....	430
pmode	Processing mode for 2D data (P).....	430
poly0	Display mean of the data in regression.inp file (M).....	431
pos1,pos2,pos3	Position of voxel center (P).....	431
pp	Decoupler pulse length (P).....	432
ppa	Plot a parameter list in plain English (M).....	432
ppcal	Proton decoupler pulse calibration (M).....	432
ppe	Position of image center on 2D phase encode axis (P).....	433
ppf	Plot peak frequencies over spectrum (C).....	433
pph	Print pulse header (M).....	433
pplvl	Proton pulse power level (P).....	434
ppmm	Resolution on printers and plotters (P).....	434
pprofile	Plot pulse excitation profile (M).....	434
pps	Plot pulse sequence (C).....	435
presat	Set up parameters for PRESAT pulse sequence (M).....	435
presig	Preamplifier signal level selection (P).....	435
prevpl	Display the previous 3D plane (M).....	436
printer	Printer device (P).....	436
printoff	Stop sending text to printer and start print operation (C).....	436
printon	Direct text output to printer (C).....	437
pro	Position of image center on the readout axis (P).....	437
probe	Probe type (P).....	437
Probe_edit	Edit probe for specific nucleus (U).....	437
probe_edit	Edit probe for specific nucleus (M).....	438
probe_protection	Probe protection control (P).....	438
proc	Type of processing on np FID (P).....	438
proc1	Type of processing on ni interferogram (P).....	439
procl1d	Processing macro for simple (non-arrayed) 1D spectra (M).....	439
proc2	Type of processing on ni2 interferogram (P).....	439
proc2d	Process 2D spectra (M).....	440
proccarray	Process arrayed 1D spectra (M).....	440
process	Generic automatic processing (M).....	441
procplot	Automatically process FIDs (M).....	441
profile	Set up pulse sequence for gradient calibration (M).....	442
proj	Project 2D data (C).....	442
PROTON	Set up parameters for proton spectrum (M).....	443
prune	Prune extra parameters from current tree (C).....	443
pscale	Plot scale below spectrum or FID (C).....	443
pseudo	Set default parameters for pseudo-echo weighting (M).....	444
psg	Display pulse sequence generation errors (M).....	444
psggen	Compile a user PSG object library (M,U).....	444
psgset	Set up parameters for various pulse sequences (M).....	444
psgupdateon	Enable update of acquisition parameters (C).....	445
psgupdateoff	Prevent update of acquisition parameters (C).....	445
pshape	Plot pulse shape or modulation pattern (M).....	445

pshapef	Plot the last created pulse shape (M).....	445
psi	Euler angle psi from magnet frame (P).....	446
pslabel	Pulse sequence label (P).....	446
pss	Slice position (P).....	446
ptext	Print out a text file (M).....	446
ptspec3d	Region-selective 3D processing (P).....	447
ptsval	PTS frequency synthesizer value (P).....	447
pulsecal	Update and display pulse calibration data file (M).....	448
pulseinfo	Shaped pulse information for calibration (M).....	448
pulsetool	RF pulse shape analysis (U).....	449
purge	Remove macro from memory (C).....	449
puttxt	Put text file into VNMR data file (C).....	449
putwave	Write a wave into Pbox.inp file (M).....	449
pw	Enter pulse width pw in degrees (C).....	450
pw	Pulse width (P).....	450
pw90	90° pulse width (P).....	450
pwd	Display current working directory (C).....	451
pwpat	Shape of refocusing pulse (P).....	451
pwr	Set power mode in directly detected dimension (C).....	451
pwr1	Set power mode in 1st indirectly detected dimension (C).....	452
pwr2	Set power mode in 2nd indirectly detected dimension (C).....	452
pwrlist	Active pulse power level parameter list (P).....	453
pwsadj	Adjust pulse interval time (M).....	453
pxcal	Decoupler pulse calibration (M).....	454
pxset	Assign Pbox calibration data to experimental parameters (M).....	454
pxshape	Generates a single-band shape file (M).....	454
Pxsim	Simulate Bloch profile for a shaped pulse (U).....	455
Pxspy	Create shape definition using Fourier coefficients (U).....	455
 Q		
QKexp	Set up quick experiment (M).....	456
qtune	Tune probe using swept-tune graphical tool (C).....	456
? (question mark)	Display individual parameter value (C).....	456
 R		
r	Recall display parameter set (M).....	458
r1-r7	Real-value storage for macros (P).....	458
ra	Resume acquisition stopped with sa command (C).....	458
rcvr	Receiver version in system (P).....	459
rcvrs	Which receivers to use (P).....	459
rcvrwt	Weighting for different receivers (P).....	459
rcvry	Pre-trigger delay (P).....	460
react	Recover from error conditions during werr processing (M).....	460
readallshims	Read all shims from hardware (M).....	460
readbrutape	Read Bruker data files from 9-track tape (U).....	461
readhw	Read current values of acquisition hardware (C).....	461
readlk	Read current lock level (C).....	461
readultra	Read shim coil setting for Ultra•nmr shim system (M).....	462
real	Create a real variable without a value (C).....	462
record	Record keyboard entries as a macro (M).....	462
redor1	Set up parameters for REDOR1 pulse sequence (M).....	463
redosy	Restore 2D DOSY display from subexperiment (M).....	463
reffrq	Reference frequency of reference line (P).....	463
reffrq1	Reference frequency of reference line in 1st indirect dimension (P).....	464
reffrq2	Reference frequency of reference line in 2nd indirect dimension (P).....	464
refpos	Position of reference frequency (P).....	464
refpos1	Position of reference frequency in 1st indirect dimension (P).....	464
refpos2	Position of reference frequency in 2nd indirect dimension (P).....	465

refsource1	Center frequency in 1st indirect dimension (P).....	465
refsource2	Center frequency in 2nd indirect dimension (P)	465
region	Divide spectrum into regions (C).....	465
relayh	Set up parameters for RELAYH pulse sequence (M)	466
rename	Move and/or rename a file (C)	466
rescal	Calculate pixel size and spatial resolution (M)	467
resetf3	Reset parameters after a partial 3D Fourier transform (M).....	467
resolv	Set resolution enhancement parameters (M).....	468
resto	NMR resonance offset frequency (P).....	468
resume	Resume paused acquisition queue (C)	468
return	Terminate execution of a macro (C).....	468
rev	System software revision level (P).....	469
revdate	System software preparation date (P)	469
rfband	RF band in use (P).....	469
rfblk	Reverse FID block (C)	469
rfchannel	Independent control of rf channel selection (P)	470
rfctype	Type of rf channel (P)	471
rfcoil	RF pulse calibration identity (P)	472
rfdata	Reverse FID data (C).....	472
rf1	Reference peak position in directly detected dimension (P).....	473
rf11	Reference peak position in 1st indirectly detected dimension (P)	473
rf12	Reference peak position in 2nd indirectly detected dimension (P).....	473
rfp	Reference peak frequency in directly detected dimension (P).....	474
rfp1	Reference peak frequency in 1st indirectly detected dimension (P).....	474
rfp2	Reference peak frequency in 2nd indirectly detected dimension (P).....	474
rftrace	Reverse FID trace (C)	474
rftype	Type of rf generation (P)	475
rfwg	RF waveform generator (P).....	476
right	Set display limits to right half of screen (C)	476
input	Input data for a regression analysis (M).....	477
rl	Set reference line in directly detected dimension (M)	477
rl1	Set reference line in 1st indirectly detected dimension (M)	477
rl2	Set reference line in 2nd indirectly detected dimension (M)	478
rm	Delete file (C).....	478
rmdir	Remove directory (C).....	479
rmsAddData	Add transformed data files with weighting (U).....	479
ROESY	Change parameters for ROESY experiment (M)	479
roesy	Set up parameters for ROESY pulse sequence (M)	479
rof1	Receiver gating time preceding pulse (P)	479
rof2	Receiver gating time following pulse (P).....	480
rotate	Rotate 2D data (C)	480
rotorsync	Rotor synchronization (P)	480
rp	Zero-order phase in directly detected dimension (P)	480
rp1	Zero-order phase in 1st indirectly detected dimension (P)	481
rp2	Zero-order phase in 2nd indirectly detected dimension (P).....	481
rsliceplan	Generate absolute magnet frame data (M).....	481
rt	Retrieve FIDs (M)	481
rtcmx	Return Spinsight data into current experiment (C)	482
rtp	Retrieve parameters (M).....	482
rtphf	Return stored phasefile to current VNMR phasefile (C)	483
rts	Retrieve shim coil settings (C)	483
rtshims	Extract shim parameter values (obsolete)	484
rttmp	Retrieve experiment data from experiment subfile (M)	484
rtv	Retrieve individual parameters (C)	484
S		
s	Save display parameters as a set (M)	486
s2pul	Set up parameters for standard two-pulse sequence (M)	486

s2pulr	Set up parameters for standard 2-pulse sequence in “reverse” (M)	486
sa	Stop acquisition (C)	487
sample	Submit change sample, Autoshim experiment to acquisition (M)	487
savefile	Base file name for saving files (P).....	488
saveglobal	Save selected parameters from global tree (P)	488
sb	Sinebell constant in directly detected dimension (P)	488
sb1	Sinebell constant in 1st indirectly detected dimension (P)	488
sb2	Sinebell constant in 2nd indirectly detected dimension (P)	489
sbs	Sinebell shift in directly detected dimension (P)	489
sbs1	Sinebell shift in 1st indirectly detected dimension (P).....	489
sbs2	Sinebell shift in 2nd indirectly detected dimension (P)	489
sc	Start of chart (P)	490
sc2	Start of chart in second direction (P).....	490
scalelimits	Set limits for scales in regression (M).....	490
scalesw	Set scaling factor for multipulse experiments (M).....	490
scalesw	Scale spectral width in directly detected dimension (P)	491
scalesw1	Set f_1 scaling factor for 2D multipulse experiments (M).....	491
scalesw1	Scale spectral width in 1st indirectly detected dimension (P).....	491
scalesw2	Scale spectral width in 2nd indirectly detected dimension (P)	491
sd	Set first decoupler frequency to cursor position (M)	492
sd2	Set second decoupler frequency to cursor position (M).....	492
sd3	Set third decoupler frequency to cursor position (M)	492
sda	Set first decoupler frequency array (M)	492
sd2a	Set second decoupler frequency array (M)	493
sd3a	Set third decoupler frequency array (M)	493
sdp	Show diffusion projection (M)	493
sediff	Set up spin-echo diffusion imaging sequence (M).....	493
select	Select spectrum, FID, trace, or 2D plane without display (C).....	494
selex	Defines excitation band (M)	494
selexcit	Set up PFG selective excitation pulse sequence (M)	495
sems	Set up basic imaging sequence with oblique capability (M).....	495
send2vnmr	Send a command to VNMR (U).....	495
seqcon	Acquisition loop control (P).....	496
seqfil	Pulse sequence name (P)	496
seqgen	Initiate compilation of user’s pulse sequence (M,U)	497
set2D	General setup for 2D experiments (M)	497
set2d	General setup for 2D experiments (M)	498
set3dproc	Set 3D processing (C)	498
setallshims	Set all shims into hardware (M)	499
setarray	Set up a parameter array (M)	499
setcenter	Set up parameters for center sequence calibration (M).....	499
setcolor	Set colors for graphics window and for plotters (C)	499
setdecpar	Set decoupler parameter values from probe file (M).....	501
setdec2pars	Set decoupler 2 parameter values from probe file (M).....	501
setdgroup	Set the Dgroup of a parameter in a tree (C)	501
setenumerat	Set values of a string parameter in a tree (C)	501
setether	Connect or reconnect host computer to Ethernet (U).....	502
setflip	Set rf power levels to desired flip angle (M).....	502
setfrq	Set frequency of rf channels (C)	502
setgauss	Set a Gaussian fraction for lineshape (M).....	503
setgcal	Set the gradient calibration constant (M).....	503
setgcoil	Assign sysgcoil configuration parameter (M).....	504
setglideexp	Set up <i>GLIDE</i> experiment from command line (M)	504
setGgrp	Add user to specific <i>GLIDE</i> group (U)	504
setgpe	Set phase encode gradient levels (M).....	505
setgrid	Divide graphics window into rows and columns (C)	505
setgro	Set readout gradient (M)	505
setgroup	Set group of a parameter in a tree (C)	506
setgss	Select slice or voxel selection gradient levels (M).....	506

sethw	Set values for hardware in acquisition system (C).....	507
setint	Set value of an integral (M).....	509
setlimit	Set limits of a parameter in a tree (C).....	509
setlk	Set up lock parameters (M).....	510
setlockfreq	Set lock frequency on systems other than UNITY and VXR-S (M).....	510
setloop	Control arrayed and real-time looping (M).....	511
setLP1	Set F1 linear prediction parameters.....	512
setnoether	Disconnect host computer from Ethernet (U).....	512
setoffset	Calculate offset frequency for given nucleus and ppm (M).....	512
setparams	Write parameter to current probe file (M).....	513
setpen	Set maximum number of HP plotter pens (M).....	513
setplotdev	Return characteristics of a named plotter (C).....	513
setpower	Set power and pulsewidth for a given γ B1 value (M).....	513
setprotect	Set protection mode of a parameter (C).....	514
setref	Set frequency referencing (M).....	515
setref1	Set frequency referencing for 1st indirectly detected dimension (M).....	516
setref2	Set frequency referencing for 2nd indirect detected dimension (M).....	516
setscout	Set up a scout run (M).....	516
setssfilter	Set sslsfrq to the frequencies of each of the suppressed solvents (M).....	517
setsw	Set spectral width (M).....	517
setsw1	Set spectral width in evolution dimension (M).....	517
setsw2	Set spectral width in 2nd evolution dimension (M).....	517
setselfrqc	Set selective frequency and width (M).....	518
setselinv	Set up selective inversion (M).....	518
settcldefault	Select default display templates for pulse sequence (M).....	518
settype	Change type of a parameter (C).....	518
setup	Set up parameters for basic experiments (M).....	519
setup_dosy	Set up gradient levels for DOSY experiments (M).....	519
setvalue	Set value of any parameter in a tree (C).....	520
setwave	Write a wave definition string into Pbox.inp file (M).....	520
setwin	Activate selected window (C).....	521
sf	Start of FID (P).....	521
sf1	Start of interferogram in 1st indirectly detected dimension (P).....	521
sf2	Start of interferogram in 2nd indirectly detected dimension (P).....	522
sfrq	Transmitter frequency of observe nucleus (P).....	522
sh2pul	Set up for a shaped observe excitation sequence (M).....	522
shdec	Set up for shaped observe excitation sequence (M).....	522
shell	Start a UNIX shell (C).....	523
shelli	Start an interactive UNIX shell (C).....	523
shellreturn	Run UNIX shell program and return arguments (obsolete).....	523
shim	Submit an Autoshim experiment to acquisition (C).....	523
shimset	Type of shim set (P).....	524
shimspath	Path to user's shims directory (P).....	525
showconsole	Show ^{UNITY} INOVA console configuration parameters (U).....	525
showfit	Display numerical results of deconvolution (M).....	525
showoriginal	Restore first 2D spectrum in 3D DOSY experiment (M).....	526
showplotq	Display plot jobs in plot queue (M).....	526
showprintq	Display print jobs in print queue (M).....	526
showstat	Display information about status of acquisition (M,U).....	526
sin	Find sine value of an angle (C).....	526
sine	Find values for a sine window function (M).....	527
sinebell	Select default parameters for sinebell weighting (M).....	527
sinesq	Find values for a sine-squared window function (M).....	528
slamp	Measured line amplitudes (obsolete).....	528
slfreq	Measured line frequencies (P).....	528
sliceorder	Reorder the slice position list (M).....	528
sliceplan	Set slice parameters for target slice (M).....	529
slp	Family of offset Frequencies of SLP shapes (P).....	529
slw	Spin simulation linewidth (P).....	529

small	Use small graphics window (C)	530
smaxf	Maximum frequency of any transition (P)	530
sminf	Minimum frequency of any transition (P).....	530
smsport	Sample Management System serial port connection (P).....	530
sn	Signal-to-noise ratio (P)	531
solppm	Return ppm and peak width of solvent resonances (M).....	531
solvent	Lock solvent (P)	531
solvfactor	Solvent correction factor (obsolete)	531
solvinfo	Retrieve information from solvent table (C)	532
sp	Start of plot in directly detected dimension (P)	532
sp1	Start of plot in 1st indirectly detected dimension (P).....	532
sp2	Start of plot in 2nd indirectly detected dimension (P)	532
spadd	Add current spectrum to add/subtract experiment (C).....	532
spcfrq	Display frequencies of rf channels (M).....	533
specdc3d	3D spectral dc correction (P).....	533
spin	Submit a spin setup experiment to acquisition (C)	534
spin	Sample spin rate (P)	534
spinll	Set up a slfreq array (M)	535
spinner	Open the Spinner Control window (C)	535
spinopt	Spin automation (P)	536
spins	Perform spin simulation calculation (C)	536
split	Split difference between two cursors (M).....	538
spmin	Take minimum of two spectra in add/subtract experiment (C).....	538
spsm	Enter spin system (M)	539
spsub	Subtract current spectrum from add/subtract experiment (C).....	539
sqcosine	Set up unshifted cosine-squared window function (M)	540
sq sinebell	Set up unshifted sinebell-squared window function (M)	540
srate	Spinning rate for magic angle spinning (P)	540
sread	Read converted data into VNMR (C).....	541
ss	Steady-state transients (P)	541
ss3d	f ₃ solvent subtraction option (obsolete)	541
ssecho	Set up solid-state echo pulse sequence (M)	541
ssecho1	Set up parameters for SSECHO1 pulse sequence (M).....	541
ssfilter	Full bandwidth of digital filter to yield a filtered FID (P).....	542
sslsfrq	Center of solvent-suppressed region of spectrum (P)	542
ssntaps	Number of coefficients in digital filter (P)	542
ssorder	Order of polynomial to fit digitally filtered FID (P)	543
ssplan	Set slice parameters for target slice (M)	543
sslist	Conjugate gradient list (P)	544
ssprep	Calculate slice gradient and slice selection parameters (M).....	544
stack	Fix stacking mode for processing and plotting arrayed spectra (M).....	544
stackmode	Stacking control for processing arrayed 1D spectra (P).....	545
status	Display status of sample changer (C,U).....	545
stdshm	Interactively create a method string for autoshimming (M)	545
steam	Set up volume localized spectroscopy sequence (M).....	546
sth	Minimum intensity threshold (P)	546
string	Create a string variable (C)	546
strtext	Starting point for LP data extension in np dimension (P).....	546
strtext1	Starting point for LP data extension in ni dimension (P).....	547
strtext2	Starting point for LP data extension in ni2 dimension (P).....	547
strtlp	Starting point for LP calculation in np dimension (P)	547
strtlp1	Starting point for LP calculation in ni dimension (P)	548
strtlp2	Starting point for LP calculation in ni2 dimension (P)	548
su	Submit a setup experiment to acquisition (M)	548
sub	Subtract current FID from add/subtract experiment (C)	549
substr	Select a substring from a string (C)	549
suselfrq	Select peak, continue selective excitation experiment (M).....	550
svdat	Save data (C)	550
svdef	Copy .def files with FID (M).....	551

svf	Save FIDs in current experiment (M)	551
svfdf	Save FID data in FDF format (M).....	551
svib	Generate and save images as ImageBrowser FDF files (M)	552
svimg	Generate and save images as FDF files (M).....	553
svp	Save parameters from current experiment (M)	554
svphf	Save current VNMR phasefile (C)	554
svs	Save shim coil settings (C).....	555
svs	Spin simulation vertical scale (P).....	555
svsis	Generate and save images as FDF files (M).....	555
svtmp	Move experiment data into experiment subfile (M).....	556
sw	Spectral width in directly detected dimension (P)	557
sw1	Spectral width in 1st indirectly detected dimension (P).....	558
sw2	Spectral width in 2nd indirectly detected dimension (P)	558
sw3	Spectral width in 3rd indirectly detected dimension (P).....	558
syn	Number of frequency synthesizers (obsolete).....	558
sysgcoil	System gradient coil (P).....	559
system	System type (P)	559
systemdir	VNMR system directory (P)	559
 T		
t1	T_1 exponential analysis (M)	560
t1image	Fit arrayed imaging data to T_1 exponential data (M)	560
t1s	T_1 exponential analysis with short output table (M)	560
t2	T_2 exponential analysis (M)	561
t2image	Fit arrayed imaging data to T_2 exponential data (M)	561
t2s	T_2 exponential analysis with short output table (M)	561
tabc	Convert data in table order to linear order (M)	562
tan	Find tangent value of an angle (C).....	563
tape	Read tapes from VXR-style system (M,U)	563
tape	Control tape options of files program (P).....	564
tbox	Draw a tilted box (C).....	564
tcapply	Apply table conversion reformatting to data (C).....	565
tcclose	Close table conversion file (C)	565
tcl	Send Tcl script to Tcl version of dg window (C).....	565
tcopen	Open table conversion file (C)	565
te	Echo time (P)	566
techron	Set up parameters for gradient amplifier tests (M).....	566
temp	Open the Temperature Control window (C).....	566
temp	Sample temperature (P).....	567
tempcal	Temperature calculation (C).....	567
tep	Post-acquisition delay in EPI experiments (P).....	567
testct	Check ct for resuming signal-to-noise testing (M)	568
testsn	Test signal-to-noise of a spectrum (M)	568
text	Display text or set new text for current experiment (C).....	568
textis	Return the current text display status (C).....	569
textvi	Edit text file of current experiment (M)	569
th	Threshold (P).....	570
th2d	Threshold for integrating peaks in 2D spectra (P)	570
thadj	Adjust threshold for peak printout (M)	570
theta	Euler angle theta from magnet frame (P).....	571
thk	Slice thickness (P).....	571
ti	Inversion recovery time (P).....	571
ticks	Number of trigger pulses (P).....	571
time	Display experiment time or recalculate number of transients (M)	571
tin	Temperature interlock (P)	572
title	Plot a title on a plotter (M).....	572
tlt	First-order baseline correction (P)	573
tmove	Left-shift FID to time-domain cursor (M)	573

tmsref	Reference 1D proton or carbon spectrum to TMS (M).....	573
tn	Nucleus for observe transmitter (P)	573
tncosyps	Set up parameters for TNCOSYPS pulse sequence (M).....	574
tndqcosy	Set up parameters for TNDQCOSY pulse sequence (M)	574
tnmqcosy	Set up parameters for TNMQCOSY pulse sequence (M).....	574
tnnoesy	Set up parameters for TNNOESY pulse sequence (M)	574
tnroesy	Set up parameters for TNROESY pulse sequence (M)	574
tntocsy	Set up parameters for TNTOCOSY pulse sequence (M).....	574
tnuc	Retrieve nucleus table parameters for transmitter (obsolete).....	575
TOCSY	Change parameters for TOCSY experiment (M)	575
tocsy	Set up parameters for TOCSY pulse sequence (M)	575
TOCSY1D	Change parameters for TOCSY1D experiment (M)	575
tof	Frequency offset for observe transmitter (P).....	575
tpe	Duration of the phase encoding gradient pulse (P)	576
tpe2,tpe3	Duration of second and third phase encoding gradient periods (P)	576
tpwr	Observe transmitter power level with linear amplifiers (P).....	576
tpwr1	Intensity of an excitation pulse (P).....	577
tpwr2	Intensity of an excitation pulse (P).....	577
tpwrcal	Calibrate power levels of 90° and 180° pulse (M)	577
tpwrf	Observe transmitter fine power (P)	578
tpwri	Intensity of inversion pulse (P)	578
tpwrm	Observe transmitter linear modulator power (P).....	578
tr	Repetition time in imaging and localized spectroscopy experiments (P)	578
trace	Mode for <i>n</i> -dimensional data display (P).....	579
transfer	Move parameters to target experiment (M).....	579
traymax	Sample changer tray slots (P).....	580
trfunc	Translate screen coordinates (M)	580
trfuncd	Translate screen distance (M)	581
trise	Gradient rise time (P)	581
troesy	Set up parameters for TROESY pulse sequence (M).....	581
tshift	Adjust tau2 to current cursor position (M).....	582
tspoil	Gradient spoiling time (P).....	582
tugain	Amount of receiver gain used by qtune (P).....	582
tune	Assign a frequency to a channel for probe tuning (C)	582
tuneoff	Turn off probe tuning mode on MERCURY-Series, GEMINI 2000 (M)	583
U		
undospins	Restore spin system as before last iterative run (M)	585
undosy	Restore original 1D NMR data from subexperiment (M).....	585
unit	Define conversion units (C).....	585
unix_vxr	Convert UNIX text files to VXR-style format (M,U)	586
unlock	Remove inactive lock and join experiment (C)	587
updatepars	Update all parameter sets saved in a directory (M).....	587
updateprobe	Update probe file (M).....	587
updaterev	Update after installing new VNMR version (M).....	588
updtgcoil	Update gradient coil (M)	588
updtparam	Update specified acquisition parameters (C).....	588
usemark	Use “mark” output as deconvolution starting point (M)	588
userdir	VNMR user directory (P).....	589
usergo	Experiment setup macro called by go, ga, and au (M).....	589
userfixpar	Macro called by fixpar (M)	589
V		
vast1d	Set up initial parameters for VAST experiments (M).....	590
vastget	Selects and displays VAST spectra (M)	590
vastglue	Assemble related 1D datasets into a 2D (or pseudo-2D) dataset (M).....	590
vastglue2	Assemble related 1D datasets into a 2D (or pseudo-2D) dataset (M).....	591
vastgo	Turn off LC stopped flow automation and start VAST automation (M)	591

vbg	Run VNMR processing in background (U).....	591
vf	Vertical scale of FID (P).....	592
vi	Edit text file with vi text editor (M)	592
vn	Start VNMR directly (U).....	594
vnmr	Start VNMR in current windowing system (U)	595
vnmr2sc	VNMR to SpinCAD pulse sequence translator (M)	595
vnmr_accounting	Open VNMR Accounting window (U)	596
vnmrexit	Exit from the VNMR system (C)	596
vnmrplot	Plot files (U)	596
vnmrprint	Print text files (U).....	597
vo	Vertical offset (P)	597
vox1,vox2,vox3	Voxel dimensions (P)	597
voxplan	Set voxel parameters for voxel defined by 2D box cursor (M)	598
vp	Vertical position of spectrum (P)	598
vpf	Current vertical position of FID (P)	598
vpfi	Current vertical position of imaginary FID (P).....	598
vs	Vertical scale (P)	599
vs2d	Vertical scale for 2D displays (P).....	599
vsadj	Automatic vertical scale adjustment (M)	599
vsadj2	Automatic vertical scale adjustment by powers of 2 (M)	600
vsadjc	Automatic vertical scale adjustment for ¹³ C spectra (M)	600
vsadjh	Automatic vertical scale adjustment for ¹ H spectra (M).....	601
vsproj	Vertical scale for projections and traces (P)	601
vtc	Variable temperature cutoff point (P).....	601
vttype	Variable temperature controller present (P)	602
vtwait	Variable temperature wait time (P)	602
vxr_unix	Convert VXR-style text files to UNIX format (M,U)	602
vxrprint	Script for interface between VNMR and UNIX printing (obsolete).....	603
 W		
w	Who is using system (C)	604
walkup	Walkup automation (M)	604
waltz	WALTZ decoupling present (P)	604
wbs	Specify action when bs transients accumulate (C).....	604
wbs	When block size (P)	605
wc	Width of chart (P).....	605
wc2	Width of chart in second direction (P)	605
wcmax	Maximum width of chart (P).....	605
wc2max	Maximum width of chart in second direction (P)	606
werr	Specify action when error occurs (C).....	606
werr	When error (P)	606
wet1d	Set up parameters for a WET1D pulse sequence (M).....	607
wetdqcosy	Set up parameters for a WETDQCOSY pulse sequence (M)	607
wetgcosy	Set up parameters for a WETGCOSY pulse sequence (M)	607
wetghmqcps	Set up parameters for a WETGHMQCPS pulse sequence (M)	607
wetghsqc	Set up parameters for a WETGHSQC pulse sequence (M)	607
wetgmqcosy	Set up parameters for a WETGMSQC pulse sequence (M)	607
wetnoesy	Set up parameters for a WETNOESY pulse sequence (M)	607
wetpwxcal	Set up parameters for a WETPWXCAL pulse sequence (M).....	608
wettntocsy	Set up parameters for a WETTNTOCSY pulse sequence (M)	608
wetshape	Shape for pwet pulses (P).....	608
wexp	Specify action when experiment completes (C).....	608
wexp	When experiment completes (P)	608
wf	Width of FID (P)	609
wf1	Width of interferogram in 1st indirectly detected dimension (P).....	609
wf2	Width of interferogram in 2nd indirectly detected dimension (P)	609
wfgtest	Waveform generator test (M)	610
wft	Weight and Fourier transform 1D data (C)	610

wft1d	Weight and Fourier transform f_2 for 2D data (C).....	610
wft1da	Weight and Fourier transform phase-sensitive data (M).....	610
wft1dac	Combine arrayed 2D FID matrices (M).....	611
wft2d	Weight and Fourier transform 2D data (C).....	611
wft2da	Weight and Fourier transform phase-sensitive data (M).....	612
wft2dac	Combine arrayed 2D FID matrices (M).....	612
wftt3	Process f_3 dimension during 3D acquisition (M).....	612
which	Display which VNMR command or macro is used (M).....	613
wnt	Specify action when nt transients accumulate (C).....	613
wnt	When number of transients (P).....	614
wp	Width of plot in directly detected dimension (P).....	614
wp1	Width of plot in 1st indirectly detected dimension (P).....	614
wp2	Width of plot in 2nd indirectly detected dimension (P).....	614
write	Write formatted text to a device (C).....	614
writefid	Write numeric text file using a FID element (C).....	616
wstram	Send hardware configuration to acquisition console (C).....	616
wshim	Conditions when shimming is performed (P).....	617
wtfile	User-defined weighting in directly detected dimension (P).....	617
wtfile1	User-defined weighting in 1st indirectly detected dimension (P).....	618
wtfile2	User-defined weighting in 2nd indirectly detected dimension (P).....	618
wtgen	Compile user-written weighting functions (M,U).....	618
wti	Interactive weighting (C).....	619
wtia	Interactive weighting for 2D absorptive data (M).....	619
wysiwyg	Set plot display or full display (P).....	620
X		
x0	X-zero position of HP pen plotter or Postscript device (P).....	621
x1	X1 shim gradient (P).....	621
x2y2	X2Y2 shim gradient (P).....	621
x3	X3 shim gradient (P).....	621
x4	X4 shim gradient (P).....	621
xdiag	Threshold for excluding diagonal peaks when peak picking (P).....	621
xgate	Load time counter (M).....	622
xpol	Cross-polarization (P).....	622
xpolar	Set up parameters for XPOLAR pulse sequence (M).....	622
xpolar1	Set up parameters for XPOLAR1 pulse sequence (M).....	623
xy	XY shim gradient (P).....	623
xz	XZ shim gradient (P).....	623
xz2	XZ2 shim gradient (P).....	623
Y		
y0	Y-zero position of HP pen plotter or Postscript device (P).....	624
y1	Y1 shim gradient (P).....	624
y3	Y3 shim gradient (P).....	624
y4	Y4 shim gradient (P).....	624
yz	YZ shim gradient (P).....	624
yz2	YZ2 shim gradient (P).....	624
Z		
z	Add integral reset point at cursor position (C).....	626
z0	Z0 field position (P).....	626
z1	Z1 shim gradient (P).....	626
z1c	Z1C shim gradient (P).....	626
z2	Z2 shim gradient (P).....	627
z2c	Z2C shim gradient (P).....	627
z2x2y2	Z2X2Y2 shim gradient (P).....	627
z2x3	Z2X3 shim gradient (P).....	627

z2xy	Z2XY shim gradient (P).....	627
z2y3	Z2Y3 shim gradient (P).....	627
z3	Z3 shim gradient (P)	627
z3c	Z3C shim gradient (P).....	628
z3x	Z3X shim gradient (P).....	628
z3x2y2	Z3X2Y2 shim gradient (P).....	628
z3x3	Z3X3 shim gradient (P).....	628
z3xy	Z3XY shim gradient (P).....	628
z3y	Z3Y shim gradient (P).....	628
z3y3	Z3Y3 shim gradient (P).....	628
z4	Z4 shim gradient (P)	628
z4c	Z4C shim gradient (P).....	629
z4x	Z4X shim gradient (P).....	629
z4x2y2	Z4X2Y2 shim gradient (P).....	629
z4xy	Z4XY shim gradient (P).....	629
z4y	Z4Y shim gradient (P).....	629
z5	Z5 shim gradient (P)	629
z5flag	Z5 shimming present (obsolete).....	629
z5x	Z5X shim gradient (P).....	629
z5y	Z5Y shim gradient (P).....	630
z6	Z6 shim gradient (P)	630
z7	Z7 shim gradient (P)	630
z8	Z8 shim gradient (P)	630
zap	Set up for gradient refocused high-speed imaging sequences (M)	630
zeroneg	Set all negative intensities of 2D spectra to zero (C)	630
zoom	Adjust display to given width (M)	630
zx2y2	ZX2Y2 shim gradient (P).....	631
zx3	ZX3 shim gradient (P).....	631
zxy	ZXY shim gradient (P).....	631
zy3	ZY3 shim gradient (P).....	631
Index	633

Introduction

The *VNMR Command and Parameter Reference* describes in detail the commands, macros, and parameters in VNMR 6.1C software. Information new to VNMR in this version is shown by a change bar (as shown to the left of this paragraph).

Title Line Codes

Each entry has a letter in parentheses in the title line that identifies the type of entry:

(C)	VNMR command
(M)	VNMR macro command (from the <code>mac.lib</code> directory)
(P)	VNMR parameter
(U)	UNIX command (not executable within VNMR)
(C,U) (M,U)	Executable from UNIX or VNMR (note that syntax is different)

Applicability

An entry with applicability information applies only to the system or accessory listed. If the entry does not include applicability information, the entry applies to all systems.

Command and Macro Syntax

Each command and macro entry includes the syntax used when entering it into the system. The following examples illustrate this syntax:

<code>halt</code>	If no parentheses are shown, enter the command or macro exactly as shown, e.g., enter <code>halt</code> .
<code>delexp(exp_num)</code>	If parentheses are shown, enter the command or macro name as shown, but replace arguments with a value, e.g., if <code>exp_num</code> is 5, enter <code>delexp(5)</code> .
<code>rttmp(file)</code>	Arguments can be a string (e.g., name of file or solvent), number, variable, or parameter (e.g., <code>pw</code>),. If a string, enclose it with single quote marks, e.g., if <code>file</code> is <code>samp02</code> , enter <code>rttmp('samp02')</code> . If number, variable, or parameter, do <i>not</i> use marks.
<code>r1<(frequency)></code>	Angle brackets (< and >) indicate optional input, e.g., if <code>frequency</code> not needed or the default value of <code>frequency</code> is acceptable, enter <code>r1</code> , but if <code>frequency</code> has a value such as 10, enter <code>r1(10)</code> .
<code>md(<from_exp,>to_exp)</code>	Arguments can also be optional. Use a comma to separate arguments, e.g., <code>md(2,3)</code> . Unless stated otherwise, the order of arguments is often important.
<code>n1l('<pos')></code>	A keyword is frequently used as an argument. In the syntax, keywords are shown in single quotes and are entered exactly as shown, e.g., to use the optional keyword <code>'pos'</code> for <code>n1l</code> , enter <code>n1l('pos')</code> .

<code>dc2d('f1' 'f2')</code>	A vertical bar indicates an OR condition, e.g., either 'f1' or 'f2' can be an argument to <code>dc2d</code> .
<code>sin(angle)<:n></code>	Some commands return values to a calling macro. This is shown by a colon followed by one or more variables, e.g., if <code>angle</code> is variable <code>x</code> and <code>n</code> is variable <code>rt</code> , then <code>sin(x):rt</code> returns the value of <code>sin(x)</code> to the calling macro via the variable <code>rt</code> .
<code>z(reset1,reset2,...)</code>	Three dots indicate the sequence of arguments continues. Unless a limit is given, you can enter one argument, two, three, or as many as needed.

Parameter Syntax

Parameter syntax is always in the form `parameter_name=value`. If `value` is a string, enclose it in single quote marks; otherwise, no marks are used, e.g., `auto='y'`, `plotter='ThinkJet'`, `spin=5`. Note that some parameters are not user-enterable.

Notational Conventions

Throughout all VNMR manuals, typewriter-like characters identify commands, parameters, directories, file names, and text displayed on the screen.

Because pressing the Return key is required at the end of almost every command or line of text you type on the keyboard, assume this use of the Return key unless stated otherwise.

GLIDE and Menu Buttons

Many commands can be executed by selecting buttons in the *GLIDE* user interface and the VNMR menu system. For example, moving the mouse cursor to the File button in the Main menu and clicking the left button on the mouse is the same as entering the `files` command. This is shown by the following entry in the description of the `files` command:

Alternate: File button in the Main menu

Refer to the online interactive help and the manual *Getting Started* for a complete description of *GLIDE* and the VNMR menu system.

Other Sources of Information

For further information about an entry, refer to the manual listed under "See also." For general coverage on VNMR, refer to the following manuals (each manual is also online):

Release Notes

Getting Started

Walkup NMR Using GLIDE

User Guide: Liquids NMR

User Guide: Solid-State NMR

User Guide: Imaging

VNMR User Programming

VNMR and Solaris Software Installation

A

aa **Abort acquisition with error (C)**

Syntax: aa

Description: Aborts an experiment that has been submitted to acquisition. If the experiment is active, it is aborted immediately, all data is discarded, and the experiment is interpreted as an error. Any data collected from an earlier block size transfer is retained. If any `werr` processing is defined, that processing occurs, followed by any queued experiments. The `login` name, and the FID directory path in `file` are used as keys to find the proper experiment to abort.

In some circumstances, there is a delay between the time `go` is entered and the acquisition is started. During this time, instructions based on the selected pulse sequence are being generated. This is signified by the letters “PSG” appearing in the upper left corner of the status window. An `aa` command issued under these circumstances reports that no acquisition is active but it instead stops the instruction generation process and the message “PSG aborted” appears.

See also: *Getting Started*

Related:	<code>file</code>	File name of a parameter set (P)
	<code>go</code>	Submit experiment to acquisition (C)
	<code>halt</code>	Abort acquisition with no error (C)
	<code>werr</code>	Specify action when error occurs (C)
	<code>werr</code>	When error (P)

abort **Terminate action of calling macro and all higher macros (C)**

Syntax: abort

Description: Terminates the action of the calling macro and all higher levels of nested macros. `abort` is used only in macros and not entered from the keyboard.

See also: *VNMR User Programming*

Related:	<code>abortoff</code>	Terminate normal functioning of <code>abort</code> in a macro (C)
	<code>aborton</code>	Restore normal functioning of <code>abort</code> in a macro (C)
	<code>return</code>	Terminate execution of a macro (C)

abortallacqs **Reset acquisition computer in a drastic situation (C)**

Applicability: All systems except *MERCURY* and *GEMINI 2000*

Syntax: abortallacqs

Description: Reboots the acquisition system from the host computer. Wait at least 30 seconds before attempting new acquisitions.

See also: *Getting Started*

abortoff **Terminate normal functioning of abort in a macro (C)**

Syntax: abortoff

Description: Changes the action of an `abort` command in a macro. Normally, `abort` terminates the action of the calling macro and all higher levels of nested macros; however if the `abortoff` command is executed prior to a macro containing

the `abort` command, only the macro containing `abort` terminates and execution continues to the next macro. The operation of the `abortoff` command is nullified by the `aborton` command. `abortoff` is used only in macros and not entered from the keyboard.

See also: *VNMR User Programming*

Related: `abort` Terminate action of calling macro and all higher macros (C)
`aborton` Restore normal functioning of `abort` in a macro (C)

aborton Restore normal functioning of abort in a macro (C)

Syntax: `aborton`

Description: Nullifies the operation of a `abortoff` command and restores the normal functioning of the `abort` command. `aborton` is used only in macros and not entered from the keyboard.

See also: *VNMR User Programming*

Related: `abortoff` Terminate normal functioning of `abort` in a macro (C)

abs Find absolute value of a number (C)

Syntax: `abs(number) <:value>`

Description: Finds the absolute value of a number. Absolute value is a nonnegative number equal in numerical value to the given number (e.g., `abs(-6.5)` is 6.5).

Arguments: `number` is the given real number.

`value` is the return value with the absolute value of the given number. The default is to display the value in the status window.

Examples: `abs(-25)`
`abs(n):abs_val`

See also: *VNMR User Programming*

AC1-AC9 Automated calibration (obsolete)

Description: These macros are no longer used in VNMR and are replaced by `AC1S-AC11S`.

Related: `AC1S-AC11S` Autocalibration macros (M)

AC1S-AC11S Autocalibration macros (M)

Applicability: `UNITYINOVA`, `MERCURY-VX`, `MERCURY`, and `GEMINI 2000` systems

Syntax: `ACnS`, where `n` is a number from 1 to 11.

Description: Performs automatic system calibration. When finished with the calibration routines, the current probe file is updated. If the probe is new to the system (i.e., all values in the probe file are zero), system power levels are determined followed by calibration. If power levels are listed in the current probe file, these values are used. The macro `AC1S` determines ^1H pw90, `AC5S` begins ^{13}C calibration, including decoupler power calibrations. `AC10S` performs ^{19}F calibration, and `AC11S` performs ^{31}P calibration.

See also: *Getting Started*

ACbackup Make backup copy of current probe file (M)

Applicability: `UNITYINOVA`, `MERCURY-VX`, `MERCURY`, and `GEMINI 2000` systems

Syntax: ACbackup

Description: Called by the autocalibration macros **AC1S-AC11S** to back up the probe file after calibration ends. This macro is not usually called by the user.

See also: *Getting Started*

Related: **AC1S-AC11S** Autocalibration macros (M)

ACreport Print copy of probe file after autocalibration (M)

Applicability: *UNITY INOVA, MERCURY-VX, MERCURY, and GEMINI 2000* systems

Syntax: ACreport

Description: Called by the autocalibration macros **AC1S-AC11S** to print a copy of the probe file before beginning a new autocalibration run.

See also: *Getting Started*

Related: **AC1S-AC11S** Autocalibration macros (M)

acos Find arc cosine of number (C)

Syntax: `acos (value) <:n>`

Description: Finds the arc cosine (also called the inverse cosine) of a number.

Arguments: `value` is a number in the range of ± 1.0 to $+1.0$.

`n` is a return argument giving the arc cosine, in radians, of `value`. The default is to display the arc cosine value in the status window.

Examples: `acos (.5)`
`acos (value) : acos_val`

See also: *VNMR User Programming*

Related: **sin** Find sine value of an angle (C)

acosy Automatic analysis of COSY data (C)

Syntax: `acosy`

Description: Automatically analyzes a 2D COSY data set with **fn=fn1** and **sw=sw1**. In this algorithm, a fuzzy pattern recognition technique is used to detect peaks and cluster the cross peaks into groups. Symmetry measures and chemical shifts for all cross peaks are calculated. Connectivities and the correlation table are displayed on the computer screen. This method is less sensitive to the threshold and rejects most artifacts in the peak list. The old algorithm used in the previous version of VNMR, **acosyold**, is still available for comparison.

Alternate: Find Correlations button in the Automatic COSY Analysis Menu.

See also: *User Guide: Liquids NMR*

Related: **acosyold** Automatic analysis of COSY data (C)
fn Fourier number in 1st indirectly detected dimension (P)
fn1 Fourier number in directly detected dimension (P)
l12d Automatic and interactive 2D peak picking (C)
sw Spectral width in directly detected dimension (P)
sw1 Spectral width in 1st indirectly detected dimension (P)

acosyold Automatic analysis of COSY data, old algorithm (C)

Syntax: `acosyold`

Description: Analyzes COSY data using the algorithm from previous versions of VNMR.

Related: `acosy` Automatic analysis of COSY data (C)

`acqdisp` Display message on the acquisition status line (C)

Syntax: `acqdisp (message)`

Description: Displays the message specified on the acquisition status line. `acqdisp` is used primarily by the acquisition process to update the VNMR screen.

Arguments: `message` is a text string, up to 8 characters long.

See also: *Getting Started*

`acqi` Interactive acquisition display process (C)

Syntax: `acqi (<'par' | 'disconnect' | 'exit' | 'standby'> <:$ret>)`

Description: Opens the Acquisition window for interactive locking and shimming on the lock signal, FID, or spectrum. When using a spectrometer, `acqi` normally automatically starts. On ^{UNITY}INOVA systems only, you can use the Acquisition window to shim on the sample while an acquisition is in progress. This feature is not available on other systems. On all systems, if the console has been recently rebooted, enter `su` before running `acqi`.

If `acqi` is connected to the console and you start an acquisition (`su/go/au`), `acqi` automatically disconnects.

The pulse sequence and parameter set for the FID/spectrum display can be selected by entering `gf` from VNMR. Note that if clicking the FID button in `acqi` causes `acqi` to “disconnect,” the common cause is that `gf` had not been executed from VNMR.

The FID display is controlled by the parameters `lsfid`, `phfid`, and `dmgf`. These display parameters are automatically sent to `acqi` when `acqi` is first invoked. These parameters may subsequently be changed and sent again to `acqi` with the command `acqi ('par')`. If `phfid` is not set to “Not Used” for the FID display in `acqi`, a slide control will be available in `acqi` for the interactive adjustment of the `phfid` parameter. The slide will be in the IPA set of adjustments. If the parameter `dmgf` exists and is set to 'av', the FID display in `acqi` displays the square root of the sum of the squares of the real and imaginary channels.

The spectrum display is controlled by parameters `sp`, `wp`, `dmg`, `rp`, `lp`, `rfl`, `rfp`, `vs`, `vp`, `sw`, and `fn`. These parameters are automatically sent to `acqi` when `acqi` is first invoked. These parameters can subsequently be changed and sent again to `acqi` with the command `acqi ('par')`. The preparation macro `gf` also calls `acqi ('par')`, thereby causing these parameters to be sent to `acqi`. If `fn` is greater than 64K, it is lowered to 64K.

A convenient method of setting these parameters is to acquire a spectrum with `go`, then `ft` and adjust the display with the `ds` command options. Once the display is set the way you want, enter `gf`. The same display should then appear when the spectrum display is selected from `acqi`. Note that weighting parameters are not used in the `acqi` spectrum display.

The manual *Getting Started* has a step-by-step description of using `acqi`.

Arguments: 'par' causes the current values of parameters `lsfid`, `phfid`, `dmgf`, `sp`, `wp`, `dmg`, `rp`, `lp`, `rfl`, `rfp`, `vs`, `sw`, and `fn` to be sent to `acqi`.

'disconnect' causes `acqi` to be disconnected. Clicking the Close button in `acqi` is equivalent, and puts `acqi` in the standby mode. Lock parameters,

the `spin` parameter, and the shim values are sent back to the current VNMR experiment when `acqi` is “disconnected.” If the experiment has the `load` parameter set to 'y', then the shim values are not delivered to the experiment. (Spin adjustment is optional on *MERCURY* and *GEMINI 2000* systems.)

'`exit`' causes an exit from `acqi`. Clicking the exit button in the Acquisition window is equivalent.

`$ret` is a return value with the success or failure of running `acqi`. The default is a warning displayed in the status window if `acqi` fails.

'`standby`' starts `acqi` and puts it into the standby mode. In this mode, a button labeled `Acqi` is present in VNMR's permanent menu.

Examples: `acqi`
`acqi('par')`
`acqi('disconnect')`
`acqi('exit')`
`acqi:$ok`

See also: *Getting Started*

Related:	<code>Acqstat</code>	Bring up the acquisition status display (U)
	<code>dmg</code>	Display mode in directly detected dimension (P)
	<code>dmgf</code>	Absolute-value display of FID data or spectrum in <code>acqi</code> (P)
	<code>ds</code>	Display a spectrum (C)
	<code>fn</code>	Fourier number in directly detected dimension (P)
	<code>ft</code>	Fourier transform 1D data (C)
	<code>gf</code>	Prepare parameters for FID/spectrum display in <code>acqi</code> (M)
	<code>go</code>	Submit an experiment to acquisition (C)
	<code>load</code>	Load status of displayed shims (P)
	<code>lkof</code>	Track changes in lock frequency (P)
	<code>lp</code>	First-order phase in directly detected dimension (P)
	<code>lsfid</code>	Number of complex points to left-shift the <code>np</code> FID (P)
	<code>phfid</code>	Zero-order phasing constant for <code>np</code> FID (P)
	<code>rfl</code>	Ref. peak position in 1st indirectly detected dimension (P)
	<code>rfp</code>	Ref. peak frequency in directly detected dimension (P)
	<code>rp</code>	Zero-order phase in directly detected dimension (P)
	<code>sp</code>	Start of plot in directly detected dimension (P)
	<code>spin</code>	Sample spin rate (P)
	<code>sw</code>	Spectral width in directly detected dimension (P)
	<code>vp</code>	Vertical position of the spectrum (P)
	<code>vs</code>	Vertical scale (P)
	<code>wp</code>	Width of plot in directly detected dimension (P)

`acqmeter` **Open Acqmeter window (M)**

Syntax: `acqmeter<(remote_system)>`

Description: Opens the Acqmeter window and shows a time line of lock level, temperature (VT), and/or spinner speed. When first opened, only lock level is displayed. By clicking anywhere in the lock level window with the right mouse button, a menu pops up with choices to close the lock level window, show a temperature (VT) window, show a spinner window, open a properties window, or close the Acqmeter window. Click on the choice desired in the menu with either the left or right mouse button. In the properties window, the host, font, color, and graphical mode can be changed. Continue to click in any Acqmeter window with the right mouse button to open the menu and then open or close windows, or close the Acqmeter window, as desired.

Arguments: `remote_system` is the host name of a remote machine on the same network. The default is the local machine. To activate the remote feature, the local and remote machines must be on the same Ethernet LAN (local area network) and the local machine must be able to get the Internet address of the remote machine (usually in the `/etc/hosts` file).

Examples: `acqmeter`
`acqmeter('inova500')`

See also: *Getting Started; User Guide: Liquids NMR*

Related: `acqi` Interactive acquisition display (C)
`Acqmeter` Open Acqmeter window (U)

Acqmeter **Open Acqmeter window (U)**

Syntax: `Acqmeter <remote_system> <-f file> <&>`

Description: Opens the Acqmeter window and shows a time line of lock level, temperature (VT), and/or spinner speed. When first opened, only lock level is displayed. By clicking anywhere in the lock level window with the right mouse button, a menu pops up with choices to close the lock level window, show a temperature (VT) window, show a spinner window, open a properties window, or close the Acqmeter window. Click on the choice desired in the menu with either the left or right mouse button. In the properties window, the host, font, color, and graphical mode can be changed. Continue to click in any Acqmeter window with the right mouse button to open the menu and then open or close windows, or close the Acqmeter window, as desired.

Arguments: `remote_system` is the host name of a remote machine on the same network. The default is the local machine. To activate the remote feature, the local and remote machines must be on the same Ethernet LAN (local area network) and the local machine must be able to get the Internet address of the remote machine (usually in the `/etc/hosts` file).

`-f file` is the name of a template file in the directory `$vnmruser/vnmrsys/templates/acqstat` used to set the attributes of the Acqmeter window when it opens. This allows customizing the Acqmeter window for different users and experiments. The default name of the file is `default`.

`&` (ampersand) character added to the command makes `Acqmeter` into a background process. For example, if "lab" is the remote machine host name, entering the command `Acqmeter lab &` displays the acquisition status of the "lab" remote machine as a background process. To activate the remote feature, the local and remote machines must be on the same Ethernet LAN (local area network) and the local machine must be able to get the Internet address of the remote machine (usually in the `/etc/hosts` file).

Examples: `Acqmeter &`
`Acqmeter inova400 &`
`Acqmeter gem300 -f inova500.lisa &`

See also: *Getting Started; User Guide: Liquids NMR*

Related: `acqi` Interactive acquisition display (C)
`acqmeter` Open Acqmeter window (M)

acqstat **Open Acquisition Status window (M)**

Syntax: `acqstat<(remote_system)>`

Description: Opens the Acquisition Status window, which displays acquisition information such as the current acquisition task, experiment number, spinner status, and

temperature status. When the host computer is attached to a spectrometer, this window should open automatically when VNMR is started. In the properties window, the host, font, color, and graphical mode can be changed. For a complete description of these windows, refer to the manual *Getting Started*.

Arguments: `remote_system` is the host name of a remote machine on the same network. The default is the local machine. To activate the remote feature, the local and remote machines must be on the same Ethernet LAN (local area network) and the local machine must be able to get the Internet address of the remote machine (usually in the `/etc/hosts` file).

Examples: `acqstat`
`acqstat('u500')`

Alternate: Acquisition Status choice in the Workspace menu.

See also: *Getting Started*

Related: `Acqstat` Open the Acquisition Status window (U)
`showstat` Display information about status of acquisition (C,U)

Acqstat **Open Acquisition Status window (U)**

Syntax: `Acqstat <remote_system> <-f file> <&>`

Description: Opens the Acquisition Status window, which displays acquisition information such as the current acquisition task, experiment number, spinner status, and temperature status. When the host computer is attached to a spectrometer, this window should open automatically when VNMR is started. In the properties window, the host, font, color, and graphical mode can be changed. For a complete description of these windows, refer to the manual *Getting Started*.

Arguments: `remote_system` is the host name of a remote machine on the same network. The default is the local machine. To activate the remote feature, the local and remote machines must be on the same Ethernet LAN (local area network) and the local machine must be able to get the Internet address of the remote machine (usually in the `/etc/hosts` file).

`-f file` is the name of a template file in the directory `$vnmruser/vnmrsys/templates/acqstat` used to set the attributes of the Acquisition Status window when it opens. This allows customizing the Acquisition Status window for different users and experiments. The default name of the file is `default`.

`&` (ampersand) character added to the command makes `Acqstat` into a background process. For example, if “lab” is the remote machine host name, entering the command `Acqstat lab &` displays the acquisition status of the “lab” remote machine as a background process. To activate the remote feature, the local and remote machines must be on the same Ethernet LAN (local area network) and the local machine must be able to get the Internet address of the remote machine (usually in the `/etc/hosts` file).

Examples: `Acqstat &`
`Acqstat inova400 &`
`Acqstat gem300 -f inova500.lisa &`

Alternate: Acquisition Status choice in the Workspace menu.

See also: *Getting Started*

Related: `Acqstat` Open the Acquisition Status window (U)
`showstat` Display information about status of acquisition (C,U)

acqstatus Acquisition status (P)

Applicability: All systems, except codes marked with an asterisk (*) are not used on *MERCURY* and *GEMINI 2000* systems.

Description: Whenever **wbs**, **wnt**, **wexp**, or **werr** processing occurs, the acquisition condition that initiated that processing is available from the parameter **acqstatus**. This acquisition condition is represented by two numbers, a “done” code and an “error” code. The done code is set in **acqstatus[1]** and the error code is set in **acqstatus[2]**. Macros can take different actions depending on the acquisition condition.

The done codes and error codes are listed below and in the file **acq_errors** in **/vnmr/manual**. For example, a **werr** macro could specify special processing if the maximum number of transients is accumulated. The appropriate test in the macro would be:

```
if (acqstatus[2] = 200) then
  "do special processing, e.g. dp='y' au"
endif
```

Done codes:

11. FID complete
12. Block size complete (error code indicates **bs** number completed)
13. Soft error
14. Warning
15. Hard error
16. Experiment aborted
17. Setup completed (error code indicates type of setup completed)
101. Experiment complete
102. Experiment started

Error codes:

Warnings

101. Low-noise signal
102. High-noise signal
103. ADC overflow occurred
104. Receiver overflow occurred*

Soft errors

200. Maximum transient completed for single-precision data
201. Lost lock during experiment (LOCKLOST)

300. *Spinner errors:*

301. Sample fails to spin after three attempts at repositioning
302. Spinner did not regulate in the allowed time period (RSPINFAIL)*
303. Spinner went out of regulation during the experiment (SPINOUT)*
395. Unknown spinner device specified (SPINUNKNOWN)*
396. Spinner device is not powered up (SPINNOPOWER)*
397. RS-232 cable not connected from console to spinner (SPINRS232)*
398. Spinner does not acknowledge commands (SPINTIMEOUT)*

400. *VT (variable temperature) errors:*

400. VT did not regulate in the given time **vttime** after being set
401. VT went out of regulation during the experiment (VTOUT)
402. VT in manual mode after automatic command (see Oxford manual)*
403. VT safety sensor has reached limit (see Oxford manual)*
404. VT cannot turn on cooling gas (see Oxford manual)*
405. VT main sensor on bottom limit (see Oxford manual)*
406. VT main sensor on top limit (see Oxford manual)*
407. VT sc/ss error (see Oxford manual)*

- 408. VT oc/ss error (see Oxford manual)*
- 495. Unknown VT device specified (VTUNKNOWN)*
- 496. VT device not powered up (VTNOPOWER)*
- 497. RS-232 cable not connected between console and VT (VTRS232)*
- 498. VT does not acknowledge commands (VTTIMEOUT)
- 500. *Sample changer errors:*
- 501. Sample changer has no sample to retrieve
- 502. Sample changer arm unable to move up during retrieve
- 503. Sample changer arm unable to move down during retrieve
- 504. Sample changer arm unable to move sideways during retrieve
- 505. Invalid sample number during retrieve
- 506. Invalid temperature during retrieve
- 507. Gripper abort during retrieve
- 508. Sample out of range during automatic retrieve
- 509. Illegal command character during retrieve*
- 510. Robot arm failed to find home position during retrieve*
- 511. Sample tray size is not consistent*
- 512. Sample changer power failure during retrieve*
- 513. Illegal sample changer command during retrieve*
- 514. Gripper failed to open during retrieve*
- 515. Air supply to sample changer failed during retrieve*
- 525. Tried to insert invalid sample number*
- 526. Invalid temperature during sample changer insert*
- 527. Gripper abort during insert*
- 528. Sample out of range during automatic insert
- 529. Illegal command character during insert*
- 530. Robot arm failed to find home position during insert*
- 531. Sample tray size is not consistent*
- 532. Sample changer power failure during insert*
- 533. Illegal sample changer command during insert*
- 534. Gripper failed to open during insert*
- 535. Air supply to sample changer failed during insert*
- 593. Failed to remove sample from magnet*
- 594. Sample failed to spin after automatic insert
- 595. Sample failed to insert properly
- 596. Sample changer not turned on
- 597. Sample changer not connected to RS-232 interface
- 598. Sample changer not responding*
- 600. *Shimming errors:*
- 601. Shimming user aborted*
- 602. Lost lock while shimming*
- 604. Lock saturation while shimming*
- 608. A shim coil DAC limit hit while shimming*
- 700. *Autolock errors:*
- 701. User aborted (ALKABORT)*
- 702. Autolock failure in finding resonance of sample (ALKRESFAIL)
- 703. Autolock failure in lock power adjustment (ALKPOWERFAIL)*
- 704. Autolock failure in lock phase adjustment (ALKPHASFAIL)*
- 705. Autolock failure, lock lost in final gain adjustment (ALKGAINFAIL)*
- 800. *Autogain errors.*
- 801. Autogain failure, gain driven to 0, reduce pw (AGAINFAIL)
- Hard errors
- 901. Incorrect PSG version for acquisition
- 902. Sum-to-memory error, number of points acquired not equal to np

- 903. FIFO underflow error (a delay too small?)*
- 904. Requested number of data points (**np**) too large for acquisition*
- 905. Acquisition bus trap (experiment may be lost)*
- 1000. *SCSI errors:*
- 1001. Recoverable SCSI read transfer from console*
- 1002. Recoverable SCSI write transfer from console**
- 1003. Unrecoverable SCSI read transfer error*
- 1004. Unrecoverable SCSI write transfer error*
- 1100. *Host disk errors:*
- 1101. Error opening disk file (most likely a UNIX permission problem)*
- 1102. Error on closing disk file*
- 1103. Error on reading from disk file*
- 1104. Error on writing to disk file*

See also: *Getting Started*

Related: **react** Recover from error conditions during `werr` processing (M)
werr Specify action when error occurs (C)
werr When error (P)

add **Add current FID to add/subtract experiment (C)**

Syntax: (1) `add<(multiplier<,'new'>)>`
 (2) `add('new')`
 (3) `add('trace',index)`

Description: Adds the last displayed or selected FID to the current contents of the add/subtract experiment (`exp5`). The parameters `lsfid` and `phfid` can be used to shift or phase rotate the selected FID before it is combined with the data in the add/subtract experiment. A multi-FID add/subtract experiment can be created by using the 'new' keyword. Individual FIDs in a multi-FID add/subtract experiment can subsequently be added to using the 'trace' keyword followed by the index number of the FID.

Arguments: `multiplier` is a value that the FID is to be multiplied by before being added to the add/subtract experiment (`exp5`). The default is 1.0.
 'new' is a keyword to create a new FID element in a add/subtract experiment.
 'trace' is a keyword to use the next argument (`index`) as the number of the FID to add to in an add/subtract experiment. The default is to add to the first FID in a multi-FID add/subtract experiment.
`index` is the index number of the FID to be used as a target in a multi-FID add/subtract experiment.

Examples: `add`
`add(0.75)`
`add('new')`
`add('trace',2)`

See also: *User Guide: Liquids NMR*

Related: **clradd** Clear add/subtract experiment (C)
lsfid Number of complex points to left-shift `ni` interferogram (P)
phfid Zero-order phasing constant for `np` FID (P)
select Select a spectrum without displaying it (C)
spadd Add current spectrum to add/subtract experiment (C)
sub Subtract current FID from add/subtract experiment (C)

addfids **Add a series of FIDs together (M)**

Applicability: Systems with LC-NMR accessory.

Syntax: `addfids<(start, finish)>`

Description: Improves signal-to-noise by adding adjacent FIDs that represent the same peak. Given a series of FIDs that represent separate data, such as occur during an LC-NMR run, some of the adjacent FIDs can actually represent the same peak in the LC run.

To obtain the FID numbers to use, you can enter `dss` or `dsw` (e.g., enter `dsw(25, 35)` and then determine that peak numbers 28 to 31 contain the peaks of interest), or you can enter `dconi` and then read the Index counter on line 1 of the display.

Arguments: `start` is the number of the first FID to be co-added. The default is that you are prompted for the value.

`finish` is the number of the last FID to be co-added. The default is that you are prompted for the value.

Examples: `addfids`
`addfids(25, 28)`

See also: *User Guide: Liquids NMR*

addi **Start interactive add/subtract mode (C)**

Syntax: `addi`

Description: Starts the interactive add/subtract mode. Before entering `addi`, start the process with `clradd` and `spadd`, then display a second spectrum on the screen. This may involve changing experiments, selecting a second member of an array of spectra, a different trace of a 2D spectrum, or displaying a spin simulated spectrum. The Fourier numbers (`fn`) *must* be the same in the two spectra to be manipulated. The width (`sw`) of the two spectra need *not* be identical, although adding spectra of different widths will probably not be meaningful. Having selected the second spectrum and ensuring it is in `nm` mode, enter `addi` to begin the interactive process.

After `addi` is invoked, spectrum 1, the spectrum selected by the `spadd` command, appears in the center of the display. Spectrum 2, the spectrum that was active when `addi` was entered, appears on the bottom. The sum or difference of these spectra appears on top of the screen. When `addi` is first entered, this spectrum will be the sum (1 + 2) by default. The spectra is manipulated using the mouse.

The select button toggles between different modes of control.

- When the label at the screen bottom reads “active: current”, all of the parameters (except `wp`) control spectrum 2, and spectrum 2 can be phased, scaled, or shifted relative to spectrum 1.
- After clicking on select, the label at the screen bottom reads “active: addsub”, and now all of the parameters except `wp` control spectrum 1.
- Clicking select again toggles the label to read “active: result”, and now parameter changes affect only the sum or difference spectrum.

Note that `wp` always controls all spectra, because differential expansions of the two spectra are not supported. Note also that the colors of the labels change to match the colors of the different spectra.

The sum/difference spectrum displayed on the screen while `addi` is active is strictly a temporary display. Once all manipulations have been performed, and

assuming the sum/difference is something you wish to perform further operations with (such as plotting), it must be saved into the add/subtract experiment (`exp5`) by clicking on save. At this point, spectrum 1, which was in the add/subtract experiment, is overwritten by the sum or difference spectrum, and `addi` ceases operation. In most cases, you will next want to enter `jexp5 ds` to display the difference spectrum on the screen, ready for further manipulation (expansion, line listing, etc.) and plotting. If you wish to continue with the add/subtract process by adding in a third spectrum, display that spectrum in the usual way and enter `addi` again.

Alternate: Interactive Mode button in the Add/Subtract Menu.
Add/Subtract button in the Deconvolution Menu.

See also: *User Guide: Liquids NMR*

Related:	<code>clradd</code>	Clear add/subtract experiment (C)
	<code>jexp</code>	Join existing experiment (C)
	<code>nm</code>	Select normalized intensity mode (C)
	<code>spadd</code>	Add current spectrum to add/subtract experiment (C)
	<code>spmin</code>	Take minimum of two spectra in add/subtract experiment (C)
	<code>spsub</code>	Subtract current spectrum from add/subtract experiment (C)
	<code>wp</code>	Width of plot in directly detected dimension (P)

addnucleus **Add new nucleus to existing probe file (M)**

Syntax: `addnucleus<(nucleus)>`

Description: Appends entries for nuclei not in the default probe file to the end of the file.

Arguments: If no argument is entered, a prompt is displayed requesting the nucleus entry.
`nucleus` is a nucleus entry in the `nuctable`.

Examples: `addnucleus`
`addnucleus('Si29')`

See also: *Getting Started*

Related:	<code>addprobe</code>	Create new probe directory and probe file (M)
	<code>getparam</code>	Receive parameter from probe file (M)
	<code>probe</code>	Probe type (P)
	<code>setparams</code>	Write parameter to current probe file (M)

addpar **Add selected parameters to current experiment (M)**

Syntax: `addpar<('2d'|'3d'|'3rf'|'4d'|'downsamp'|'fid'|
'image'|'l12d'|'lp'<,dim>|'oversamp'|'ss')>`

Applicability: The `'3d'`, `'3rf'`, `'4d'`, `'fid'`, and `'image'` arguments work on all systems but are only useful if system has the proper hardware.

Description: Creates selected parameters in the current experiment.

Arguments: If no argument is entered, `addpar` displays instructions for its use.
`'2d'`, `'3d'`, `'3rf'`, `'4d'`, `'downsamp'`, `'fid'`, `'image'`, `'l12d'`, `'lp'`, `'oversamp'`, and `'ss'` are keywords (only one keyword is used at a time) specifying the parameters to be created:

- `'2d'` specifies creating `ni`, `phase`, and `sw1`, which can be used to acquire a 2D data set (functions the same as macro `par2d`).
- `'3d'` specifies creating `d3`, `ni2`, `phase2`, and `sw2`, which can be used to acquire a 3D data set (functions the same as macro `par3d`).

- '3rf' specifies retrieving the `ap` and `dg2` display templates for third rf channel and 3D parameters (functions the same as macro `par3rf`).
- '4d' specifies creating the acquisition parameters `d4`, `ni3`, `phase3`, and `sw3`, which can be used to acquire a 4D data set (functions the same as macro `par4d`).
- 'downsamp' specifies creating the parameters `downsamp`, `dscoef`, `dslsfrq`, `dsfb`, and `filtfile` for digital filtering and downsampling (functions the same as macro `pards`).
- 'fid' specifies creating FID display parameters `axisf`, `crf`, `deltaf`, `dotflag`, `vpf`, and `vpfi` if the parameter set is older and lacks these parameters (functions the same as macro `fidpar`).
- '112d' specifies creating `th2d` and `xdiag` for the `112d` 2D peak picking program (functions the same as macro `par112d`).
- 'lp' specifies creating `lpalg`, `lpopt`, `lpfilt`, `lpnupts`, `strtlp`, `lpext`, `strtext`, `lptrace`, and `lpprint` for linear prediction in the acquisition dimension (functions the same as macro `parlp`). The display template for the `dg1p` macro is also created if necessary.
- 'oversamp' specifies creating parameters `def_osfilt`, `filtfile`, `oscoef`, `osfb`, `osfilt`, `oslsfrq`, and `oversamp` for oversampling and digital filtering (functions the same as macro `paros`).
- 'ss' specifies adding parameters `ssorder`, `ssfilter`, `ssntaps`, and `sslsfrq` for time-domain solvent subtraction (functions the same as macro `parfidss`).

`dim` specifies the dimension when adding linear prediction parameters: 1 for the first implicit dimension or 2 for the second implicit dimension. Default is the acquisition dimension. Therefore, `addpar('lp')` creates the parameters listed above; `addpar('lp',1)` creates `lpalg1`, `lpopt1`, `lpfilt1`, `lpnupts1`, `strtlp1`, `lpext1`, `strtext1`, `lptrace1`, and `lpprint1`; and `addpar('lp',2)` creates `lpalg2`, `lpopt2`, `lpfilt2`, `lpnupts2`, `strtlp2`, `lpext2`, `strtext2`, `lptrace2`, and `lpprint2`. Each separate dimension of a multidimensional data set can have its own unique parameters.

Examples: `addpar`
`addpar('3d')`
`addpar('lp',1)`

See also: *Getting Started; User Guide: Liquids NMR; User Guide: Imaging*

Related:	<code>def_osfilt</code>	Default value of <code>osfilt</code> (P)
	<code>dg1p</code>	Display group of linear prediction parameters (M)
	<code>fidpar</code>	Add parameters for FID display in current experiment (M)
	<code>osfilt</code>	Oversampling filter for real-time DSP (P)
	<code>par2d</code>	Create 2D acquisition parameters (M)
	<code>par3d</code>	Create 3D acquisition parameters (M)
	<code>par3rf</code>	Get display templates for 3rd rf channel parameters (M)
	<code>par4d</code>	Create 4D acquisition parameters (M)
	<code>pards</code>	Create digital filtering and downsampling parameters (M)
	<code>parfidss</code>	Set up parameters for time-domain solvent subtraction (M)
	<code>paros</code>	Create oversampling and digital filtering parameters (M)
	<code>par112d</code>	Create parameters for 2D peak picking (M)
	<code>parlp</code>	Create parameters for linear prediction (M)

addparams Add parameter to current probe file (M)

Syntax: `addparams (param,value,nucleus<,'tmplt'><,'system'>)`

Description: Adds a new parameter and its value for a specified nucleus to the probe file or to the probe template.

Arguments: `param` is the name of the parameter to be added.

`value` is a string with the value to be written for the parameter.

`nucleus` is the nucleus to add in the probe file.

'`tmplt`' is a keyword to add the parameter to the local template. The default is the probe file.

'`system`' is a keyword to add the parameter to the system-level template or probe file, provided that you have write permission to that file. The default is to add the parameter to the local template or probe file.

Examples: `addparams('ref_pwr','53',tn)`
`addparams('ref_pwx','00',dn,'tmplt')`
`addparams('ref_pwx2','00',dn2,'tmplt','system')`

See also: *Getting Started*

Related: `getparam` Receive parameter from probe file (M)
`setparams` Write parameter to current probe file (M)
`updateprobe` Update probe file (M)

addprobe Create new probe directory and probe file (M)

Syntax: `addprobe (probe_name<,'stdar'|'system'><,'stdpar'>)`

Description: Creates a new probe directory and a probe file. Default nuclei included in this file are ^1H , ^{19}F , ^{13}C , and ^{15}N . The information is saved in the user's directory `vnmr/sys/probes`.

Arguments: `probe_name` is the name to be given to the probe directory and probe file.

'`stdpar`' and '`system`' are keywords for the second and third arguments:

- If the second argument is '`stdpar`', calibration values from the standard parameter sets (`stdpar/H1.par`, `stdpar/C13.par`, etc.) will be read and written into the probe file.
- If the second argument is '`system`' and the user has write permission into the VNMR system probes directory (typically `/vnmr/probes`), then a system-level probe directory will be made.
- If the second argument is '`system`' and the third argument is '`stdpar`', then both actions in the preceding bullets will occur.
- The default is the probe file is created with all parameters initialized to zero.

Examples: `addprobe('idpfg')`
`addprobe('idpfg','stdpar')`
`addprobe('idpfg','system','stdpar')`

See also: *Getting Started; Walkup NMR Using GLIDE*

Related: `addnucleus` Add new nucleus to existing probe file (M)
`getparam` Receive parameter from probe file (M)
`probe` Probe type (P)
`setparams` Write parameter to current probe file (M)

adept Automatic DEPT analysis and spectrum editing (C)

Syntax: `adept<(<'noll'><,<'coef'><,<'theory'>>>`

Description: Automatically analyzes a set of four DEPT spectra and edits the spectra so that the spectra is arrayed as follows:

- #4 is CH₃ carbons only
- #3 is CH₂ carbons only
- #2 is CH carbons only
- #1 is all protonated carbons

Because `adept` modifies the transformed data, it should not be repeated without retransforming the data between calls. `adept` produces a text file `dept.out` in the current experiment directory, which contains the result of the analysis.

Arguments: The following keyword arguments can be supplied in any order:

'noll' causes the line listing to be skipped. If 'noll' is not supplied as an argument, `adept` first performs a line listing. In that case, the threshold parameter `th` must be set properly before starting `adept`.

'coef' causes the combination coefficients to be printed.

'theory' causes theoretical coefficients to be used. The default is optimized coefficients.

Examples: `adept`
`adept('coef')`
`adept('theory','noll')`

See also: *User Guide: Liquids NMR*

Related:	<code>autodept</code>	Automated complete analysis of DEPT data (M)
	<code>deptproc</code>	Process DEPT data (M)
	<code>padept</code>	Perform <code>adept</code> analysis and plot resulting spectra (C)
	<code>pldept</code>	Plot DEPT data, edited or unedited (M)
	<code>th</code>	Threshold (P)

addrcvrs Combine data from multiple receivers (M)

Applicability: Imaging systems with multiple receivers.

Syntax: `addrcvrs`

Description: Combines image data that has been acquired by multiple receivers. First transforms the data from each receiver separately with `wft2d`. Weights the individual images by the factors specified in the `rcvrwt` parameter and forms the RMS average.

Examples: `addrcvrs`

Related:	<code>rcvrwt</code>	Weighting for different receivers (M)
	<code>wft2d</code>	Weight and Fourier Transform 2D data (C)
	<code>rmsAddData</code>	Add transformed data files with weighting (U)

aexpp1 Automatic plot of spectral expansion (M)

Syntax: `aexpp1<(expansion_factor)>`

Description: Plots automatically expansions of given regions. Regions have to be defined first by using the `region` command or by using the cursors in `ds`.

Arguments: `expansion_factor` is a spectral expansion factor in units of Hz/mm. The default is 2 Hz/mm.

Examples: `aexpp1`
`aexpp1 (20)`

See also: *Getting Started*

Related: `ds` Display a spectrum (C)
`region` Divide spectrum into regions (C)

ai **Select absolute-intensity mode (C)**

Syntax: `ai`

Description: Selects the *absolute-intensity display mode* in which the scale is kept constant from spectrum to spectrum to allow comparison of peak heights from one spectrum to another. The alternative is the *normalized-intensity display mode* (`nm`) in which spectra are scaled so that the largest peak in the spectrum is `vs` mm high. The modes are mutually exclusive—the system is always in either `nm` or `ai` mode. Enter `aig?` to determine which mode is currently active.

See also: *Getting Started*

Related: `aig` Absolute intensity group (P)
`nm` Select normalized-intensity mode (C)
`vs` Vertical scale (P)

aig **Absolute-intensity group (P)**

Description: Contains the result of the `ai` or `nm` command. `aig` is not set in the usual way but can be queried (`aig?`) to determine which display mode is active.

Values: 'ai' indicates the absolute-intensity display mode is active.
'nm' indicates the normalized-intensity display mode is active.

See also: *Getting Started*

Related: `ai` Select absolute intensity mode (C)
`dmg` Display mode in directly detected dimension (P)
`nm` Select normalized-intensity mode (C)
`?` Display individual parameter value (C)

alfa **Set alfa delay before acquisition (P)**

Description: After the final event in the pulse sequence, including any receiver gate times occurring following the final pulse, acquisition occurs after a delay. This delay includes a fixed part, `alfa`, and a variable part, $1 / (\text{beta} * \text{fb})$.

- On *GEMINI 2000* $^1\text{H}/^{13}\text{C}$ systems, `beta` is 3.
- On *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* broadband systems, `beta` is 2.
- On systems with 4-pole Butterworth filters, `beta` is 2.
- On systems with 8-pole Butterworth (200-kHz) filters, `beta` is 3.8.
- On systems with 8-pole elliptical filters, `beta` is 1.29.
- On *UNITYINOVA* and *UNITYplus* with 4-pole Bessel filters or *UNITY* systems with 6-pole Bessel filters, `beta` is 2.3 (only systems with 2-MHz and 5-MHz Analog-to-Digital Converter boards use this filter).

Because the total delay before acquisition is the sum of `alfa` and $1 / (\text{beta} * \text{fb})$, it is possible to shorten the delay beyond “normal” values by setting `alfa` negative (to a maximum of $1 / (\text{beta} * \text{fb})$). The macros `hoult` and `calfa` frequently result in such negative values of `alfa`.

To set `alfa` to a negative number, use either the `setvalue` command to enter a specific value of `alfa`, or use the `setlimit` command to allow entry of negative values of `alfa` directly from the keyboard.

Values: 0 to 100,000,000; in μ s.

See also: *Getting Started*

Related:	<code>calfa</code>	Recalculate <code>alfa</code> so that first-order phase is zero (M)
	<code>fb</code>	Filter bandwidth (P)
	<code>hoult</code>	Set parameters <code>alfa</code> and <code>rof2</code> according to Hoult (M)
	<code>rof2</code>	Receiver gating time following pulse (P)
	<code>setlimit</code>	Set limits of a parameter in a tree (C)
	<code>setvalue</code>	Set value of any parameter in a tree (C)

alock **Automatic lock control (P)**

Description: Governs Autolock control following the insertion of a sample with `change` or `sample`, and following initiation of an acquisition with the `go`, `ga`, or `au`. Manual adjustment of lock power, gain, and phase is possible using the `acqi` command. On UNITY and VXR-S systems, switching between simple (hardware) Autolock and simple lock is possible with buttons in the Acquisition window.

Values: Possible values are 'a', 'auto', 'n', 's', 'samp', 'u', or 'y', where:

'a' or 'auto' selects the optimizing Autolock function, which performs a lock capture and an automatic lock power and gain adjustment before data acquisition begins (lock phase is *not* optimized).

'n' leaves the lock in its current state.

's' or 'samp' selects the optimizing Autolock function, which performs a lock capture and an automatic lock power and gain adjustment before data acquisition begins (lock phase is *not* optimized) but only if the sample has just been changed.

On ^{UNITY}INOVA, UNITY*plus*, UNITY, and VXR-S, 'u' turns lock off so that the experiment runs unlocked. On GEMINI 2000, 'u' is inoperative.

On ^{UNITY}INOVA, UNITY*plus*, and GEMINI 2000, 'y' turns on the software Autolock function, which searches for the correct Z0 value only.

On UNITY and VXR-S, 'y' turns on the hardware Autolock function, with lock power, lock gain, and lock phase not adjusted.

See also: *Getting Started*

Related:	<code>acqi</code>	Interactive acquisition display process (C)
	<code>au</code>	Submit experiment to acquisition and process data (C)
	<code>change</code>	Submit a change sample experiment to acquisition (M)
	<code>ga</code>	Submit experiment to acquisition and FT the result (C)
	<code>gf</code>	Prepare parameters for FID/spectrum display in <code>acqi</code> (M)
	<code>go</code>	Submit experiment to acquisition (C)
	<code>lock</code>	Submit an Autolock experiment to acquisition (C)
	<code>sample</code>	Submit change sample, Autosim experiment to acquisition (M)

ampmode **Independent control of amplifier mode (P)**

Applicability: ^{UNITY}INOVA and UNITY*plus* systems.

Description: Gives override capability over the default selection of amplifier modes. Unless overridden, the usage of rf channels determines whether the amplifier for a channel is in pulse, CW (continuous wave), or idle mode:

- Observe channel is set to the pulse mode.
- Other used channels are set to the CW mode.
- Any unused channels are set to the idle mode.

The `ampmode` parameter can be used to override this selection.

`ampmode` does not normally exist but can be created by the user with the command `create('ampmode', 'flag')`.

Values: List of characters in which the mode of the first amplifier is determined by the first character, the mode of the second amplifier by the second character, and so on. For each amplifier, one of the following characters is used:

- 'c' selects CW mode.
- 'i' selects idle mode.
- 'p' selects pulse mode.
- 'd' selects default behavior.

For example, `ampmode='ddp'` selects default behavior for the first two amplifiers and forces the third channel amplifier into pulse mode. Additional filtering is usually required when an amplifier in the same band as the observe amplifier is placed in the CW mode.

See also: *VNMR User Programming*

Related: `create` Create new parameter in a parameter tree (C)
`dn` Nucleus for the first decoupler (P)
`tn` Nucleus for observe transmitter (P)

amptype **Amplifier type (P)**

Applicability: All systems except *GEMINI 2000*.

Description: Specifies the type of amplifier on each rf channel of the spectrometer. The value is set in the CONFIG window (opened from `config`) using the label Type of Amplifier.

On ^{UNITY}*INOVA*, *UNITYplus*, *UNITY*, and *VXR-S* systems, for each channel, the types are Class C, Linear Full Band, Linear Low Band, Linear Broadband, or, for the fourth channel only, Shared. Selecting Shared means that the amplifier is fully configured for the third channel, and that the fourth channel shares this amplifier with the third channel.

When a type is selected for a channel, a letter (one of the values described below) is added to the value of `amptype`. For example, a system already set to Linear Full Band on the observe transmitter channel and the first decoupler channel would have `amptype='aa'`. Selecting the third channel as Linear Low Band would set `amptype='aal'`. Finally, selecting Shared for the fourth channel would set `amptype='aaln'`.

Values: On ^{UNITY}*INOVA*, *UNITYplus*, *UNITY*, and *VXR-S* Systems:

'a' indicates the channel uses a linear full-band amplifier. A full-band amplifier has two outputs: 12 MHz to ³¹P, and ¹⁹F/¹H.

'b' indicates the system uses a linear broadband amplifier.

'c' indicates the system uses a class C amplifier.

'l' indicates the channel uses a linear low-band amplifier. A low-band amplifier has one output from 12 MHz to ³¹P only.

'n' indicates the fourth channel shares a linear amplifier with the third.

See also: *VNMR and Solaris Software Installation, 3rd and 4th RF Channel Installation*

Related: `config` Display current configuration and possibly change it (M)

analyze **Generalized curve fitting (C)**

Syntax: (curve fitting) `analyze('expfit', xarray<, options>)`
 (regression) `analyze('expfit', 'regression'<, options>)`

Description: Provides interface to curve fitting program `expfit` (using the curve fitting syntax), supplying `expfit` with input data in the form of the text file `analyze.inp` in the current experiment. `expfit` can be called from UNIX with the syntax:

```
expfit options <analyze.inp >analyze.list
```

`expfit` does a least-squares curve fitting to the data supplied in `analyze.inp`. Macros are available for the specialized uses of `analyze`, such as the 'T1' and 'kinetics' options. These macros avoid the need to select options and get the correct file format.

In the regression mode (using the regression syntax above), the type of curve fitting, ('poly1', ...) must be selected. The regression section in the manual *User Guide: Liquids NMR* gives the input file format and describes the menus that permit choices indirectly through menu buttons.

The text file `analyze.inp` for the options 'T1', 'T2', 'kinetics', 'contact_time', and 'regression' contains the following lines (note that (1), (2), (3), etc. do not appear in the file but are used to identify lines in the explanation):

```
(1) <text line>
(2) <text line>
(3) npeaks npairs <xscale> <yscale>
(4) <NEXT npairs1>
(5) peaks
(6) x y
(6) x y
...
(4) <NEXT npairs2>
(5) peaks
(6) x y
(6) x y
...
```

Line-by-line explanation:

- (1) Optional descriptive text line, for regression only. Omit line otherwise.
- (2) Optional y-axis title, for regression only. Omit line otherwise.
- (3) Line containing an integer for the number of peaks (`npeaks`) followed by another integer for the number of (`x`, `y`) pairs per peak (`npairs`). If regression, the `x`-scale type and `y`-scale type are also listed.
- (4) In the regression mode, a line beginning with the keyword `NEXT` is inserted at the start of each data set when the number of pairs per peak is variable. In this case, the number of (`x`, `y`) pairs for the peak (`npair1`, `npair2`, etc.) is also given on the line.
- (5) Peak index.
- (6) Data pairs, one to a line, are listed by peak in the following order:

```

x y    (first peak, first pair)
x y    (first peak, second pair)
...
x y    (second peak, first pair)
x y    (second peak, second pair)
...

```

In the regression mode, the line beginning with `NEXT` is inserted at the start of the data for each peak when the number of pairs per peak is variable. In this case, the header contains the maximum number of pairs for any peak.

For `'T1'`, `'T2'`, `'kinetics'`, and `'contact_time'`, information from the file `fp.out` and values of the arrayed parameter `xarray` are used to construct the file; thus, it is necessary to run `fp` prior to `analyze`.

For regression, `analyze.inp` is made by running `expl('regression')`. If the regression mode is not selected, `analyze.inp` may be slightly different.

In addition to output to the standard output, which is usually directed to `analyze.list`, `expfit` makes a file `analyze.out`, which is used by `expl` to display the results of the analysis.

User-supplied analysis programs can be called by `analyze` in place of `expfit`. Such programs should read their input from `stdin` and write the output listing to `stdout`. No `analyze.out` file needs to be generated unless display by `expl` is desired. Use the program `expfit` as a model.

Arguments: `'expfit'` is a required first argument.

`xarray` is the name of the parameter array holding x-values in `'T1'`, `'T2'`, `'kinetics'`, and `'contact_time'`, and is used only with these options.

`'regression'` sets regression mode and signifies generalized curve fitting with choices `'poly1'`, `'poly2'`, `'poly3'`, and `'exp'`.

options are any of the following keywords:

- `'T1'` sets T_1 analysis (the default).
- `'T2'` sets T_2 analysis.
- `'kinetics'` sets kinetics analysis, with decreasing peak height.
- `'increment'` sets kinetics analysis, with increasing peak height.
- `'list'` makes an extended listing for each peak.
- `'diffusion'` sets a special analysis for diffusion experiments.
- `'contact_time'` sets a special analysis for solids cross-polarization spin-lock experiments.
- `'poly1'` sets a linear fitting. It is used in regression mode only.
- `'poly2'` sets a quadratic fitting. It is used in regression mode only.
- `'poly3'` sets a cubic fitting. It is used in regression mode only.
- `'exp'` sets exponential curve fitting. It is used in regression mode only.

Examples: `analyze('expfit','d2','T1','list')`
`analyze('expfit','pad','kinetics','list')`
`analyze('expfit','p2','contact_time','list')`
`analyze('expfit','regression','poly1','list')`

See also: *User Guide: Liquids NMR*

Related: `contact_time` MAS cross-polarization spin-lock contact time (M)
`expfit` Least squares fit to polynomial or exponential curve (U)
`expl` Display exponential or polynomial curves (C)

<code>pexpl</code>	Plot exponential or polynomial curves (C)
<code>kini</code>	Kinetics analysis, increasing intensity (M)
<code>t1</code>	T_1 exponential analysis (M)
<code>t2</code>	T_2 exponential analysis (M)

ap Print out “all” parameters (C)

Syntax: `ap<(template)>`

Description: Prints a parameter list containing “all” parameter names and values.

Arguments: `template` is the name of the template. The default is a template controlled by the parameter `ap`, which can be modified with the command `paramvi('ap')`. See the manual *VNMR User Programming* for rules on building a template.

Examples: `ap`
`ap('newap')`

See also: *Getting Started; VNMR User Programming*

Related:	<code>addpar</code>	Add selected parameters to the current experiment (M)
	<code>ap</code>	“All” parameters display control (P)
	<code>dg</code>	Display group of acquisition/processing parameters (C)
	<code>hpa</code>	Plot parameters on special preprinted chart paper (C)
	<code>pap</code>	Plot out “all” parameters (C)
	<code>paramvi</code>	Edit a variable and its attributes with <code>vi</code> text editor (C)
	<code>ppa</code>	Plot a parameter list in “English” (M)

ap “All” parameters display control (P)

Description: Controls the display of the `ap` and `pap` commands to print and plot a parameter list. Use `paramvi('ap')` to modify the string value of `ap`.

See also: *Getting Started; VNMR User Programming*

Related:	<code>ap</code>	Print out “all” parameters (C)
	<code>dg</code>	Display group of acquisition/processing parameters (C)
	<code>pap</code>	Plot out “all” parameters (C)
	<code>paramvi</code>	Edit a variable and its attributes with <code>vi</code> text editor (C)

apa Plot parameters automatically (M)

Syntax: `apa`

Description: Selects automatically the appropriate command on different plotter devices to plot the parameter list. For example, `apa` generates a `ppa` on Zeta plotter or an `hpa` on an Hewlett-Packard plotter.

See also: *VNMR User Programming*

Related:	<code>hpa</code>	Plot parameters on special preprinted chart paper (C)
	<code>ppa</code>	Plot a parameter list in “English” (M)

aph Automatic phase adjustment of spectra (C)

Syntax: `aph<:$ok,$rp,$lp>`

Description: Automatically calculates the phase parameters `lp` and `rp` required to produce an absorption mode spectrum and applies these parameters to the current spectrum. Values calculated do *not* depend on the initial values of `lp` and `rp`.

Arguments: `$ok` is 1 if the phase adjustment succeeds, or 0 if the adjustment fails.

$\$rp$ is the calculated value of rp . If $\$rp$ is requested as a return value, rp is returned but not applied to the current spectrum.

$\$lp$ is the calculated value of lp . If $\$lp$ is requested as a return value, lp is returned but not applied to the current spectrum.

Alternate: Autophase button in the 1D Data Manipulation Menu.

See also: *Getting Started*

Related: **aph0** Automatic phase of zero-order term (C)
aphx Perform optimized automatic phasing (M)
lp First-order phase in directly detected dimension (P)
rp Zero-order phase in directly detected dimension (P)

aph0 Automatic phase of zero-order term (C)

Syntax: `aph0 < :$ok, $rp, $lp >`

Description: Automatically adjusts only the zero-order frequency-independent term rp and does not rely on the frequency-dependent term lp being previously adjusted. In favorable circumstances, spectra may be obtained in such a way that only rp is expected to change. In these cases, if lp has been determined for one spectrum, then rp only can be computer-adjusted for subsequent spectra by `aph0` (“aph-zero”). Note that `aph0` does not correctly phase an exactly on-resonance peak.

Arguments: $\$ok$ is 1 if the phase adjustment succeeds, or 0 if the adjustment fails.

$\$rp$ is the calculated value of rp .

$\$lp$ is the current value of lp .

See also: *Getting Started*

Related: **aph** Automatic phase adjustment of spectra (C)
aphx Perform optimized automatic phasing (M)
lp First-order phase in directly detected dimension (P)
rp Zero-order phase in directly detected dimension (P)

aphb Auto phasing for Bruker data (C)

Syntax: `aphb < (threshold) >`

Description: Phases Bruker data using the autophasing program.

Arguments: `threshold` determines if a data point is large enough to qualify it as part of a peak. If no argument is given, or if the value is equal to or less than 0, the threshold is calculated from the spectrum.

Examples: `aphb`
`aphb (2)`

See also: *Getting Started*

Related: **aph** Automatic phase adjustment of spectra (C)
aph0 Automatic phase of zero-order term only (C)

aphx Perform optimized automatic phasing (M)

Syntax: `aphx`

Description: Optimizes parameters and arguments for the `aph` command. `aphx` first performs an `aph` then calculates a theoretical value for lp . If lp set by the `aph` is different from the calculated value by 10 per cent, the calculated value is used and an `aph0` is performed.

See also: *Getting Started*

Related: **aph** Automatic phase adjustment of spectra (C)
aph0 Automatic phase of zero-order term only (C)
lp First order phase along directly detected dimension (P)

apinterface **AP Interface board type (P)**

Applicability: UNITY and VXR-S systems. On ^{UNITY}*INOVA* and *UNITYplus* systems, **apinterface** does not apply, and the value of the label AP Interface Type should be set to N/A in the CONFIG window. **apinterface** does not apply to *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Description: Sets the type of AP Interface board on UNITY and VXR-S systems. The system value is set within the CONFIG window (opened from **config**) using the label AP Interface Type.

Values: 1 is for systems with the older XL Interface board (Type 1 in CONFIG window). This board is present on 200-MHz through 400-MHz systems with class C amplifiers and on early systems with ENI and/or TPL linear amplifiers (mostly VXR-500 systems but also including some others).
 2 is for systems with the newer AP Interface board present on all 200- through 600-MHz systems configured with linear amplifiers and can include systems with fine attenuators (Type 2 in CONFIG window).
 3 is for systems with an AP Interface board that contains additional control lines to enable setting the decoupler modulation mode with the AP bus instead of with high-speed lines (Type 3 in CONFIG window).

See also: *VNMR and Solaris Software Installation*

Related: **config** Display current configuration and possibly change it (M)

appmode **Application mode (P)**

Applicability: All systems except *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Description: A global parameter that allows selection of specialized system applications modes, such as imaging, by setting the global parameters **sysmaclibpath**, **sysmenulipath**, and **syshelpath**.

For example, in **/vnmr/maclib** is a subdirectory **maclib.imaging** that contains macros used primarily with imaging applications. Similarly, in **/vnmr/menulib** is a subdirectory **menulib.imaging** for imaging-related menus. By separating the imaging macros and menus into subdirectories, access to imaging-specific macros and menus is more convenient. This separation also allows minor modifications to some macros and menus while retaining the names that are in common use or required by other VNMR commands.

The **dconi** menu illustrates how **appmode** works. In normal 2D spectroscopy operation (defined by setting **appmode= 'standard'**), the **dconi** menu displays a button labeled **Peak** that provides access to interactive 2D peak-picking. With **appmode= 'imaging'**, however, this button is labeled **Mark** instead and now performs the 1D or 2D **mark** function, which is more appropriate for imaging data. The **dconi** menu tailored for imaging is found in **menulib.imaging**, which is searched by VNMR before searching the **/vnmr/menulib** directory when **appmode** is set to **'imaging'**. The search order is **userdir+ ' / ? '** followed by **vnmrsys/maclib**, **maclibpath**, **sysmaclibpath**, and then **/vnmr/maclib**.

The value of **appmode** can be set either by entering its value directly from the command line or by selecting the 2:Setup button from the Main Menu and then

clicking on the 5:App Mode button. New applications modes can be added by creating the appropriate subdirectories in `/vnmr/macclib`, `/vnmr/menulib`, and `/vnmr/help`, and adding the desired applications mode name to the `_appmode` macro. Subdirectories should be named by adding the file extension `.appmodename` to the corresponding parent directory name (e.g., `macclib.solids`, `menulib.automation`).

Values: 'standard' sets standard application mode.

'imaging' sets imaging application mode.

Alternate: App Mode button in the Setup Menu.

See also: *VNMR and Solaris Software Installation; User Guide: Imaging*

Related: `config` Display current configuration and possibly change it (M)

apt Set up parameters for APT pulse sequence (M)

Syntax: `apt<(solvent)>`

Description: Converts a parameter set to the APT (attached proton test) experiment.

Arguments: `solvent` is the name of the solvent used. The default for `solvent` is `CDCl3` or, if in the automation mode, the default is read from the file `sampleinfo`.

Alternate: APT button in the 1D Pulse Sequence Setup Menu.

See also: *User Guide: Liquids NMR*

Related: `aptaph` Automatic processing for APT spectra (M)

`capt` Automated carbon and APT acquisition (M)

`hcapt` Automated proton, carbon, and APT acquisition (M)

APT Change parameters for APT experiment (M)

Syntax: `APT<('GLIDE')>`

Description: Converts the current parameter set to an APT experiment.

Arguments: 'GLIDE' is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding carbon spectrum for the experiment..

Related: `apt` Set up parameters for APT experiment (M)

aptaph Automatic processing for APT spectra (M)

Syntax: `aptaph`

Description: Automatically phases APT spectra.

See also: *User Guide: Liquids NMR*

Related: `apt` Set up parameters for APT pulse sequence (M)

arccos Calculate arc cosine of real number (M)

Applicability: Systems with imaging capabilities.

Syntax: `arccos(x<, 'silent'>)<:rad,deg>`

Description: Calculates the arc cosine value of a real number. The answer is given, in radians and degrees, in the top VNMR display window and is optionally returned to two destination variables. The calculation is based on the identity $\arccos(x) = \arctan(\sqrt{1-x^2}/x)$. Since `arccos` calls the macro `arctan` rather than the built-in math function `atan`, the calculation is somewhat slow.

Arguments: x is a real number in the range of ± 1.0 .
 'silent' is a keyword to suppress the display of the results in the top VNMR display window.
 rad is a return value in radians.
 deg is a return value in degrees.

Examples: `arccos(.5)`
`arccos(-.2, 'silent'):r1,d1`

See also: *User Guide: Imaging*

Related: **acos** Find arc cosine of number (C)
arcsin Calculate arc sine of a real number (M)
arctan Calculate arc tangent of a real number (M)
atan Find arc tangent of a number (C)

arcsin Calculate arc sine of real number (M)

Applicability: Systems with imaging capabilities.

Syntax: `arcsin(x<, 'silent'><:rad,deg>`

Description: Calculates the arc sine value of a real number. The answer is given, in radians and degrees, in the top VNMR display window and is optionally returned to two destination variables. The calculation is based on the identity $\arcsin(x) = \arctan(x/\sqrt{1-x^2})$. Since *arcsin* calls the macro **arctan** rather than the built-in math function **atan**, the calculation is somewhat slow.

Arguments: x is a real number in the range of ± 1.0 .
 'silent' is a keyword to suppress the display of the results in the top VNMR display window.
 rad is a return value in radians.
 deg is a return value in degrees.

Examples: `arcsin(.5)`
`arcsin(-.2, 'silent'):r1,d1`

See also: *User Guide: Imaging*

Related: **arccos** Calculate arc cosine of a real number (M)
arctan Calculate arc tangent of a real number (M)
asin Find arc sine of number (C)
atan Find arc tangent of a number (C)

arctan Calculate arc tangent of real number (M)

Applicability: Systems with imaging capabilities.

Syntax: `arctan(x<, 'silent'><:rad,deg>`

Description: Calculates the arc tangent value of a real number. The answer is given, in radians and degrees, in the top VNMR display window and is optionally returned to two destination variables. The calculation is based on a rational approximation.

Arguments: x is a real number.
 'silent' is a keyword to suppress the display of the results in the top VNMR display window.
 rad is a return value in radians.

deg is a return value in degrees.

Examples: `arctan(.5)`
`arctan(-.2, 'silent'):r1,d1`

See also: *User Guide: Imaging*

Related: **arccos** Calculate arc cosine of a real number (M)
arcsin Calculate arcsine of a real number (M)
asin Find arc sine of number (C)
atan Find arc tangent of a number (C)

array Easy entry of linearly spaced array values (M)

Syntax: `array<(parameter<,number_steps,start,step_size)>`

Description: Arrays a parameter to the number of steps, starting value and step size given by the user. All values of the array will satisfy the limits of the parameter.

If `array` is typed with none or only some of its arguments, you enter an interactive mode in which you are asked for the missing values.

Arguments: `parameter` is the name of the parameter to be arrayed. The default is an interactive mode in which you are prompted for the parameter. Only numeric parameters can be arrayed.

`number_steps` is the number of values of the parameter. The default is an interactive mode in which you are prompted for the number of steps.

`start` is the starting value of the parameter array. The default is an interactive mode in which you are prompted for the starting value.

`step_size` is the magnitude of the difference between elements in the array. The default is an interactive mode in which you are prompted for the step size.

Examples: `array`
`array('pw')`
`array('tof',40,1400,-50)`

See also: *User Guide: Liquids NMR*

array Parameter order and precedence (P)

Description: Whenever an array of one or more parameters is set up, the string parameter `array` tells the system the name of the parameter or parameters that are arrayed and the order and precedence in which the arraying is to take place. The parameter `array` is automatically updated when acquisition parameters are set. "Diagonal arrays" (those corresponding to using parentheses in the parameter `array`) must be entered by hand.

Values: `' '` (two single quotes with no space between) indicates no parameter is arrayed.

`'x'` indicates the parameter `x` is arrayed.

`'x,y'` indicates the parameters `x` and `y` are arrayed, with `y` taking precedence. That is, the order of the experiments is $x_1y_1, x_1y_2, \dots, x_1y_n, x_2y_1, x_2y_2, \dots, x_2y_n, \dots, x_my_n$, with a total of $m \times n$ experiments being performed.

`'y,x'` indicates the parameters `x` and `y` are arrayed, with `x` taking precedence. That is, the order of the experiments is $x_1y_1, x_2y_1, \dots, x_ny_1, x_1y_2, x_2y_2, \dots, x_my_2, \dots, x_my_n$, with total of $m \times n$ experiments being performed.

`'(x,y)'` indicates the parameters `x` and `y` are jointly arrayed. The number of elements of the parameters `x` and `y` must be identical, and the order of experiments is $x_1y_1, x_2y_2, \dots, x_ny_n$, with `n` experiments being performed.

Joint arrays can have up to 10 parameters. Regular multiple arrays can have up to 20 parameters, with each parameter being either a simple parameter or a diagonal array. The total number of elements in all arrays can be $2^{32}-1$.

See also: *User Guide: Liquids NMR*

Related: `array` Easy entry of linearly spaced array values (M)

arraydim Dimension of experiment (P)

Description: After `calcdim` calculates the dimension of an experiment, the result is put into the parameter `arraydim`. If an experiment is arrayed, `arraydim` is the product of the size of the arrays.

See also: *Getting Started*

Related: `calcdim` Calculate dimension of experiment (C)
`celem` Completed FID elements (P)

asin Find arc sine of number (C)

Syntax: `asin(value)<:n>`

Description: Finds the arc sine (also called the inverse sine) of a number.

Arguments: `value` is a number in the range of ± 1.0 .

`n` is a return argument giving the arc sine, in radians, of `value`. The default is to display the arc sine value in the status window.

Examples: `asin(.5)`
`asin(val):asin_val`

See also: *VNMR User Programming*

Related: `sin` Find sine value of an angle (C)

asize Make plot resolution along f_1 and f_2 the same (M)

Syntax: `asize`

Description: Adjusts the 2D display parameters (`sc`, `wc`, `sc2`, and `wc2`) so that the displayed resolution along both f_1 and f_2 is the same. It is not suggested for heteronuclear experiments where the chemical shift spread of one nucleus is much greater than that of the other.

See also: *User Guide: Liquids NMR*

Related: `sc` Start of chart (P)
`sc2` Start of chart in second direction (P)
`wc` Width of chart (P)
`wc2` Width of chart in second direction (P)

assign Assign transitions to experimental lines (M)

Syntax: (1) `assign('mark')>`
(2) `assign(transition_number,line_number)`

Description: Assigns the nearest calculated transition to the lines from a `d11` or `n11` listing after `spin11` has placed them in `selfreq`. All lines may not be assigned and transitions must be greater than `sth`. The next `spins('iterate')` determines new parameters to minimize the differences in position of the assigned pairs.

Arguments: 'mark' makes assign use the lines selected with the mark button in place of **d11**. The results of the **mark** operation are stored in the file `mark1d.out`, which is cleared by the command `mark('reset')`.

`transition_number` is a single calculated transition number that is assigned to a line from the **d11** listing.

`line_number` is the index of the line from the **d11** listing. Setting `line_number=0` removes an assignment from a calculated transition.

Alternate: auto assign button in the Spin Simulation Line Assignment Menu.

Examples: `assign`
`assign('mark')`
`assign(4,0)`

See also: *User Guide: Liquids NMR*

Related:	d11	Display listed line frequencies and intensities (C)
	mark	Determine intensity of the spectrum at a point (C)
	n11	Find line frequencies and intensities (C)
	slfreq	Measured line frequencies (P)
	spin11	Set up <code>slfreq</code> array (M)
	spins	Perform spin simulation calculation (C)
	sth	Minimum intensity threshold (P)

at Acquisition time (P)

Description: Length of time during which each FID is acquired. Since the sampling rate is determined by the spectral width **sw**, the total number of data points to be acquired ($2 * sw * at$) is automatically determined and displayed as the parameter **np**. `at` can be entered indirectly by using the parameter **np**.

Values: Number, in seconds. A value that gives a number of data points not a multiple of 64 (*MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems and systems with an Output board) or a multiple of 2 (systems with an Acquisition Controller or Pulse Sequence Controller board) are readjusted automatically to be a multiple of 64 or 2 (refer to the description of `acquire` in the manual *VNMR User Programming* to identify these boards).

See also: *Getting Started; VNMR User Programming*

Related:	np	Number of data points (P)
	sw	Spectral width in directly detected dimension (P)

atan Find arc tangent of a number (C)

Syntax: `atan(value)<:n>`

Description: Finds the arc tangent (also called the inverse tangent) of a number.

Arguments: `value` is a number between $\pi/2$ and $-\pi/2$.

`n` is a return argument giving the arc tangent, in radians, of `value`. The default is to display the arc tangent value in the status window.

Examples: `atan(.5)`
`atan(val):atan_val`

See also: *VNMR User Programming*

Related:	sin	Find sine value of an angle (C)
----------	------------	---------------------------------

atan2 Find arc tangent of two numbers (C)

Syntax: `atan2(y,x)<:n>`

Description: Finds the arc tangent (also called the inverse tangent) of the quotient of two numbers.

Arguments: `y` and `x` are two numbers, where the quotient y/x is between $\pi/2$ and $-\pi/2$ and `x` is not equal to zero.

`n` is a return argument giving the arc tangent, in radians, of y/x . The default is to display the arc tangent value in the status window.

Examples: `atan2(1,2)`
`atan2(val):atan2_val`

See also: *VNMR User Programming*

Related: `sin` Find sine value of an angle (C)

atext Append string to current experiment text file (M)

Syntax: `atext(string)`

Description: Adds a line of text to the current experiment text file.

Arguments: `string` is a single line of text.

Examples: `atext('T1 Experiment')`

See also: *Getting Started*

Related: `ctext` Clear the text of the current experiment (C)
`text` Display text or set new text for current experiment (C)
`write` Write formatted text to a device (C)

attens Fast attenuators present (P)

Applicability: *GEMINI 2000* broadband systems. *GEMINI 2000* $^1\text{H}/^{13}\text{C}$ systems have fixed levels, therefore the value should be set to 'n'.

Description: Sets the version of RF Control board. The RF Control board provides computer control of the output power of the transmitter and decoupler channel. The value is set in the CONFIG window (opened from `config`) using the label BB Atten Type. The most recent version of the board, which began shipping in April 1991, is the *diode switching* version. This replaced the *relay switching* version. The diode switching version is much faster—it changes power levels in typically 3 to 4 μs , as compared to 2 to 5 ms for the relay switching board.

To identify which version is on a system, locate the RF Control board (labeled RF CTRL at the left end of the lower card cage) and examine the space below the three small gold coaxial connectors on the board. The relay switching version has a 0.5-inch black potentiometer below the connectors; the diode switching version has either blank space or an unused connector. If the version is still not identified, shut down the system and pull out the RF Control board. If the part number printed on the board is 00-990988-00, the board is the diode switching version. If the part number is 00-966900-00, it is the relay switching version.

Values: 'n' for the relay switching version (Slow choice in the CONFIG window).

'y' for the diode switching version (Fast choice in the CONFIG window).

See also: *VNMR and Solaris Software Installation*

Related: `config` Display current configuration and possibly change it (M)
`pfilter` Programmable filters (P)

au **Submit experiment to acquisition and process data (M)**

Syntax: `au(<'nocheck'><,<'next'><,<'wait'>>>`

Description: Performs the experiment described by the current acquisition parameters, checking the parameters `loc`, `spin`, `gain`, `wshim`, `load`, and `method` to determine the necessity to perform various actions in addition to simple data acquisition. This may involve a single FID or multiple FIDs, as in the case of arrays or 2D experiments. `au` causes the data to automatically be processed according to the following parameters:

- `wbs` specifies what happens after each block.
- `wnt` specifies what happens after each FID is collected.
- `wexp` specifies what happens when the entire acquisition is complete (which may involve several complete FIDs in the case of 1D arrays or 2D experiments).

Before starting the experiment, `au` executes the two user-created macros if they exist. The first is `usergo`, a macro that allows the user to set up general conditions for the experiment. The second is a macro whose name is formed by `go_` followed by the name of the pulse sequence (from `seqfil`) to be used (e.g., `go_s2pul`, `go_dept`). This macro allows a user to set up experiment conditions suited to a particular sequence.

Arguments: `'nocheck'` is a keyword to override checking if there is insufficient free disk space for the complete 1D or 2D FID data set to be acquired.

`'next'` is a keyword to put the experiment started with `au('next')` at the head of the queue of experiments to be submitted to acquisition.

`'wait'` is a keyword to stop submission of experiments to acquisition until `wexp` processing of the experiment, started with `au('wait')`, is finished.

Examples: `au`
`au('wait')`

Alternate: Automatic button in the Acquire Menu.

See also: *Getting Started; User Guide: Liquids NMR*

Related:	<code>auto_au</code>	Controlling macro for automation (M)
	<code>change</code>	Submit a change sample experiment to acquisition (M)
	<code>ga</code>	Submit experiment to acquisition and FT the result (M)
	<code>gain</code>	Receiver gain (P)
	<code>go</code>	Submit experiment to acquisition (M)
	<code>go_</code>	Pulse sequence setup macro called by <code>go</code> , <code>ga</code> , and <code>au</code> (M)
	<code>load</code>	Load status of displayed shims (P)
	<code>loc</code>	Location of sample in tray (P)
	<code>lock</code>	Submit an Autolock experiment to acquisition (C)
	<code>method</code>	Autoshim method (P)
	<code>sample</code>	Submit change sample, Autoshim experiment to acquisition (M)
	<code>seqfil</code>	Pulse sequence name (P)
	<code>shim</code>	Submit an Autoshim experiment to acquisition (C)
	<code>spin</code>	Submit a spin setup experiment to acquisition (C)
	<code>spin</code>	Sample spin rate (P)
	<code>su</code>	Submit a setup experiment to acquisition (M)
	<code>usergo</code>	Experiment setup macro called by <code>go</code> , <code>ga</code> , and <code>au</code> (M)
	<code>wbs</code>	Specify action when <code>bs</code> transients accumulate (C)
	<code>wexp</code>	Specify action when experiment completes (C)
	<code>wnt</code>	Specify action when <code>nt</code> transients accumulate (C)
	<code>wshim</code>	Conditions when shimming is performed (P)

AuC **Get parameters for carbon 1D experiment in *GLIDE* (M)**Applicability: *GLIDE*

Syntax: AuC

Description: Retrieves standard carbon parameter set and *GLIDE*-related parameters.**AuCALch3i** **Set up autocalibration with CH₃I sample (M)**

Syntax: AuCALch3i

Description: Retrieves standard proton parameter set and setup for automatic calibration of proton (observe and decouple), carbon (observe and decouple), *gcal*, and C/H gradient ratio. The AuCALch3i macro is the same as the AuCALch3i1 macro.Related: *AuCALch3i1* Get autocalibration with CH₃I sample (M)*gcal* Gradient calibration constant (P)**AuCALch3i1** **Get autocalibration with CH₃I sample (M)**

Syntax: AuCALch3i1

Description: Retrieves standard proton parameter set and setup for automatic calibration of proton (observe and decouple), carbon (observe and decouple), *gcal*, and C/H gradient ratio. The AuCALch3i1 macro is the same as the AuCALch3i macro. ~~Uses AuCALch3i2 and AuCALch3i3.~~Related: *AuCALch3i* Set up autocalibration macros with CH₃I sample (M)*gcal* Gradient calibration constant (P)~~**AuCALch3i2** **Autocalibration macros with CH₃I sample (M)**~~~~Syntax: AuCALch3i2~~~~Description: Internal macro used by AuCALch3i1.~~Related: *AuCALch3i1* Get autocalibration with CH₃I sample (M)~~**AuCALch3i3** **Autocalibration macros with CH₃I sample (M)**~~~~Syntax: AuCALch3i3~~~~Description: Internal macro used by AuCALch3i1.~~Related: *AuCALch3i1* Get autocalibration with CH₃I sample (M)**AuCALch3oh** **Set up autocalibration with Autotest sample (M)**

Syntax: AuCALch3oh

Description: Retrieves standard proton parameter set and setup for automatic calibration of proton (observe), carbon (decouple), *gcal* and C/H gradient ratio. The AuCALch3oh macro is the same as the AuCALch3oh1 macro. ~~Uses AuCALch3oh2.~~Related: *AuCALch3oh1* Autocalibration macros with Autotest sample (M)*gcal* Gradient calibration constant (P)**AuCALch3oh1** **Get autocalibration with Autotest sample (M)**

Syntax: AuCALch3oh1

Description: Retrieves standard proton parameter set and setup for automatic calibration of proton (observe), carbon (decouple), `gcal` and C/H gradient ratio. The `AuCALch30h1` macro is the same as the `AuCALch30h` macro. ~~Uses `AuCALch30h2`.~~

Related: `AuCALch3oh` Autocalibration macros with Autotest sample (M)
`gcal` Gradient calibration constant (P)

~~`AuCALch3oh2`~~ **Autocalibration macros with Autotest sample (M)**

~~Syntax: `AuCALch3oh2`~~

~~Description: Internal macro used with `AuCALch3oh1`.~~

Related: `AuCALch3oh1` Get autocalibration with Autotest sample (M)

`AuCalibz0` **Automatic Hz to DAC calibration for Z0 (M)**

Applicability: Autocalibration routine

Syntax: Called by `Augmapz0` calibration routine

Description: Called by `Augmapz0` calibration routine. Automatically calibrates lock frequency change per Z0 DAC unit change. the calibrated value is written out in the probe file as `lkhzdac` parameter.

Related: `Augmapz0` Automatic lock gradient map generation and Z0 calibration (M)
`Aufindz0` Automatic adjustment of Z0 (M)

~~`AuCdec_n`~~ **Carbon decoupler calibration macros (M)**

~~Syntax: `AuCdec_n`, where *n* is 1 to 8~~

~~Description: Used by `AuCALch3i1` and `AuCALch3oh1` autocalibration routines to do carbon decoupler calibrations. Calibrates high-power pulse widths and `dmf`.~~

Related: `AuCALch3i1` Get autocalibration with CH₃I sample (M)
`AuCALch3oh1` Get autocalibration with Autotest sample (M)
`dmf` Decoupler modulation frequency for first decoupler (P)

`AuCdec` **Carbon decoupler calibration macro (M)**

Syntax: `AuCdec`

Description: Used by `AuCALch3i` and `AuCALch3oh` autocalibration routines to do carbon decoupler calibrations. Calibrates high-power pulse widths and `dmf`.

Related: `AuCALch3i` Get autocalibration with CH₃I sample (M)
`AuCALch3oh` Get autocalibration with Autotest sample (M)
`dmf` Decoupler modulation frequency for first decoupler (P)

`AuCDEPT` **Get parameters for carbon and DEPT experiments in *GLIDE* (M)**

Applicability: *GLIDE*

Syntax: `AuCDEPT`

Description: Retrieves standard carbon parameter set and *GLIDE*-related parameters, and sets up carbon and DEPT chains.

`AuCexp` **Get parameters for ¹³C & ¹³C-detected experiments in *GLIDE* (M)**

Applicability: *GLIDE*

Syntax: AuCexp

Description: Retrieves standard carbon parameter set and *GLIDE*-related parameters. Supports APT, DEPT, HETCOR (phase-sensitive), PROTON, and COSY.

AuCgrad **Carbon/proton gradient ratio calibration macro (M)**

Syntax: AuCgrad

Description: Used by [AuCALch3i1](#) and [AuCALch3oh1](#) autocalibration routines for C/H gradient ratio calibrations.

Related: [AuCALch3i1](#) Get autocalibration with CH₃I sample (M)
[AuCALch3oh1](#) Get autocalibration with Autotest sample (M)

AuCobs **Carbon observe calibration macro (M)**

Syntax: AuCobs

Description: Used by [AuCALch3i1](#) autocalibration routines for carbon observe calibrations.

Related: [AuCALch3i1](#) Get autocalibration with CH₃I sample (M)

audiofilter **Audio filter board type (P)**

Applicability: All systems except *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Description: Sets the type of audio filter board used where the spectral width ([sw](#)) is less than 100 kHz. The filter type is set in the CONFIG window (opened from [config](#)) using the label Audio Filter Type.

Values: 'b' indicates the system has a 100-kHz Butterworth filter board (100 kHz Butterworth choice in the CONFIG window).

'e' indicates the system has a 100-kHz elliptical filter board (100 kHz Elliptical choice in the CONFIG window).

'2' indicates the system has a 200-kHz Butterworth filter board (200 kHz Butterworth choice in the CONFIG window).

'5' indicates the system has a 500-kHz elliptical filter board (500 kHz Elliptical choice in the CONFIG window).

See also: *VNMR and Solaris Software Installation*

Related: [config](#) Display current configuration and possibly change it (M)
[sw](#) Spectral width in directly detected dimension (P)

AuF **Get parameters for fluorine 1D experiment in GLIDE (M)**

Applicability: *GLIDE*

Syntax: AuF

Description: Retrieves standard fluorine parameter set and *GLIDE*-related parameters.

Aufindz0 **Automatic adjustment of Z0 (M)**

Syntax: Aufindz0

Description: Finds z0 by doing lock 1D spectrum. The frequency is then used along with the [lkhzdac](#) value in the probe file to calculate the z0 value for a given solvent

and autolocking is done. This requires previous calibration of the hzdac value done using the **Aucalibz0** macro.

Related: **Aucalibz0** Automatic Hz to DAC calibration for Z0 (M)
Aumapz0 Automatic lock gradient map generation and Z0 calibration (M)

Augcal Probe gca1 calibration macro (M)

Syntax: Augcal

Description: Used by **AuCALch3i1** and **AuCALch3oh1** autocalibration routines for probe **gca1** calibrations.

Related: **AuCALch3i1** Get autocalibration with CH₃I sample (M)
AuCALch3oh1 Get autocalibration with Autotest sample (M)
gca1 Gradient calibration constant (P)

Augmap Automated gradient map generation (M)

Syntax: Augmap

Description: Automatically adjusts gradient level, offset, window, and pulse width to generate a z1–z4 gradient map using a 2-Hz D₂O sample. This macro is used by the **Aumakegmap** auto gradient map generation macro and is applicable only for a lock gradient map.

Related: **Aumakegmap** Auto lock gradient map generation (M)
gsize Number of z-axis shims used by gradient shimming (P)

Augmapz0 Automatic lock gradient map generation and z0 calibration (M)

Syntax: Augmapz0

Description: Using the 2-Hz D₂O sample, the **augmapz0** macro automatically creates a lock gradient map, followed by Hz to DAC calibration of Z0 for the autolocking procedure.

Related: **Aucalibz0** Automatic Hz to DAC calibration for Z0 (M)
Aufindz0 Automatic adjustment of Z0 (M)

AuH Get parameters for proton 1D experiment in *GLIDE* (M)

Applicability: *GLIDE*

Syntax: AuH

Description: Retrieves standard proton parameter set and *GLIDE*-related parameters.

AuH4nuc Set up parameters for selectable 4nuc (HCPF) experiment (M)

Applicability: *GLIDE*

Syntax: AuH4nuc

Description: Retrieves standard proton parameter set and *GLIDE*-related parameters. Fluorine, carbon, and phosphorus can be added to the chain.

AuHCOSY Set up parameters for ¹H and COSY experiments in *GLIDE* (M)

Applicability: *GLIDE*

Syntax: AuHCOSY

Description: Retrieves standard proton parameter set and *GLIDE*-related parameters, and sets up a proton and COSY chain.

AuHdec **Proton decoupler calibration (M)**

Syntax: AuHdec

Description: Used by [AuCALch3i](#) autocalibration routine to do proton decoupler calibrations. Calibrates high-power pulse widths and [dmf](#).

Related: [AuCALch3i](#) Get autocalibration with CH3I sample (M)
[dmf](#) Decoupler modulation frequency for first decoupler (P)

AuHexp **Get parameters for ¹H & ¹H-detected experiments in *GLIDE* (M)**

Applicability: *GLIDE*

Syntax: AuHexp

Description: Retrieves standard proton parameter set and *GLIDE*-related parameters. Supports COSY, HMQC, HMBC, HSQC, HMQCTOXY, HSQCTOXY, NOESY, ROESY, TOCSY, DEPTHMQC, and CARBON. A gradient coherence selection option is included for COSY, HMQC, HMBC, HSQC, and DEPTHMQC.

AuHobs **Proton observe calibration macro (M)**

Syntax: AuHobs

Description: Used by [AuCALch3i](#) and [AuCALch3oh](#) autocalibration routines for proton observe calibrations.

Related: [AuCALch3i](#) Get autocalibration with CH3I sample (M)
[AuCALch3oh](#) Get autocalibration with Autotest sample (M)

AuHsel **Get parameters for ¹H and ¹H-detected experiments in *GLIDE* (M)**

Applicability: *GLIDE*

Syntax: AuHsel

Description: Retrieves standard proton parameter set and *GLIDE*-related parameters for ¹H and ¹H-detected selective excitation experiments. Supports TOCSY1D and NOESY1D, ROESY1D, and HOMODEC.

Aumakegmap **Auto lock gradient map generation (M)**

Syntax: Aumakegmap (<lk or hs or H1>)

Description: Generates z1–z4 lock gradient ('lk' argument), lock homospoil ('hs' argument), or ¹H gradient map ('H1' argument). If no argument is given, the default is 'lk', if `gradtype='nnh'` to 'hs'. The doped 2-Hz D₂O should be used for hs and lk maps. H1 map is typically done on the sample. Automatically adjusts gradient level, offset, window, and pulse width. The map name is automatically stored in the probe file.

Related: [Aumapz0](#) Automatic lock gradient map generation and Z0 calibration (M)

AuNuc **Get parameters for a given nucleus (M)**

Syntax: AuNuc(nucleus, solvent)

Description: Retrieves standard parameter set for a given nucleus and adds all required parameters for *GLIDE* or Tcl/dg driven parameters. If no parameter set exists in `stdpar`, then carbon parameters are retrieved and `tn` changed.

AuP Get parameters for phosphorus 1D experiment in *GLIDE* (M)

Applicability: *GLIDE*

Syntax: AuP

Description: Retrieves standard phosphorus parameter set and *GLIDE*-related parameters.

auto Prepare for an automation run (C)

Applicability: Systems with an automatic sample changer.

Syntax: `auto<(automation_directory)>`

Description: Prepares the automation directory for an automation run. `auto` aborts if the spectrometer is already in automation mode.

Arguments: `automation_directory` is the name of the automation directory, either an absolute UNIX path (i.e. the first character is a "/") or a relative path (the first character is not a "/"). The default is the value of the parameter `autodir`. If for some reason `autodir` is not defined, you are prompted to provide the location of the automation directory. If not given as an argument, you are prompted for the path. If the automation directory is not present, it is created with full access for all users. `auto` aborts if it fails to create this directory.

Examples: `auto`
`auto('/home/vnmr1/autorun_620')`

See also: *User Guide: Liquids NMR*

Related:	<code>auto_au</code>	Controlling macro for automation (M)
	<code>autodir</code>	Automation directory absolute pathname (P)
	<code>autogo</code>	Start an automation run (C)
	<code>autoname</code>	Prefix for automation data file (P)

auto Automation mode active (P)

Applicability: Systems with an automatic sample changer.

Description: A global variable that shows whether or not an automation run is in progress. Macros typically test this parameter because actions can differ between the automation and non-automation modes. The value of `auto` is not enterable by the user. An automation experiment is initiated with the `autogo` command. The `auto` parameter is only set to 'y' for those macros and commands that are run as part of an automation experiment.

Values: 'y' indicates automation mode is active.
'n' indicates automation mode is inactive.

See also: *User Guide: Liquids NMR*

Related:	<code>auto_au</code>	Controlling macro for automation (M)
	<code>autogo</code>	Start an automation run (C)
	<code>autora</code>	Resume suspended automation run (C)
	<code>autosa</code>	Suspend current automation run (C)

AutoAddEXP Add selected experiment to *GLIDE* chain (M)

Applicability: *GLIDE*

Syntax: `AutoAddEXP (experiment)`

Description: Adds a specified experiment to the *GLIDE* chain.

Arguments: `experiment` is the name of the experiment to be added.

Examples: `AutoAddEXP (' COSY ')`

auto_au Controlling macro for automation (M)

Applicability: Systems with an automatic sample changer.

Syntax: `auto_au`

Description: Reads `sampleinfo` file (defines an automation experiment) using the `lookup` facility, sets the `solvent` and `loc` parameters based on the `SOLVENT` and `SAMPLE#` fields of `sampleinfo`, runs `exec` on the entry in the `MACRO` field, and writes the experiment text based on the `TEXT` field. After that, `auto_au` examines the value of the `wexp` parameter:

- If `wexp` is set to `'procplot'`, then `auto_au` calls `au`.
- If `wexp` is set to `'autolist'`, then `auto_au` inserts `'auto'` as the first argument to `autolist` and calls `au('wait')`.
- If `wexp` is set to anything else, `auto_au` does not call `au`.

If no data is generated from the requested `MACRO` field, due to an error or some other reason, `auto_au` sets the `STATUS` field to “No Data Requested.”

`auto_au` is used only during automation and should not be called directly. It provides a starting point for all automation experiments. As such, it is a convenient point for user customization of automation.

See also: *User Guide: Liquids NMR*

Related:	<code>au</code>	Submit experiment to acquisition and process data (M)
	<code>auto</code>	Prepare for an automation run (C)
	<code>autolist</code>	Set up and start chained acquisition (M)
	<code>exec</code>	Execute a VNMR command (C)
	<code>loc</code>	Location of sample in tray (P)
	<code>lookup</code>	Look up words and lines from a text file (C)
	<code>solvent</code>	Lock solvent (P)
	<code>wexp</code>	When experiment completes (P)

Autobackup Back up current probe file (M)

Syntax: `Autobackup`

Description: Makes a copy of the probe file before starting the calibrations and prints the current calibration file. `Autobackup` is called by the autocalibration routines `AuCALch3i1` and `AuCALch3oh1`.

Related:	<code>AuCALch3i1</code>	Get autocalibration with CH ₃ I sample (M)
	<code>AuCALch3oh1</code>	Get autocalibration with Autotest sample (M)

Autocalnext Run next item in calibration chain (M)

Applicability: *GLIDE*

Syntax: `Autocalnext`

Description: Starts the current routine in the list of experiments.

- Autocalpar Add *GLIDE* calibration parameters (M)**
 Applicability: *GLIDE*
 Syntax: Autocalpar
 Description: Adds *GLIDE* calibration-related parameters to the current parameter sets. Used by *GLIDE* setup macros.
- Autocalsave Save current item in the calibration directory (M)**
 Applicability: *GLIDE*
 Syntax: Autocalsave (file)
 Description: Saves the current FID in the directory userdir+/data/+sample.
 Arguments: file is the name of the file to be saved.
- AutoCheck Check for FID file (M)**
 Applicability: *GLIDE*
 Syntax: AutoCheck (file) : \$present, \$file_name
 Description: Checks if a FID file exists in the userdir+/data/+sample directory.
 Arguments: file is the file name of the FID file to be checked.
 present is a return value of 1 if the FID file exists or 0 if it does not exist.
 file_name is a return string with the full file name if the FID file exists.
 Examples: AutoCheck ('PROTON') : \$FID_yes, \$file
 Related: **AutoStrtfid** Recall stored FID (M)
 AutoStrtpar Recall stored parameters (M)
- Autoclrexp Clean up current experiment (M)**
 Applicability: *GLIDE*
 Syntax: Autoclrexp
 Description: Removes .def, macdir, and eou* files.
- AutoDelCAL Delete selected calibration routine from *GLIDE* chain (M)**
 Applicability: *GLIDE*
 Syntax: AutoDelCAL (experiment)
 Description: Removes the specified calibration routine from the *GLIDE* chain.
 Arguments: experiment is the name of the calibration routine to delete.
 Examples: AutoDelCAL ('Cdec')
- AutoDeleXP Delete selected experiment from *GLIDE* chain (M)**
 Applicability: *GLIDE*
 Syntax: AutoDeleXP ('experiment')
 Description: Removes the specified experiment from the *GLIDE* chain.
 Arguments: experiment is the name of experiment to be deleted.
 Examples: AutoDeleXP ('COSY')

autodept Automated complete analysis of DEPT data (M)

Syntax: `autodept`

Description: Processes DEPT spectra, plots the unedited spectra, edits the spectra, plots the edited spectra, and prints out editing information.

Alternate: Full Analysis button in the Automatic DEPT Analysis Menu.

See also: *User Guide: Liquids NMR*

Related: `adept` Automatic DEPT analysis and spectrum editing (C)
`deptproc` Process DEPT data (M)
`padept` Perform `adept` analysis and plot resulting spectra (C)
`pldept` Plot DEPT data, edited or unedited (M)

autodir Automation directory absolute path (P)

Applicability: Systems with an automatic sample changer or LC-NMR accessory.

Description: When using a sample changer, `autodir` is a global variable that holds the absolute path of the currently active automation directory. When VNMR is started, `autodir` is set to the absolute path of the last automation run.

When using the LC-NMR accessory, `autodir` specifies a directory in which experiments using a stored queue are saved.

See also: *User Guide: Liquids NMR*

Related: `auto` Set up an automation directory (C)
`autoname` Prefix for automation data file (P)

Autoexplist Display current *GLIDE* selection (M)

Applicability: *GLIDE*

Syntax: `Autoexplist`

Description: Shows current *GLIDE* selection from the list of experiments in the text window.

autogo Start automation run (C)

Applicability: Systems with an automatic sample changer.

Syntax: `autogo<(file<,automation_directory>)>`

Description: Starts an automation run. The `autogo` parameter cannot be entered while the spectrometer is in automation mode. You must have an `enter` queue prepared to start an automation run. The queue is checked to verify that it was prepared using the `enter` command (`autogo` aborts if an error in the format is found.) Your automation directory is also checked for the presence of a non-empty `enter` queue (`autogo` aborts if the current queue in the automation directory is present and not empty). Finally, `autogo` checks the automation directory and runs the `auto` command if this directory is not present or another problem is found. When `autogo` completes, the system is in automation mode and your automation run starts.

Arguments: `file` is the file name of your `enter` queue. The default is that the system prompts you for the location of the `enter` queue.

`automation_directory` is the pathname of the automation directory. The default is the current value of the parameter `autodir`.

Examples: `autogo`
`autogo('MySamples')`
`autogo('MySamples','/home/vnmr1/AutoRun_621')`

See also: *User Guide: Liquids NMR*

Related: **auto** Set up an automation directory (C)
autodir Automation directory absolute path (P)
autoname Prefix for automation data file (P)
enter Enter sample information for automation run (C)

AutoLIST Run chained experiments (M)

Applicability: *GLIDE*

Syntax: `AutoLIST(experiment1, experiment2, ...)`

Description: Saves the current FID (first argument), executes the second argument, and executes `au('next', 'wait')` to start the next acquisition.

Arguments: `experiment1, experiment2, ...` are names of the experiments to be run.

Examples: `AutoLIST('PROTON')`

autolist Set up and start chained acquisition (M)

Syntax: `autolist(<options>, >experiment1<, >experiment2<, ...)`

Description: Sets up parameters for chained experiments by executing the experiments given as arguments and then starting a chained acquisition. Note that the macro **au** is executed as part of **autolist** and should not be included in the arguments to **autolist**.

Arguments: `options` is one or more of the following keywords:

- 'auto' is a keyword to add 'wait' to the **au** call (e.g. `au('wait', 'next')`).
- 'glide' is a keyword to process the current data with the `glidewexp` macro instead of the `procplot` macro. Typically, the macros that chain experiments, such as `hcosy`, `hcapt`, `hc`, and `hccorr`, start the experiment and then set `wexp` to 'autolist' with the `autolist` arguments being the list of experiments.
- 'start' is a keyword to make the first experiment in the list as one that needs to be acquired rather than processed.

`experiment1, experiment2, ...` are experiments written as strings (e.g., 'dept' or 'c13'). `experiment1` is the current experiment and, when it finishes, the macro `procplot` is called to process the data. If `experiment2` is listed, that experiment is executed and then the macro `au('next')` is performed. For subsequent experiments, the text, `solvent` and `temp` are used from the preceding experiment. Also, the `wexp` parameter is reset to 'autolist' with the first experiment removed.

Examples: `autolist('h1', 'c13', 'dept')`
`autolist('glide', 'h1', 'hcosy')`

See also: *Getting Started; User Guide: Liquids NMR*

Related: **auto_au** Controlling macro for automation (M)
au Submit experiment to acquisition and process data (M)
hc Automated proton and carbon acquisition (M)
hcapt Automated proton, carbon, and APT acquisition (M)
hccorr Automated proton, carbon, and HETCOR acquisition (M)
hcosy Automated proton and COSY acquisition (M)
procplot Automatically process FIDs (M)
solvent Lock solvent (P)

`temp` Sample temperature (P)
`wexp` When experiment completes (P)

Automacrodir Create directory to save macros in *GLIDE* run (M)

Applicability: *GLIDE*

Syntax: Automacrodir

Description: Makes a directory in `userdir+/maclib.glide` to save *GLIDE*-created macros. The directory name is the same as the experiment number (`exp1`, `exp2`, etc). Macros are stored and executed at the appropriate juncture in the *GLIDE* run.

Automkdir Create directory to save data in *GLIDE* run (M)

Applicability: *GLIDE*

Syntax: Automkdir

Description: Makes a directory in `userdir+/data` to save *GLIDE* run data. The directory name is supplied by the file `expsolv.def`. FIDs are stored in the file `seqfil.fid`.

autoname Create path for data storage (C)

Syntax: `autoname(<text_file>,<parameter_name>):$path`

Description: Determines a path where data can be stored. This command provides the functionality of the `autoname` parameter without being in automation mode.

Arguments: `text_file` is the name of a text file from which information can be extracted to construct the path name. Any file can be used to get information. The file `sampleinfo` in the current experiment directory is used as the default if a `text_file` is not specified.

`parameter_name` is the name of an alternate parameter to be used as the `autoname` parameter. The default is to use `autoname`. The specifications of a `parameter_name` are similar to those used by the `autoname` parameter during an automation run. If an alternate parameter is used, it will probably need to be created in the global tree as a string.

`$path` is a return argument with the path. If no return argument is present, the result is displayed on line 3.

Examples: `autoname:$autoname_path`
`autoname(curexp+' /text '):$p1`

See also: *User Guide: Liquids NMR*

Related: `auto` Set up an automation directory (C)
`autogo` Start automation run (C)
`autodir` Automation directory absolute path (P)
`autoname` Prefix for automation data file (P)
`enter` Enter sample information for automation run (M)
`status` Display status of sample changer (C,U)

autoname Prefix for automation data file (P)

Applicability: Systems with an automatic sample changer.

Description: Stores a string in the global tree that determines a prefix to the file name of the FID data (e.g., `0204.fid`) during an automation run. Percent signs (%) are

used to delimit a string to search for in the `sampleinfo` file, and the word after the delimited string is used in the file name. This word can be terminated with a space, tab, or carriage return. Text not delimited by percent signs is copied from `autoname` without any changes.

If `autoname` does not start with a slash mark (/), the file is stored in the path given by `autodir`; otherwise, the name is used as is. The sample number is not automatically appended, but a revision number is appended.

Values: If `autoname` is a null string, the file name `%SAMPLE#:%PEAK#:%` is the default, resulting in the name `sample_number+revision_number.fid` (LC-NMR uses `PEAK#:` in the `sampleinfo` file, resulting in the name `peak_number+revision_number.fid`). Note that the `autoname` of the user doing the automation run is used for all file names and that the resulting path and file name must be accessible (with read-write permission) by that user. `autoname` controls the version number attached to the name of a file and uses the value of VNMR parameters as part of the file's name. For example, `autoname=' $seqfil$ _tn` names a file with the current value of the parameters `seqfil` and `tn`. The resulting file name might be `s2pul_H1` or `dept_C13`. If a numeric value is used, this value is truncated to an integer. For example, if `autoname=' $sfrq$ '`, the file name would be `500`, not `500.456`. `%Rn%`, where `n` is 0 to 9 (default is 2) is a special substitute string. `n` determines how the revision number is appended to the FID file name:

- If `n` is 0, no revision digits are appended (all names must be uniquely constructed without these revision digits).
- If `n` is 1 to 9, the revision number is padded with leading zeroes to form an `n`-digit number. If more places are needed than specified, more zeroes are used.

If `n` is greater than 9 (more than one digit), `Rnn` is still used as a search string in the `sampleinfo` file. `Rn` must be specified at the end of the `autoname` string; the revision digits are always appended.

You can also specify the starting number to be used when constructing the version number by appending a colon (:) and start number after `Rn`. The default starting value is 1. A zero is not allowed.

Examples: Using the `enter` program, a sample is entered with the following information (which is copied to the `sampleinfo` file):

```
SAMPLE#: 3
MACRO: h1
USER: John Doe
SOLVENT: CDCl3
TEXT: EthylBenzene in CDCl3
      Page 01-3015
      This is a text
USERDIR: ...
```

This entry creates the following file names for each `autoname` string:

<i>autoname string</i>	<i>File name created</i>
' '	0301.fid
'%USER:%'	John01.fid
'%Page%'	01-301501.fid
'%USER:%/%Page%'	John/01-301501.fid
'/export/home/%TEXT:%'	/export/home/EthylBenzene01.fid
'%USER:%R0%'	John.fid

<i>autoname string</i>	<i>File name created</i>
'%USER:%-R5%'	John-00001.fid
'%USER:%-R1%'	John-10.fid (if tenth revision)

See also: *User Guide: Liquids NMR*

Related:	auto	Set up an automation directory (C)
	autogo	Start automation run (C)
	autodir	Automation directory absolute path (P)
	autoname	Create path for data storage (C)
	enter	Enter sample information for automation run (C)
	status	Display status of sample changer (C,U)

Autoplot2D **Check for *GLIDE*-selected plot options (M)**

Applicability: *GLIDE*

Syntax: `Autoplot2D`

Description: Checks if *GLIDE*-selected plot options are present before executing `plot2D`.

Related: **plot2D** Plot 2D spectra (M)

Autopsgset **Set up parameters for various experiments (M)**

Applicability: *GLIDE*

Syntax: `Autopsgset (file,par1,par2,...,par11)`

Description: Sets up parameters for various experiments using information in a `parlib` file. Same as `psgset` except `Autopsgset` does not do `prune`.

Arguments: `file` is the name of a file from the user or system `parlib` that provides information on setting up the parameters listed. The parameters `seqfil` and `pslabel` are set to the supplied file name.

`par1,par2,...par11` are 1 to 11 parameters to be returned from `parlib`.

Related:	prune	Prune extra parameters from current tree (C)
	psgset	Set up parameters for various experiments (M)
	pslabel	Pulse sequence label (P)
	seqfil	Pulse sequence name (P)

autora **Resume suspended automation run (C)**

Applicability: Systems with an automatic sample changer.

Syntax: `autora`

Description: Resumes a previously suspended automation run. No matter what caused the interruption (including `autosave`, power failure, or system bootup), the system examines the condition of the automation file and resumes acquisition for all experiments that have not finished. If `autora` is executed while an automation run is in progress, it has no effect.

See also: *User Guide: Liquids NMR*

Related: **autosave** Suspend current automation run (C)

Autormmac **Delete macro set from `curexp+ /macdir` (M)**

Applicability: *GLIDE*

Syntax: `Autormmac`

Description: Removes `macdir` entries for a specified experiment.

autosuspend **Suspend current automation run (C)**

Applicability: Systems with an automatic sample changer.

Syntax: `autosuspend`

Description: Suspends the automation mode at the conclusion of the current experiment and changes the system to the manual mode. The currently running experiment is not interrupted.

See also: *User Guide: Liquids NMR*

Related: `autostart` Resume suspended automation run (C)

autoscale **Resume autoscaling after limits set by scalelimits macro (M)**

Syntax: `autoscale`

Description: Returns to autoscaling in which the scale limits are determined by the `expl` command such that all the data in the `expl` input file is displayed.

See also: *User Guide: Liquids NMR*

Related: `expl` Display exponential or polynomial curves (C)
`scalelimits` Set limits for scales in regression (M)

Autosetgpar **Add GLIDE parameters to current parameter sets (M)**

Applicability: *GLIDE*

Syntax: `Autosetgpar`

Description: Used by *GLIDE* setup macros to add *GLIDE*-related parameters to the current parameter sets.

Autosetwexp **Create AutoLIST from experiments list (M)**

Applicability: *GLIDE*

Description: Makes `AutoLIST` using the experiments list and sets up the `wexp` parameter.

Related: `AutoLIST` Run chained experiments (M)
`wexp` Specify action when experiment completes (P)

autostack **Automatic stacking for processing and plotting arrays (M)**

Syntax: `autostack`

Description: When processing and plotting arrayed 1D spectra, VNMR automatically determines whether the stacking mode is horizontal, vertical or diagonal from the number of traces and the number of lines in the spectrum. If this automatic function is not desirable (or makes an undesirable decision), it can be overridden by placing the `stack` macro in the experiment startup macro or by calling `stack` before processing (or reprocessing) a spectrum. `autostack` switches back to automatic determination of the stack mode by destroying the `stackmode` parameter.

See also: *Getting Started*

Related: `procarray` Process arrayed 1D spectra (M)
`plarray` Plot arrayed 1D spectra (M)
`stack` Fix stacking mode for processing / plotting arrayed spectra (M)
`stackmode` Stacking control for processing (P)

AutoStrtfid Recall stored FID (M)

Applicability: *GLIDE*

Syntax: `AutoStrtfid(file)`

Description: Searches for presence of a FID file in the `userdir+' /data/' +sample` directory. If the FID exists, is retrieved and processed. `AutoStrtfid` is used by `NOESY1D` and `TOCSY1D` macros in *GLIDE*.

Arguments: `file` is the name of the FID file to be recalled.

Examples: `AutoStrtfid('PROTON')`

AutoStrtpar Recall stored parameters (M)

Applicability: *GLIDE*

Syntax: `AutoStrtpar(file)`

Description: Searches for presence of a FID file in the `userdir+' /data/' +sample` directory. If the FID exists, the parameters are retrieved. `AutoStrtpar` is used by most macros in *GLIDE*.

Arguments: `file` is the name of the FID file to be recalled.

Examples: `AutoStrtpar('PROTON')`

autotest Open Auto Test Window (C)

Syntax: `autotest`

Description: Opens the Auto Test window.

See also:

autotime Displays approximate time for automation (M)

Syntax: `autotime(<automation directory>)`

Description: Displays approximate time for each experiment and for each location in an automation run. If no argument is given, time is calculated for the current automation run (`enterQ`).

Related: `explist` Display approximate time for current experiment chain (M)

Autowrmac Write a string to a macro in curexp+' /macdir' (M)

Applicability: *GLIDE*

Syntax: `Autowrmac(expt<,'acq'|'prc'|'plt'>,string)`

Description: Writes out the specified string to `exptacq`, `exptprc`, or `exptplt` files in `curexp+' /macdir'`. These files are executed at an appropriate juncture in a *GLIDE* chain.

Arguments: `expt` is the name of the experiment.

`'acq'|'prc'|'plt'` is a keyword to set the file to write to `exptacq`, `exptprc`, or `exptplt`, respectively. The default is `'acq'`.

`string` is the text to write to the file.

Examples: `Autowrmac('COSY','prc','ni=256')`

av Set abs. value mode in directly detected dimension (C)

Syntax: `av`

Description: Selects the absolute-value spectra display mode by setting the parameter **dmg** to the string value 'av'. In the *absolute-value display mode*, each real point in the displayed spectrum is calculated as the square root of the sum of the squares of the real and imaginary points comprising each respective complex data point. All information, including noise, is always positive, and the relationship between signal and noise is linear.

For multidimensional data, av has no effect on data prior to the second Fourier transform. If **pmode**= 'full', av acts in concert with commands **ph1**, **av1**, or **pwr1** to yield the resultant contour display for the 2D data.

See also: *Getting Started*

Related:	av1	Set abs. value mode in 1st indirectly detected dimension (C)
	av2	Set abs. value mode in 2nd indirectly detected dimension (C)
	dmg	Display mode in directly detected dimension (C)
	dmgf	Absolute-value display of FID data or spectrum in acqi (P)
	ft	Fourier transform 1D data (C)
	ft1d	Fourier transform along f_2 dimension (C)
	ft2d	Fourier transform 2D data (C)
	pa	Set phase angle mode in directly detected dimension (C)
	pa1	Set phase angle mode in 1st indirectly detected dimension (C)
	ph	Set phased mode in directly detected dimension (C)
	ph1	Set phased mode in 1st indirectly detected dimension (C)
	pmode	Processing mode for 2D data (P)
	pwr1	Set power mode in 1st indirectly detected dimension (C)
	wft	Weigh and Fourier transform 1D data (C)
	wft1d	Weigh and Fourier transform of 2D data (C)
	wft2d	Weigh and Fourier transform 2D data (C)

av1 **Set abs. value mode in 1st indirectly detected dimension (C)**

Syntax: **av1**

Description: Selects the absolute-value spectra display mode along the first indirectly detected dimension by setting the parameter **dmg1** to the value 'av1'. If the parameter **dmg1** does not exist, av1 creates it and set it to 'av1'.

In the *absolute-value display mode*, each real point in the displayed trace is calculated as the square root of the sum of the squares of the real and imaginary points comprising each respective complex data point. For hypercomplex data, the real-real and imaginary-real points from each respective hypercomplex data point are used in the summation. In this mode, all information, including noise, is always positive; and the relationship between signal and noise is linear.

The av1 command is only needed if mixed-mode display is desired. If the parameter **dmg1** does not exist or is set to the null string, the display mode along the first indirectly detected dimension defaults to the display mode of the directly detected dimension (characterized by the parameter **dmg**). For the contour display of multidimensional data, the result of av1 is the same as for traces provided that **pmode**= 'partial' or **pmode**= '' (two single quotes with no space between).

See also: *User Guide: Liquids NMR*

Related:	av	Set abs. value mode in directly detected dimension (C)
	dmg1	Data display mode in 1st indirectly detected dimension (P)

av2 **Set abs. value mode in 2nd indirectly detected dimension (C)**

Syntax: **av2**

Description: Selects absolute-value spectra display mode for the second indirectly detected dimension by setting the parameter `dmg2` to the value 'av2'. If `dmg2` does not exist or is set to the null string, `av2` creates `dmg2` and set it equal to 'av2'.

In the *absolute-value display mode*, all information, including noise, is positive; and the relationship between signal and noise is linear. Each real point in the displayed trace is calculated as the square root of the sum of the squares of the real and imaginary points comprising each respective complex data point. For hypercomplex data, the real-real and imaginary-real points from each respective hypercomplex data point are used in the summation.

The `av2` command is only needed if mixed-mode display is desired. If the parameter `dmg2` does not exist or is set to the null string, the display mode along the second indirectly detected dimension defaults to the display mode of the directly detected dimension (characterized by the parameter `dmg`). For the contour display of multidimensional data, the result of `av2` is the same as for traces provided that `pmode='partial'` or `pmode=''` (two single quotes with no space between).

See also: *Getting Started*

Related: `av` Set abs. value mode in directly detected dimension (C)
`dmg2` Data display mode in 2nd indirectly detected dimension (P)

averag Calculate average and standard deviation of input (C)

Syntax: `averag(number1,number2,...):average,sd,
number_arguments,sum_numbers,sum_squares`

Description: Finds average, standard deviation, and other characteristics of a set of numbers.

Arguments: `number1,number2,...` is a finite set of numbers.

`average` is the average of the numbers.

`sd` is the standard deviation of the numbers.

`number_arguments` is the number of `number1,number2,...` arguments.

`sum_numbers` is the sum of the numbers

`sum_squares` is the sum of squares of the numbers.

Examples: `averag(3.4,4.3,3.5,5.4):r1,r2`

See also: *VNMR User Programming*

awc Additive weighting const. in directly detected dimension (P)

Description: Adds the current value of `awc` to each value of the weighting function along the directly detected dimension. This dimension is often referred to as the f_2 dimension in 2D data sets, the f_3 dimension in 3D data sets, and so forth. `awc` is applied *after* the sinebell and exponential function, but *before* the Gaussian function. This allows using `gf` as a Gaussian apodization even when `awc` is non-zero. Typical value of `awc` is 'n'.

See also: *Getting Started*

Related: `awc1` Additive weighting const. in 1st indirectly detected dimension (P)
`awc2` Additive weighting const. in 2nd indirectly detected dim. (P)
`gf` Gaussian function in directly detected dimension (P)

- awc1** **Additive weighting const. in 1st indirectly detected dimension (P)**
- Description: Adds the current value of awc1 to each value of the weighting function along the first indirectly detected dimension This dimension is often referred to as the f_1 dimension of a multidimensional data set. awc1 is analogous to the parameter awc. The “conventional” parameters (lb, gf, etc.) operate on the detected FIDs, while this “2D” parameter is used during processing of the interferograms.
- See also: *User Guide: Liquids NMR*
- Related: awc Additive weighting const. in directly detected dimension (P)
-
- awc2** **Additive weighting const. in 2nd indirectly detected dimension (P)**
- Description: Adds the current value of awc2 to each value of the weighting function along the second indirectly detected dimension This dimension is often referred to as the f_2 dimension of a multidimensional data set. awc2 is analogous to the parameter awc. The value of awc2 can be set with wti on the 2D interferogram data.
- See also: *User Guide: Liquids NMR*
- Related: awc Additive weighting const. in directly detected dimension (P)
 wti Interactive weighting (C)
-
- axis** **Provide axis labels and scaling factors (C)**
- Syntax: axis('fn' | 'fn1' | 'fn2')
 <:\$axis_label,\$freq_scaling,\$scaling_factor>
- Description: Displays or returns values of the axis labels and scaling factors to the calling macro. See the macro r1 for an example of using this command.
- Arguments: 'fn' | 'fn1' | 'fn2' is the Fourier number parameter for the axis of interest. \$axis_label is the axis label (e.g., ppm, kHz, cm, or ppm(sc)). \$freq_scaling is the divisor needed to convert from units of Hz to the units defined by the axis parameter with any scaling. axis uses the current value of the axis parameter for that dimension and also checks for axis scaling using the corresponding scalesw, scalesw1, or scalesw2 parameter. \$scaling_factor is a second scaling factor, determined solely by the scalesw type of parameter. This last scaling factor is independent of the value of the axis parameter.
- Examples: axis('fn')
 axis('fn1'):\$lab,\$fr,\$scl
- See also: *VNMR User Programming*
- Related: axis Axis label for displays and plots (P)
 r1 Set reference line (M)
 scalesw Scale spectral width in directly detected dimension (P)
 scalesw1 Scale spectral width in 1st indirectly detected dimension (P)
 scalesw2 Scale spectral width in 2nd indirectly detected dimension (P)
-
- axis** **Axis label for displays and plots (P)**
- Applicability: Certain arguments work only if system has the proper hardware.
- Description: Specifies the units for the axis display and plot.

For 1D experiments, `axis` uses a single letter that includes 'h' for Hz, 'p' for ppm, and 'k' for kHz (e.g., `axis='h'`).

For 2D experiments, `axis` uses two letters, with the first letter describing the detected spectral axis (f_2), and the second letter describing the indirectly detected axis (f_1). Thus `axis='ph'` is appropriate for a homonuclear 2D-J experiment, with a referenced ppm scale along the spectral axis and an axis in Hz ('h') along the J-axis. `axis='pp'` is appropriate for COSY or NOESY experiments.

For 3D experiments, `axis` uses three letters with the first letter describing the detected spectral axis (f_3), the second letter describing the first indirectly detected axis (f_1), and the third letter specifying the second indirectly detected axis (f_2).

The special letter `d` is used to reference the indirectly detected axis to the parts per million of the decoupler channel, as appropriate for heteronuclear chemical shift correlation experiments, which would typically have `axis='pd'`. The letter `n` is used to suppress the axis display on one or both axes (e.g., `axis='nn'`, `axis='pn'`).

For systems with multiple decouplers, the characters '1', '2', and '3' can be used to reference an axis relative to the frequency of that decoupler. Setting `axis='p1'` is effectively the same as `axis='pd'`.

For image display, `axis` can have values 'c' (for centimeters), 'm' (for millimeters), and 'u' (for microns). These values rely on the parameters `lro` and `lpe` for scaling. If both f_1 and f_2 dimensions are spatial, the display aspect ratio is adjusted to retain the aspect ratio of the imaging.

Values:

- '1' sets the axis label for units of ppm relative to the first decoupler.
- '2' sets the axis label for units of ppm relative to the second decoupler.
- '3' sets the axis label for units of ppm relative to the third decoupler.
- 'c' sets the axis label for units of centimeters.
- 'd' sets the axis label for units of ppm relative to the first decoupler.
- 'h' sets the axis label for units of hertz.
- 'k' sets the axis label for units of kilohertz.
- 'm' sets the axis label for units of millimeters.
- 'n' sets no axis label display.
- 'p' sets the axis label for units of ppm relative to the observe transmitter.
- 'u' sets the axis label for units of micrometers.

See also: *Getting Started; User Guide: Liquids NMR*

Related:	<code>axis</code>	Provide axis labels and scaling factors (C)
	<code>axisf</code>	Axis label for FID displays and plots (P)
	<code>dscale</code>	Display scale below spectrum or FID (C)
	<code>lpe</code>	Field of view parameter for phase encode, in cm (P)
	<code>lro</code>	Field of view parameter for readout, in cm (P)
	<code>pscale</code>	Plot scale below spectrum or FID (C)

axisf Axis label for FID displays and plots (P)

Description: Specifies the units for the FID axis display and plot. To create the FID display parameters `axisf`, `dotflag`, `vpf`, `vpfi`, `crf`, and `deltaf` (if the parameter set is older and lacks these parameters), enter `addpar('fid')`.

Values:

- 's' sets the axis label for units of seconds.
- 'm' sets the axis label for units of ms.
- 'u' sets the axis label for units of μ s.

A

'n' sets no axis label display.

See also: *Getting Started*

Related: `addpar` Add selected parameters to the current experiment (M)
`axis` Axis label for displays and plots (P)
`dscale` Display scale below spectrum or FID (C)
`pscale` Plot scale below spectrum or FID (C)

B

B0 Magnet main static field (P)

Applicability: Systems with imaging capabilities.

Description: The field strength, in gauss, of the main magnetic field. This value is used by planning macros in their calculations.

Values: Number, in units of gauss. Nominal value is $234.9 * h1freq$. For example, a 4.7T (200 MHz) system has a value of approximately 47,000.

See also: *User Guide: Imaging*

Related: `h1freq` Proton frequency of spectrometer (P)

bandinfo Shaped pulse information for calibration (M)

Applicability: Information only useful on systems capable of shaped pulse generation.

Syntax: `bandinfo<(shape,width<,ref_power>):duration,power`

Description: Displays a table containing the duration and the predicted 90° pulse power setting for the pulse shape and bandwidth given by the arguments. No parameter settings are changed. The necessary data is contained in the `shapeinfo` file in the VNMR `shapelib` subdirectory.

Arguments: If `bandinfo` is run without arguments, VNMR prompts operator for input
`shape` is the name of the shape. The default is system prompts for a name.
`width` is the bandwidth, in Hz, desired for the pulse.
`ref_power` is value of `tpwr` to which `pw90` is set. The default is 55 dB.
`duration` is the duration, in μ s, of the pulse.
`power` is the predicted 90° pulse power setting.

Examples: `bandinfo`
`bandinfo('sinc',10):pw,tpwr`

See also: *VNMR User Programming*

Related: `pulseinfo` Shaped pulse information for calibration (M)
`pw90` 90° pulse width (P)
`tpwr` Observe transmitter power level with linear amplifiers (P)

banner Display message with large characters (C)

Syntax: `banner(message<,color>)`

Description: Displays text as large-size characters on the VNMR graphics windows.

Arguments: `message` is the text to be displayed. If the text includes a single quotation mark (`'`), it must be preceded by a backslash (`\`). Multiline displays are available by inserting two backslashes (`\\`) between lines. Any undefined characters are displayed as a “bug” shape.

`color` is the color of text on a color display: `'red'`, `'yellow'`, `'green'`, `'cyan'`, `'blue'`, `'magenta'`, and `'white'`. The default is `'yellow'`.

Examples: `banner('banner sample')`
`banner('Don\'t Touch','blue')`

See also: *VNMR User Programming*

bc **1D and 2D baseline correction (C)**
 Description: Makes 1D or 2D baseline correction using a spline or a second to twentieth order polynomial fitting of predefined baseline regions. `bc` defines every other integral (those integrals that disappear when `intmod='partial'`) as baseline and attempts to correct these points to zero.

1D baseline correction

Syntax: `bc<(n|'unbc'<,nsubregion<,minpoints<,minregion>>>>>`

Description: Performs a 1D baseline correction. The nonintegrated parts of the spectrum (i.e., every odd region between integral reset points, or the integral gaps with `intmod='partial'`) are divided into baseline subregions. The number of baseline subregions in each area are adjusted as possible, so that the subregions are more or less equal in size. Finally, the “center of gravity” (midpoint in *x* and average of the *y* values in the region) for each of the subregions is calculated.

Arguments: *n* is an integer from 1 to 20 for the baseline correction step. A polynomial of the (*n*-1)th order is calculated “through” the “baseline points” using the Chebychev least-squares fitting algorithm, and that polynomial function is subtracted from the spectrum. The coefficients of the polynomial are written into the file `cureexp+'bc.out'`. The default is 1 (a spline fit).

'unbc' is a keyword to make `bc` read in the coefficients from the file written by the previous `bc` operation and reverse that operation. This option is only functional for polynomials with two or more coefficients performing baseline correction operations on 1D spectra or individual 2D traces (i.e., baseline corrections cannot be undone with the default spline correction).

`nsubregion` defines the number of subregions (minimum 3, maximum 400). By default, the total number of subregions is 20 (if `fn<2048`), 40 (if `fn=2048` or `fn=4096`), or 80 (if `fn>4096`).

`minpoints` sets the minimum number of data points required in an integral gap for `bc` to regard it as baseline. Use this to exclude small, nonintegrated areas between close signals. The default is `fn/1000` (but at least 3).

`minregion` defines the minimum number of subregions assigned to each baseline area. The default is 1.

Examples: `bc`
`bc(3)`
`bc('unbc')`
`bc(1,200,8,2)` gives a spline correction using 200 baseline subregions, a gap of 8 data points between two (even) integral regions is regarded as baseline, and each baseline area is split into at least two subregions.

See also: *Getting Started; User Guide: Liquids NMR*

2D baseline correction

Syntax: `bc(trace_direction<,num_coeff><,trace_start<,>`
`<,trace_end>)`

Description: 2D baseline correction can be performed on three types of 2D data:

- `f2` spectra (`trace_direction='f2'`) after the first half of a 2D FT (`wft1da`).
- `f2` traces (`trace_direction='f2'`) after a full 2D FT (`wft2da`).
- `f1` traces (`trace_direction='f1'`) after a full 2D FT (`wft2da`).

Arguments: `trace_direction` specifies the direction, 'f1' or 'f2', along which the 2D baseline correction is to take place.

`num_coeff` is the number of coefficients, from 1 to 20, used in the fitting procedure. The default value is 1, which gives a spline fit. A value of 2 gives a linear baseline fit ($a + bx$), a value of 3 gives a quadratic fit ($a + bx + cx^2$), etc. The maximum value (20) gives a 19th-order polynomial fit with 20 coefficients.

`trace_start` is the trace number for the spectrum on which the 2D baseline correction is to start. It must lie within the appropriate range or an error results.

`trace_end` is the trace number for the spectrum on which the 2D baseline correction is to end. It must lie within the appropriate range or an error results.

Examples: `bc('f1')`
`bc('f2',3)`
`bc('f2',3,10,60)`

See also: *User Guide: Liquids NMR*

Related:	<code>dc</code>	Calculate spectral drift correction (C)
	<code>fn</code>	Fourier number in directly detected dimension (P)
	<code>intmod</code>	Integral display mode (P)
	<code>trace</code>	Mode for 2D data display (P)
	<code>wft1da</code>	Weight and Fourier transform phase-sensitive data (M)
	<code>wft2da</code>	Weight and Fourier transform phase-sensitive data (M)

bc2d **2D baseline correction (obsolete)**

Description: The `bc2d` macro is no longer functional. Use `bc` as the replacement.

Related: `bc` 1D and 2D baseline correction (C)

beepoff **Turn beeper off (C)**

Syntax: `beepoff`

Description: Turns off the beeper sound so that the system does not use sound to warn the user when errors occur. The default is the beeper is turned on.

See also: *VNMR User Programming*

Related: `beepon` Turn beeper on (C)

beepon **Turn beeper on (C)**

Syntax: `beepon`

Description: Turns on the beeper sound so that the user hears a sound when errors occur. The default is the beeper is turned on.

See also: *VNMR User Programming*

Related: `beepoff` Turn beeper off (C)

binom **Set up parameters for BINOM pulse sequence (M)**

Applicability: Sequence is not supplied with *MERCURY-VX*, *MERCURY* or *GEMINI 2000*.

Syntax: `binom`

Description: Sets up a binomial water suppression pulse sequence.

Alternate: BINOM button in the 1D Pulse Sequence Setup Menu.

See also: *User Guide: Liquids NMR*

bootup **Macro executed automatically when VNMR activated (M)**

Syntax: `bootup<(foreground)>`

Description: Executed automatically when VNMR is started up. The `bootup` macro displays a message, looks for a macro `login` in the user's local `maclib` directory and executes it (if found), starts `Acqstat` and `acqi` (`acqi` is not run if system is configured as a workstation), and then starts the menu system. This set of actions can be modified on a per user basis by constructing custom `bootup` or `login` macros in the user's `maclib` directory. A custom `login` macro is preferred because all custom `bootup` macros are overridden whenever a new VNMR release is installed.

Arguments: `foreground` is 0 if VNMR is being run in the foreground or nonzero if being run in the background. This argument is passed to the `login` macro.

See also: *VNMR User Programming*

Related: `acqi` Interactive acquisition display process (C)
`Acqstat` Bring up the acquisition status display (U)

boresize **Magnet bore size (P)**

Applicability: Systems with imaging capabilities.

Description: Holds the internal usable diameter of the gradient set. This parameter is used by various pulse sequence setup macros to determine the validity of the field of view and slice offset input. It is defined in the system gradient table files found in `$vnmr/system/imaging/gradtables`, and is automatically set from one of those files when a value is entered for `gcoil`.

Values: 18, 31, 33, 40 (nominal, in cm)

See also: *User Guide: Imaging*

Related: `creategtable` Generate new gradient calibration file (M)
`gcoil` Current gradient coil (P)
`gmax` Maximum gradient strength (P)
`setgcoil` Update system gcoil configuration (M)
`sysgcoil` System gradient coil (P)
`trise` Gradient rise time (P)

box **Draw a box on a plotter or graphics display (C)**

Syntax: `box(<'keywords' ,>x1mm,x2mm,y1mm,y2mm
<,'nolimit'><:r1,r2>`

Description: Draws a box on a plotter or a graphics display.

Arguments: `'keywords'` identifies the output device (`'graphics'` | `'plotter'`), drawing mode (`'xor'` | `'normal'`), and drawing capability (`'newovly'` | `'ovly'` | `'ovlyC'`).

- `'graphics'` | `'plotter'` is a keyword for the output device. The default is `'plotter'`. The output selected is passed to subsequent `pen`, `move`, or `draw` commands and remains active until a different output is specified.
- `'xor'`, `'normal'` is a keyword for the drawing mode when using the `'graphics'` output device. The default is `'normal'`. In the `'xor'` mode, if a line is drawn such that one or more points of the line are in common with a previous `'xor'` line, the common points are erased. In the `normal` mode, the common points remain. The mode selected is passed to

subsequent `pen`, `move`, and `draw` commands and remains active until a different mode is specified.

- `'newovly'`, `'ovly'` and `'ovlyC'` are keywords that specify an interactive drawing capability that is slightly slower than the `'xor'` mode but more consistent in color. `'newovly'` clears any previous draws, boxes, and writes made with the `'ovly'` modes and draws the figure. `'ovly'` draws without clearing so that multi-segment figures can be created. `'ovlyC'` clears without drawing.

`x1mm` is the left edge of the box, `x2mm` is the right edge, `y1mm` is the bottom, and `y2mm` is the top. The location of the edges are given in plotter units (mm on most plots) and are scaled in mm for the graphics display. (If units are in Hz or ppm, you can use the `hztomm` command to convert units.)

`'nolimit'` allows the box to extend outside the limits determined by the parameters `sc`, `wc`, `sc2`, and `wc2`.

`r1`, `r2` return the location of the upper left corner of the box.

Examples: `box('plotter', 20, 100, 40, 150)`
`box(25, 105, 45, 155, 'nolimit'):r1, r2`

See also: *Getting Started*

Related:	<code>gin</code>	Return current mouse position and button values (C)
	<code>hztomm</code>	Convert positions from Hz or ppm to plotter units (C)
	<code>sc</code>	Start of chart (P)
	<code>sc2</code>	Start of chart in second direction (P)
	<code>wc</code>	Width of chart (P)
	<code>wc2</code>	Width of chart in second direction (P)
	<code>wcmax</code>	Maximum width of chart (P)

boxes Draw boxes selected by the mark command (M)

Syntax: `boxes<('graphics' | 'plotter')>`

Description: Draws boxes on a plotter or a graphics display with the location of the edges given in Hz. The data to make the boxes is stored in the `mark2d.out` file produced by the `mark` command. If there is no data in `mark2d.out`, a box is drawn from the current cursor positions. The `boxes` command also numbers the boxes above the upper left corner.

Arguments: `'graphics' | 'plotter'` is a keyword to send output to the graphics display or to the plotter, respectively. The default is `'graphics'`.

Examples: `boxes`
`boxes('plotter')`

See also: *User Guide: Liquids NMR*

Related: `mark` Determine intensity of spectrum at a point (C)

bpa Plot boxed parameters (M)

Syntax: `bpa:$sc2_minimum`

Description: Plots a box around the entire chart (assuming blank paper) and then plots “chemist-style” parameters in boxes along the lower edge of the chart. `bpa` is the same as `ppa`, but with a different layout. Both `ppa` and `bpa` behave somewhat naively if the pulse sequence is more complex, but they were designed primarily for chemists, not for spectroscopists.

Arguments: `sc2_minimum` returns the minimum value for `sc2` to plot a scale properly. To use the command `pir`, `vp` has to be set to a non-zero value.

See also: *Getting Started*

Related:	<code>apa</code>	Plot parameters automatically (M)
	<code>pap</code>	Plot out “all” parameters (C)
	<code>pir</code>	Plot integral amplitudes below spectrum (C)
	<code>ppa</code>	Plot a parameter list in “English” (M)
	<code>sc2</code>	Start of chart in second direction (P)
	<code>vp</code>	Vertical position of spectrum (P)

br24 **Set up parameters for BR24 pulse sequence (M)**

Applicability: Systems with solids hardware. Sequence not supplied with *MERCURY-VX*, *MERCURY*, or *GEMINI 2000*.

Syntax: `br24`

Description: Converts a FLIPFLOP, MREV8, or S2PUL parameter set into a BR24 solids line-narrowing multiple-pulse sequence.

See also: *User Guide: Solid-State NMR*

Related:	<code>cylbr24</code>	Set up parameters for cycled BR24 pulse sequence (M)
	<code>flipflop</code>	Set up parameters for FLIPFLOP pulse sequence (M)
	<code>mrev8</code>	Set up parameters for MREV8 pulse sequence (M)
	<code>s2pul</code>	Set up standard two-pulse sequence (M)

browser **Start Image Browser application (U)**

Applicability: Systems with imaging capabilities.

Syntax: (From UNIX) `browser <macro_name> <XView_arguments>`
`<-image path> <-imagelist path>`

Description: Starts up the Image Browser application. Image Browser requires the environment variable BROWSERDIR to be set to point to the user’s directory `ib_initdir`, which contains initialization files and directories. The environment variable and the initialization directory can be created when the `makeuser` command is run.

Image Browser reads in files in Flexible Data Format (FDF) for displaying and processing. To generate files in FDF format, the following macros are available to write out single or multislice images:

- For the current imaging software, which includes sequences `sems`, `mems`, and `flash`, use the `svib` macro (the `svimg` macro is a precursor to `svib` and is being obsoleted).
- For older style SIS imaging sequences and microimaging sequences, use the macro `svsis`.
- 3D data can be saved in the FDF format by the `ft3d` macro.

The FDF format is an ASCII header describing the data, followed by the data. For more information on FDF, see the *VNMR User Programming* manual.

After images are read into Image Browser, it can write out image data in a number of other formats for use with other imaging applications.

`browser` can be used to extract up to three Maximum Intensity Projections (MIPs).

Arguments: Arguments can appear in any order.

`macro_name` is the file name of a macro, which must be stored in `$BROWSERDIR/macro/macro_name`. The macro is executed when Image Browser starts. If no macro name is specified, the macro `startup` is executed.

`XView_arguments` are any type of standard XView arguments, which can be found by typing `man xvview` on a UNIX command line.

`-image path` specifies the path of an image that should be loaded at startup. It is loaded after the `startup` macro is executed. Multiple `-image` arguments can be used to load multiple images.

`-imagelist path` specifies the path of a file containing a list of image files to be loaded.

See also: *User Guide: Imaging; VNMR User Programming*

Related: `fdfgluer` Make FDF file from header and data parts (C)
`ft3d` Perform a 3D Fourier transform on a 3D FID data set (M,U)
`svib` Generate and save images as Image Browser FDF files (M)
`svsis` Generate and save images as FDF files (M)

bs **Block size (P)**

Description: Directs the acquisition computer, as data are acquired, to periodically store a block of data on the disk, from where it can be read by the host computer.

CAUTION: If `bs='n'`, block size storage is disabled and data are stored on disk only at the end of the experiment. If the experiment is aborted prior to termination, data will be lost.

Values: 1 to 32767 transients, 'n'

See also: *Getting Started*

Related: `wbs` Specify action when `bs` transients accumulate (C)
`wbs` When block size (P)

btune **Tune broadband channel on MERCURY-Series and GEMINI 2000 (M)**

Applicability: *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* broadband systems.

Syntax: `btune`

Description: Turns on the broadband transmitter, directing to the probe about 0.5 watts of rf at frequency `sfrq`, enabling the user to tune the probe coil. Before entering `btune`, be sure to move the proper cable on the back of the left-hand magnet leg to the BNC connector labeled TUNE, and also to move the proper cable leading to the probe to the BNC connector labeled TUNE. Enter `tuneoff` to turn off the transmitter. `btune` cannot be executed while the console is acquiring or interactive acquisition (`acqi`) is connected. For the full tuning procedure, see the probe installation manual.

See also: *Getting Started; Autoswitchable NMR Probes Installation*

Related: `acqi` Interactive acquisition display process (C)
`ctune` Tune carbon channel on $^1\text{H}/^{13}\text{C}$ *GEMINI 2000* (M)
`dtune` Tune lock channel on *GEMINI 2000* (M)
`htune` Tune proton channel on *GEMINI 2000* (M)
`sethw` Set values for hardware in acquisition system (C)
`sfrq` Transmitter frequency of observe nucleus (P)
`su` Submit a setup experiment to acquisition (M)
`tuneoff` Turn off probe tuning mode, *MERCURY-Series*, *GEMINI 2000* (M)

c13 Automated carbon acquisition (M)

Syntax: `c13<(solvent)>`

Description: Prepares parameters for automatically acquiring a standard ^{13}C spectrum. The parameter `wexp` is set to 'procplot' for standard processing. If `c13` is used as the command for automation via the `enter` command, the `au` is supplied automatically and should not be entered on the MACRO line of the `enter` program. However, it is possible to customize the standard `c13` macro on the MACRO line by following it with additional commands and parameters. For example, `c13 nt=1` uses the standard `c13` setup but with only one transient.

Arguments: `solvent` is the name of the solvent. In automation mode the solvent is supplied by the `enter` program. The default is 'CDC13'.

Examples: `c13`
`c13('DMSO')`

See also: *Getting Started; User Guide: Liquids NMR*

Related:	<code>au</code>	Submit experiment to acquisition and process data (M)
	<code>c13p</code>	Process of 1D carbon spectra (M)
	<code>enter</code>	Enter sample information for automation run (C)
	<code>procl1d</code>	Processing macro for simple (non-arrayed) 1D spectra (M)
	<code>procplot</code>	Automatically process FIDs (M)
	<code>wexp</code>	When experiment completes (P)

c13p Process 1D carbon spectra (M)

Syntax: `c13p`

Description: Processes non-arrayed 1D carbon spectra using a set of standard macros. `c13p` is called by the `procl1d` macro, but can also be used directly. Fully automatic processing (up to a point where a spectrum could be plotted) is provided: Fourier transformation (using pre-set weighting functions), automatic phasing (`aphx` macro), automatic integration (`integrate` macro if required only), vertical scale adjustment (`vsadjc` macro), avoiding excessive noise (`noislm` macro), threshold adjustment (`thadj` macro), and referencing to the TMS signal if present (`setref` macro then `tmsref` macro).

See also: *Getting Started; User Guide: Liquids NMR*

Related:	<code>aphx</code>	Perform optimized automatic phasing (M)
	<code>c13</code>	Automated carbon acquisition (M)
	<code>integrate</code>	Automatically integrate 1D spectrum (M)
	<code>noislm</code>	Limit noise in spectrum (M)
	<code>procl1d</code>	Processing macro for simple (non-arrayed) 1D spectra (M)
	<code>setref</code>	Set frequency referencing for proton spectra (M)
	<code>thadj</code>	Adjust threshold (M)
	<code>tmsref</code>	Reference spectrum to TMS line (M)
	<code>vsadjc</code>	Adjust vertical scale for carbon spectra (M)

calcdim Calculate dimension of experiment (C)

Syntax: `calcdim`

Description: Calculates the dimension of an experiment and puts the result into the parameter `arraydim`. If an experiment is arrayed, `arraydim` is the product of the size of the arrays.

See also: *Getting Started*

Related: `arraydim` Dimension of experiment (P)

calfa Recalculate alfa so that first-order phase is zero (M)

Syntax: `calfa`

Description: Based upon the current `alfa` and `lp` values, `calfa` calculates a new value for `alfa` so that the first-order phase parameter `lp` is rendered approximately 0. When digital filtering is active (`dsp='r'` or `dsp='i'`), `calfa` also adjusts `rof2` as well as `alfa`. For `calfa` to work properly, a trial spectrum must be obtained and phased to pure absorption. This spectrum provides `calfa` with the current `alfa` and `lp` values. `calfa` pertains to processing 2D data. Unless `lp` is approximately 0, `fpmult` will affect both the `dc` offset and the curvature of the spectrum.

See also: *User Guide: Liquids NMR*

Related: `alfa` Set `alfa` delay before acquisition (P)
`cfpmult` Calculate first-point multiplier for 2D experiments (M)
`crof2` Recalculate `rof2` so that `lp` = 0 (M)
`dc` Calculate spectral drift correction (C)
`dsp` Type of DSP for data acquisition (P)
`fpmult` First-point multiplier for `np` FID data (P)
`hoult` Set parameters `alfa` and `rof2` according to Hoult (M)
`lp` First-order phase in directly detected dimension (P)
`rof2` Receiver gating time following pulse (P)

calibflag Correct systematic errors in DOSY experiments (P)

Syntax: `calibflag`

Description: Corrects systematic errors in DOSY experiments.

Values: 'y' corrects systematic deviations in DOSY analysis.
'n' omits gradient correction in DOSY analysis.

See also: *User Guide: Liquids NMR*

Related: `dosy` Process DOSY experiments (M)

calibrate Start a dialog for autocalibration routines (M)

Syntax: `calibrate`

Description: Starts a dialog for autocalibration routines.

capt Automated carbon and APT acquisition (M)

Syntax: `capt<(solvent)>`

Description: Prepares parameters for automatically acquiring a standard ^{13}C spectrum, followed by an APT experiment. In non-automation mode, the carbon and APT spectra are acquired in the experiment in which `capt` is entered. Following acquisition completes, the commands `rttmp('C13')` and `rttmp('apt')` can be used for further processing of the carbon and APT spectra, respectively.

Arguments: `solvent` is name of the solvent used. In automation mode, the `enter` program supplies name. In non-automation mode, the default is '`cdcl3`'.

Examples: `capt au`
`capt('dms')`

See also: *Getting Started; User Guide: Liquids NMR*

Related: `apt` Prepare parameters for APT experiment (M)
`c13` Automated carbon acquisition (M)
`enter` Enter sample information for automation run (C)
`rttmp` Retrieve experiment subfile (M)

CARBON Set up parameters for carbon spectrum (M)

Applicability: *GLIDE* only.

Syntax: `CARBON`

Description: Internal macro that sets up parameters for a carbon spectrum in *GLIDE*. `CARBON` is used only when carbon is selected in a proton experiment.

cat Display one or more text files in text window (C)

Syntax: `cat(file1<,file2,...>)`

Description: Displays the contents of one or more text files on the text window. It pauses after the window has filled and waits for the user to indicate whether it should display more or should terminate.

Arguments: `file1, file2, ...` are the names of the files to be displayed.

Examples: `cat('/vnmr/manual/cat')`
`cat('/vnmr/manual/cat','/vnmr/manual/cattn')`

See also: *Getting Started*

cattn Coarse attenuator type (P)

Applicability: Systems (not including *MERCURY-VX*, *MERCURY* and *GEMINI 2000*) with a coarse attenuator.

Description: Identifies the type of coarse attenuator if this attenuator is present on the current rf channel. The value of `cattn` is set in the CONFIG window (opened by entering `config`) using the label Coarse Attenuator.

Values: 0 for no coarse attenuator, as in the case with class C amplifiers (Not Present choice in CONFIG window).

63 for standard UNITY 63-dB attenuator (63 dB choice in CONFIG window).

79 for standard UNITY^{INOVA} or UNITY^{plus} 79-dB attenuator, optional on UNITY systems (79 dB choice in CONFIG window).

127 for imaging attenuator (63.5 dB SIS choice in CONFIG window).

63 for UNITY^{INOVA} deuterium decoupler channel.

See also: *VNMR and Solaris Software Installation*

Related: `config` Display current configuration and possibly change it (M)
`fattn` Fine attenuator (P)
`tpwr` Observe transmitter power level with linear amplifiers (P)

cd Change working directory (C)

Syntax: `cd<(directory)>`

Description: Changes current working directory to another directory.

Arguments: `directory` is the name of the directory that becomes the new current working directory. The change is made only if the directory name already exists and the user has permission to be in the directory. If no argument is included, `cd` changes the current working directory to the user's home directory.

Examples: `cd`
`cd(userdir+' /expl ')`
`cd('/home/george/vnmrsys ')`

See also: *Getting Started*

Related: `pwd` Display current working directory (C)

cdc Cancel drift correction (C)

Syntax: `cdc`

Description: Turns off the drift correction started by the `dc` command and resets the spectral drift correction parameters `lv1` (level) and `tl1t` (tilt) to zero.

See also: *Getting Started*

Related: `dc` Calculate spectral drift correction (C)
`dcg` Drift correction group (P)
`lv1` Zero-order baseline correction (P)
`tl1t` First-order baseline correction (P)

cdept Automated carbon and DEPT acquisition (M)

Syntax: `cdept<(solvent)>`

Description: Prepares parameters for automatically acquiring a standard ^{13}C spectrum, followed by a DEPT experiment. In non-automation mode, the carbon and DEPT spectra are acquired in the experiment in which `cdept` was entered. Following the completion of the acquisition, the `rttmp('C13')` and `rttmp('dept')` commands can be used for further processing of the carbon and DEPT spectra, respectively.

Arguments: `solvent` is name of the solvent used. In automation mode, the `enter` program supplies name. In non-automation mode, the default is `'cdc13'`.

Examples: `cdept au`
`cdept('DMSO')`

See also: *Getting Started; User Guide: Liquids NMR*

Related: `adept` Automatic DEPT analysis and spectrum editing (C)
`c13` Automated carbon acquisition (M)
`dept` Prepare parameters for DEPT experiment (M)
`enter` Enter sample information for automation run (C)
`rttmp` Retrieve experiment subfile (M)

celem Completed FID elements (P)

Description: Indicates the current number of completed FIDs in an experiment. When `go` or `au` is entered, `celem` is set to 0. As each FID acquisition is completed, `celem` is updated to reflect this. This parameter is most useful in conjunction with `wbs`, `wnt`, `wexp`, and `werr` processing commands.

See also: *Getting Started*

Related: `arraydim` Dimension of experiment (P)
`au` Submit experiment to acquisition and process data (C)

<code>go</code>	Submit experiment to acquisition (C)
<code>ni</code>	Number of increments in 1st indirectly detected dimension (P)
<code>wbs</code>	Specify action when <code>bs</code> transients accumulate (C)
<code>werr</code>	Specify action when error occurs (C)
<code>wexp</code>	Specify action when experiment completes (C)
<code>wnt</code>	Specify action when <code>nt</code> transients accumulate (C)

center **Set display limits for center of screen (C)**

Syntax: `center`

Description: Sets parameters `sc` and `wc` (horizontal control) and parameters `sc2` and `wc2` (vertical control) to produce a display (and subsequent plot) in the center portion of the screen (and page). For 2D data, space is left for the scales.

Alternate: Center button in the 1D Display Size Selection Menu
Center button in the 2D Display Size Selection Menu

See also: *Getting Started; User Guide: Liquids NMR*

Related:	<code>full</code>	Set display limits for a full screen (C)
	<code>fullt</code>	Set display limits for full screen with room for traces (C)
	<code>left</code>	Set display limits for left half of screen (C)
	<code>right</code>	Set display limits for right half of screen (C)
	<code>sc</code>	Start of chart (P)
	<code>sc2</code>	Start of chart in second direction (P)
	<code>wc</code>	Width of chart (P)
	<code>wc2</code>	Width of chart in second direction (P)

centersw **Move cursor to center of spectrum (M)**

Syntax: `centersw`

Description: Sets cursor position parameter `cr` in the directly detected dimension for the center of the spectrum.

See also: *Getting Started*

Related:	<code>centersw1</code>	Move cursor to center of spectrum in 1st indirect dimension (M)
	<code>centersw2</code>	Move cursor to center of spectrum in 2nd indirect dimension (M)
	<code>cr</code>	Cursor position in directly detected dimension (P)

centersw1 **Move cursor to center of spectrum in 1st indirect dimension (M)**

Syntax: `centersw1`

Description: Sets cursor position parameter `cr1` in the first indirectly detected dimension to the center of the spectrum.

See also: *User Guide: Liquids NMR*

Related:	<code>centersw</code>	Move cursor to center of spectrum (M)
	<code>cr1</code>	Cursor position in 1st indirectly detected dimension (P)

centersw2 **Move cursor to center of spectrum in 2nd indirect dimension (M)**

Syntax: `centersw2`

Description: Sets cursor position parameter `cr2` in the second indirectly detected dimension to the center of the spectrum.

See also: *User Guide: Liquids NMR*

Related: `centersw` Move cursor to center of spectrum (M)
`cr2` Cursor position in 2nd indirectly detected dimension (P)

cexp Create a VNMR experiment (M)

Syntax: `cexp (<experiment_dir , >experiment_number)`

Description: Creates a VNMR experiment as a temporary workspace that can hold a complete 1D, 2D, or 3D data set. Up to 9999 experiments can be created. Experiment 5 is special because it is the add-subtract experiment. `cexp` creates the appropriate `jexpxxx` macro so that the newly created experiment can be joined.

Arguments: `experiment_dir` specifies the path of the directory in which the particular experiment is to be created. If `experiment_dir` is not entered, the default is the VNMR user directory specified by `userdir`.

`experiment_number` specifies the number, from 1 to 9999, of the VNMR experiment to be created.

Alternate: Create New button in the Workspace Menu.

Examples: `cexp (3)`
`cexp (' /data ' , 2)`

See also: *User Guide: Liquids NMR*

Related: `delexp` Delete an experiment (C)
`jexp` Join existing experiment (C)
`userdir` VNMR user directory (P)

cf Current FID (P)

Description: Specifies which FID to operate on when working with multi-FID data. All subsequent operations such as Fourier transformation are applied to the selected data block.

When an experiment acquires `nf` number of data segments through explicit acquisition, `cf` indicates the `cf`th FID to use. For example, in the COSY-NOESY experiment with `nf=2`, `cf=1` would select the COSY part of the experiment, and `cf=2` would select the NOESY part.

Values: 1 through the value of parameter `nf`.

See also: *User Guide: Imaging*

Related: `nf` Number of FIDs (P)

cfpmult Calculate first-point multiplier for 2D experiments (M)

Syntax: `cfpmult`

Description: Calculates an `fpmult` value for the dataset, which is then used by `wft2da`. For 2D experiments, such as NOESY, run `cfpmult` on the transformed first increment, prior to entering `wft2da`, to minimize “f₂ ridges” in the final 2D spectrum. To do this manually for a 2D dataset, enter `fpmult=1.0 wft (1) cdc` in the VNMR command line and note whether the spectrum (essentially the baseline) moves up or down when `dc` is typed. Vary the value of `fpmult` until the dc correction (jump in the baseline) is as small as possible. With care, `fpmult` can be set to two decimal places. Typical values for `fpmult` range from 1.00 to 2.00. The default value is 1.0.

This calculation only needs to be performed for cosine-type experiments, such as NOESY, where both the t_2 FID and the t_1 interferogram decay. `cfpmult` might give incorrect values for first increments of experiments having baseline distortions (e.g., water suppression with 11-echo or 1331); in such cases, manual optimization of `fpmult` is more suitable.

When processing 2D data, unless the parameter `lp` is approximately 0, `fpmult` affects both the dc offset and the curvature of the spectrum. See the entries for `alfa` and `calfa` for more information.

See also: *User Guide: Liquids NMR*

Related:	<code>alfa</code>	Set <code>alfa</code> delay before acquisition (P)
	<code>calfa</code>	Recalculate <code>alfa</code> so that first-order phase is zero (M)
	<code>crof2</code>	Recalculate <code>rof2</code> so that <code>lp</code> = 0 (M)
	<code>dc</code>	Calculate spectral drift correction (C)
	<code>fpmult</code>	First point multiplier for <code>np</code> FID data (P)
	<code>lp</code>	First-order phase in directly detected dimension (P)
	<code>wft2da</code>	Weight and Fourier transform phase-sensitive data (M)

change **Submit a change sample experiment to acquisition (M)**

Applicability: Systems with automatic sample changer.

Syntax: `change`

Description: Removes the sample currently in the probe and loads the sample currently in sample location `loc`. `change` runs in the acquisition computer and is inoperative if `loc` is 0 and/or `traymax` is 'n' or 0. `change` also sets all hardware according to the current parameters.

See also: *User Guide: Liquids NMR*

Related:	<code>au</code>	Submit experiment to acquisition and process data (C)
	<code>ga</code>	Submit experiment to acquisition and FT the result (C)
	<code>go</code>	Submit experiment to acquisition (C)
	<code>loc</code>	Location of sample in tray (P)
	<code>lock</code>	Submit an autolock experiment to acquisition (C)
	<code>sample</code>	Submit change sample, Autoshim experiment to acquisition (M)
	<code>shim</code>	Submit an Autoshim experiment to acquisition (C)
	<code>spin</code>	Submit a spin setup experiment to acquisition (C)
	<code>su</code>	Submit a setup experiment to acquisition (M)
	<code>traymax</code>	Sample changer tray size (P)

c1a **Clear all line assignments (M)**

Syntax: `c1a`

Description: Clears the line assignment parameters `clindex` and `slfreq` for spin simulation iteration, which matches simulated spectra to actual data.

See also: *User Guide: Liquids NMR*

Related:	<code>assign</code>	Assign transitions to experimental lines (M)
	<code>d1a</code>	Display line assignments (M)
	<code>clindex</code>	Index of experimental frequency of a transition (P)
	<code>slfreq</code>	Measured line frequencies (P)

c1a **Calculated transition number (P)**

Description: A global arrayed parameter that stores the transition number of calculated transitions of the spin simulation program when they are above a threshold set

by `sth`. In the iterative mode, the `cla` value of an assigned transition is associated with an experimental frequency whose index is the `clindex` value.

See also: *User Guide: Liquids NMR*

Related:	<code>clamp</code>	Calculated transition amplitude (P)
	<code>clfreq</code>	Calculated transition frequency (P)
	<code>clindex</code>	Index of experimental frequency of a transition (P)
	<code>sth</code>	Minimum intensity threshold (P)

clamp **Calculated transition amplitude (P)**

Description: A global arrayed parameter that stores the transition amplitude of calculated transitions of the spin simulation program when they are above a threshold set by the parameter `sth`. Enter `dla('long')` to display `clamp`.

See also: *User Guide: Liquids NMR*

Related:	<code>cla</code>	Calculated transition number (P)
	<code>clfreq</code>	Calculated transition frequency (P)
	<code>clindex</code>	Index of experimental frequency of a transition (P)
	<code>dla</code>	Display line assignments (C)
	<code>sth</code>	Minimum intensity threshold (P)

cleanexp **Remove old files and directories from an experiment (M)**

Syntax: `cleanexp<(file1<,file2<,...>>>`

Description: Removes experiment subfiles from chained experiments that exist in an experiment directory. `cleanexp` only cleans the currently active experiment.

Arguments: `file1`, `file2`, ... are specific experiment subfiles to be removed. If no argument is given, all files in `curexp/subexp` are removed.

Examples: `cleanexp`
`cleanexp('H1','relayh')`

See also: *Getting Started; User Guide: Liquids NMR*

Related:	<code>curexp</code>	Current experiment directory (P)
	<code>hccorr</code>	Automated proton, carbon, and HETCOR acquisition (M)
	<code>hcosy</code>	Automated proton and COSY acquisition (M)

clear **Clear a window (C)**

Syntax: `clear<(window_number)>`

Description: Clears one of the four windows on the GraphOn terminal (status, input, graphics, text) or one of the two windows on the Sun (text and graphics).

Arguments: `window_number` is the number (1 to 4) of the window to be cleared:

- 1 clears the status window (GraphOn only)
- 2 clears the graphics window
- 3 clears the input window (GraphOn only)
- 4 clears the text window (the default value).

Examples: `clear`
`clear(2)`

See also: *VNMR User Programming*

cleardosy Delete temporarily saved data in current subexperiment (M)Syntax: `cleardosy`

Description: Deletes any copies of DOSY data temporarily saved in the current subexperiment.

See also: *User Guide: Liquids NMR*Related: `dosy` Process DOSY experiments (M)**clfreq Calculated transition frequency (P)**Description: A global arrayed parameter that stores the transition frequency of calculated transitions of the spin simulation program when they are above a threshold set by the parameter `sth`. Enter `dla` to display `clfreq`.See also: *User Guide: Liquids NMR*

Related: `cla` Calculated transition number (P)
`clamp` Calculated transition amplitude (P)
`clindex` Index of experimental frequency of a transition (P)
`dla` Display line assignments (M)
`sth` Minimum intensity threshold (P)

clindex Index of experimental frequency of a transition (P)Description: A global arrayed parameter where each value contains the index of an experimental frequency assigned to the associated calculated transition for use in iterative spin simulation. Use `assign` to make the assignments. A value of zero indicates no assignment.See also: *User Guide: Liquids NMR*

Related: `assign` Assign transitions to experimental lines (M)
`cla` Clear line assignments (M)
`cla` Calculated transition number (P)
`dla` Display line assignments (M)

clradd Clear add/subtract experiment (C)Syntax: `clradd`Description: Deletes the add/subtract experiment (`exp5`).

Alternate: clear button in the Add/Subtract Menu.

See also: *User Guide: Liquids NMR*

Related: `add` Add current FID to add/subtract experiment (C)
`sub` Subtract current FID from add/subtract experiment (C)

color Select plotting colors from a graphical interface (M)Syntax: `color`

Description: Displays a window with color palettes for selecting colors for plotting the background of the VNMR display screen, spectrum, integral, FID, etc.

See also: *Getting Started*

Related: `pl` Plot spectra (C)
`setcolor` Set colors for graphics window and for plotters (C)

- combiplate** **View a color map for visual analysis of VAST microtiter plate (U)**
- Syntax: (From UNIX) `combiplate`
- Description: Opens the CombiPlate window, which provides a map of microtiter plate, allowing data to be viewed from individual sample wells. The window enables viewing integral region intensities by colors and color densities.
- See also: *User Guide: Liquids NMR*
- Related: `combishow` Display regions as red, green, and blue in CombiPlate window (M)
`dlivast` Produce text file and process last wells (M)
- combishow** **Display regions as red, green, and blue in CombiPlate window (M)**
- Syntax: `combishow(r,g,b)`
- Description: Displays integral regions shown on the spectrum as red (r), green (g), and blue (b) in the CombiPlate window. CombiPlate reads the regions automatically. 1, 2, or 3 integral regions can be designated. At least one integral region must be specified. Combishow displays spectra associated with individual wells.
- See also: *User Guide: Liquids NMR*
- Related: `combiplate` View a color map for visual analysis of VAST microtiter plate (U)
`dlivast` Produce text file and process last wells (M)
- compressfid** **Compress double-precision VNMR FID data (M,U)**
- Syntax: (From VNMR) `compressfid(<inFIDdir,>outFIDdir)`
(From UNIX) `compressfid -i inFIDdir -o outFIDdir -f`
(From UNIX) `compressfid -e exp_number -o outFIDdir -f`
- Description: Compresses double-precision VNMR FID data to single-precision and updates the parameter `dp` in the file `procpur`. `compressfid` can be run through a macro interface in VNMR or directly at the UNIX level. In entering FID directory names, leave off the `.fid` directory extension.
- Arguments: `inFIDdir` is the double-precision FID directory to be compressed. If `inFIDdir` is not entered, the default FID directory is `curexp/acqfil`.
`outFIDdir` is the FID directory to receive the output.
`exp_number` is the number of the experiment that contains the FID data.
`-i` specifies that the next argument is the input FID directory.
`-o` specifies that the next argument is the output FID directory.
`-e` specifies that the next argument is the number of the experiment that contains the FID data. The `-e` and the `-i` options are mutually exclusive.
`-f` specifies that any existing directory with the name `outFIDdir.fid` is to be overwritten. Note that the macro interface in VNMR always overwrites any preexisting directory with the name specified by `outFIDdir.fid`.
- Examples: (From VNMR) `compressfid('/vnmr/fidlib/fidld', 'testfidld')compressfid('testfidld')`
(From UNIX) `compressfid -e 5 -o testfidld -f`
(From UNIX) `compressfid -i /vnmr/fidlib/fidld -o testfidld -f`
- See also: *Getting Started*
- Related: `dp` Double precision (P)

config **Display current configuration and possibly change it (M)**

Syntax: `config <('display')>`

Description: Displays the current system configuration parameters in a window (called the CONFIG window). The values of the configuration parameters can be changed if `config` is entered from the console without any arguments and the user has write access to the directories `/vnmr` and `/vnmr/compare`. If so, the user can interactively make changes to the choices in the window. Usually, `vnmr1`, the VNMR system administrator, configures the system and then sets the protection on `/vnmr/compare` to permit read-only access by other users.

If the user does not meet the conditions above, or if the VNMR system administrator enters the command `config('display')`, instead of the interactive mode, the user is restricted to the display mode. In this mode, the CONFIG window appears but the user cannot make any changes. On *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*, the mode is always interactive.

If `config` is entered without any arguments, the program checks if the user is logged in as `vnmr1`. If so, it runs in interactive mode; if not, it runs in display mode. By entering `config('display')`, `vnmr1` can run in the display mode instead of interactively.

In the interactive mode, a separate panel displays the options with the current choice appearing to the right. Position the mouse over the choice to be modified, then use the left button to cycle through each choice or use the right button to display a menu of all possible choices.

On systems other than *MERCURY*, and *GEMINI 2000*, the Use Console Data button sets parameter values in the CONFIG window using information captured during console startup.

- On *UNITYINOVA*, this button makes `config` capture from the system all values shown in the CONFIG window except Sample Changer, Sample Changer Serial Port, Rotor Synchronization, Frequency Overrange, and Upper Limit of decoupler power. For the Gradients entry, `config` recognizes the Performa I and Performa II modules but not other gradients. For the VT Controller entry, if VT is found, `config` does not change the value set, and if VT is not found, `config` changes the value to Not Present.
- On *MERCURY-VX* systems, this button captures all the values except Sample Changer and Sample Changer Serial Port. The VT Controller entry is set the same way as *UNITYINOVA* systems (see above).
- On *UNITYplus*, this button captures all values except Sample Tray Size, Rotor Synchronization, Frequency Overrange, Upper Limit of decoupler power, and Gradients. The VT Controller entry is set the same way as *UNITYINOVA* systems (see above).
- On *UNITY* and *VXR-S* systems, this button captures only System Type, Maximum Spectral Width, and Fifo Loop Size parameter values.

The EXIT, and SAVE button writes a new `compare` configuration file before leaving. The QUIT, no SAVE button terminates the session with no modifications to the `compare` file, but remember that the parameters in VNMR are always set. These two buttons require use of the left button on the mouse. In the display mode, the current choices are displayed in the text window.

To send output to the printer, enter the sequence of commands `printon config('display') printoff`.

Commands for working with parameters (such as `create`, `destroy`, `exists` and `setvalue`) have an option to select which parameter tree the

parameter is in. The `systemglobal` tree is the internal VNMR name for `/vnmr/conpar`, and it can be used to search for, modify, or create a parameter in `conpar`. But note that any changes made, either directly (e.g., by typing `vttype=0`) or by using `create` and similar commands, only affect parameters in memory. To permanently change parameters:

- For parameters in `config`, enter the change in the CONFIG window and then quit using the Exit & Save button.
- For other parameters, after creating or changing the parameter, enter `fsave('/vnmr/conpar','systemglobal')`.

Both methods, usually restricted to `vnmr1` only, overwrite `conpar`.

The CONFIG labels listed below can be changed in the interactive mode. For each label, the choices available and a short description of the label is provided. Shown in parentheses is the associated VNMR parameter, which you should refer to for further information.

CONFIG window on ^{UNITY}INOVA, UNITYplus, UNITY, VXR-S, and Imaging systems:

- System Type: Spectrometer or Data Station. Sets the basic type of system (`system`).
- Console: VXR-S, Unity, UnityPlus, UnityInova, Gemini 2000, *MERCURY*, or SISCO Imager. Sets the type of system console (`Console`). When `go`, `au`, or `ga` is entered, the value set is copied to the current experiment as the `console` parameter (lowercase c).
- Proton Frequency: 085, 100, 200, 300, 400, 500, 600, 750, 800, 900, 3T, and 4T. Sets the resonant frequency, in MHz or tesla, of ¹H as determined by magnet field strength (`hlfreq`).
- Sample Changer: For ^{UNITY}INOVA – None, Carousel, SMS 50 Sample, SMS 100 Sample, VAST, NMS, LC-NMR. For *UNITYplus*, UNITY, and VXR-S – None, Carousel, SMS/ASM 50 Sample, SMS/ASM 100 Sample, VAST, NMS, LC-NMR. Sets the type of sample changer. Set to none if a sample changer is not present or is to be disabled (`traymax`).
- Sample Changer Serial Port: Not Used, Port A, Port B. Sets the serial port used to connect the sample changer. Select Not Used if no sample changer is present (`smsport`).
- Shimset: Varian 13 Shims, Varian 14 Shims, Oxford 15 Shims, Oxford 18 Shims, Varian 18 Shims, Varian 20 Shims, Varian 23 Shims, Varian 26 Shims, Varian 28 Shims, Varian 29 Shims, Varian 35 Shims, Varian 40 Shims, Ultra Shims, and Whole Body. Sets type of shim sets on system (`shimset`).
- Audio Filter Type: 100 kHz Elliptical, 100 kHz Butterworth 200 kHz Butterworth, 500 kHz Elliptical. If the spectral width (sw) is less than 100 kHz, sets type of audio filters used (`audiofilter`).
- VT Controller: Not Present, Present. Sets whether a variable temperature controller is present or not on the system (`vttype`).
- Maximum DMF: 9900, 32700, 2.0e6. Sets maximum frequency, in Hz, for decoupler modulation (`parmax[11]`).
- Max. Spectral Width: 100 kHz, 200 kHz, 500 kHz, 2 MHz, 5 MHz. Sets maximum spectral width available to a system (`parmax[5]`).
- Max. Narrowband Width: 100 kHz, 200 kHz, 500 kHz. Defines the maximum spectral width of the Input board (`maxsw_loband`).

- AP Interface Type: Type 1, Type 2, Type 3, N/A. Sets type of AP bus interface board in the system (`apinterface`).
- Fifo Loop Size: 63, 1024, 2048. Sets size of FIFO loop, which depends on the type of controller board in the system (`fifolpsize`).
- Rotor Synchronization: Not Present, Present. Sets whether system supports the solids rotor synchronization module (`rotorsync`).
- Lock Frequency: (number entered directly). Sets lock frequency of the system. **To observe NMR signals, the lock frequency value must be set correctly** (`lockfreq`).
- IF Frequency: 10.0 MHz, 10.5 MHz, (20.0 MHz ^{UNITY}INOVA only).
- Number of RF Channels: 1, 2, 3, 4, 5. Selects which rf channel is listed in the Configure panel that appears in the lower section of the CONFIG window (`numrfch`).
- Gradients: Not Present, Present. Sets whether system has optional gradients for the X, Y, or Z axis. If present, the gradients are listed in the Configure panel in lower section of CONFIG window (Gradients is not associated with any VNMR parameter).
- Configure: RF Channel 1 (Obs), RF Channel 2 (Dec), RF Channel 3 (Dec2), RF Channel 4 (Dec 3), RF Channel 5 (Dec4), Gradients. Sets which labels appear in the Configure panel in lower section of CONFIG window (Configure is not associated with any VNMR parameter)
- Type of RF: U+ Direct Synthesis, U+ H1 Only, Direct Synthesis, Broadband, Fixed Frequency, Deuterium Decoupler (^{UNITY}INOVA only), SIS Modulator. Sets type of frequency generation on the current rf channel (`rftype` and `rfchtype`).
- Synthesizer: Not Present, PTS 160, PTS 200, PTS 250, PTS, 320, PTS 500, PTS 620, PTS 1000. Sets type of PTS frequency synthesizer on the current rf channel (`ptsval`).
- Latching: Not Present, Present. On systems equipped with a special version of the PTS frequency synthesizer, sets how frequency values are sent on the current rf channel (`latch`).
- Frequency Overrange: Not Present, 10000 Hz, 100000 Hz. On systems equipped with a special version of the PTS frequency synthesizer, sets the presence of a signal phase stability option on the current rf channel (`overrange`).
- Step Size: 0.1 Hz, 0.2 Hz, 1 Hz, 100 Hz. Sets frequency step size on current rf channel. (`parstep`[7], `parstep`[8], `parstep`[16], `parastep`[20]).
- Coarse Attenuator: Not Present, 63 dB, 79 dB, 63.5 dB (SIS). Sets range of coarse attenuator if this attenuator is present on the current rf channel (`cattn`).
- Upper Limit: (number entered directly). Sets upper limit of the coarse attenuator if this attenuator is present on the current rf channel (`parmax`[17], `parmax`[9], `parmax`[18], `parmax`[21]).
- Fine Attenuator: Not Present, Present. Sets whether a fine attenuator is present or not on the current rf channel (`fattn`).
- Waveform Generator: Not Present, Present. Sets whether a waveform generator board is present or not on current rf channel (`rfwg`).

- Type of Amplifier: Class C, Linear Full Band, Linear Low Band, Shared, Linear Broadband. (Shared is fourth channel only.) Sets type of amplifier on the current rf channel (`amptype`).
- X Axis, Y Axis, Z Axis: None, WFG + GCU, Performa I, Performa II/III, Performa II/III+WFG, Performa XYZ, Performa XYZ+WFG, SIS (12 bit), Homospoil. On systems with gradients, sets type of gradient for each axis. The value is set separately for each axis (`gradtype`).
- Imaging Gradient Coil. Detects the gradient coil configuration file that defines the current installed gradient coil (`sysgcoil`).

CONFIG window on *MERCURY-VX* and *MERCURY* systems:

Several parameters, other than those listed below, are set automatically because they have only one choice (e.g., `Console` is set to '*MERCURY*').

- System Type: 4-Nucleus, Broadband. Sets the basic type of system (`rftype`).
Effective June 2000, the *MERCURY-VX* 300-MHz 4-Nucleus system uses the Hi/Lo Reference Generator board. For this system, in CONFIG window set System Type to Broadband (`rftype='fe'`).
If the board type is unknown, look at the rf card cage in the back of the console. The third rf board from the left is the reference generator. If the top of the board is labeled Hi/Lo, select Broadband, but if it is labeled 4-Nucleus or 5-Nucleus select 4-Nucleus as the system type
- Proton Frequency: 200, 300, 400. Sets the resonant frequency, in MHz, of ¹H, as determined by magnet field strength (`h1freq`).
- VT Controller: Not Present, Present. Sets whether a variable temperature controller is present or not on the system (`vttype`).
- Auto Spinner: Not Present, Present. Sets whether spin hardware is present or not on the system (`spinopt`).
- Fine Attenuator: Not Present, arrayed as 0, 0. Present, arrayed as, 255, 0. Sets whether a fine attenuator is present or not on the high band transmitter (`fattn`).
- Waveform Generator: Not Present, setting is 'nn'. Present, setting is 'yn'. Sets whether a waveform generator board is present or not on the high band transmitter (`rfgw`).
- Type of Amplifier: 4-Nucleus (35W/35W), Broadband (75W/125W), CP/MAS(125W/300W). Sets type of amplifier in the system (`amptype`: aa on 4-Nucleus, bb on Broadband, cc on CP/MAS).
- Sample Changer: – None, Carousel, SMS 50 Sample, SMS 100 Sample, VAST, NMS. Sets the type of sample changer. Set to None if a sample changer is not present or is to be disabled (`traymax`).
- Sample Changer Serial Port: For only *MERCURY-VX* – Not Used, Port A, Port B. Sets the serial port used to connect the sample changer. Select Not Used if no sample changer is present (`smsport`).
- Pulsed Field Gradients: Not Present, Homospoil, Performa I. Sets whether the PFG hardware is present or not on the system (`gradtype`). Homospoil can be used for gradient shimming, but not for experiments like gHMQC.
- Lock Frequency: (number entered directly). Sets the lock frequency of the system. **This value must be set correctly to observe NMR signals** (`lockfreq`).
- Homodecoupler: Not Present, Present. Sets whether a homonuclear decoupler board is present or not (`homdec`).

- Max. Decoupler: (number entered directly). On broadband systems, sets maximum power level for CW decoupling (`parmax[9]`).

CONFIG window on GEMINI 2000 systems:

Several parameters, other than those listed below, are set automatically because they have only one choice (e.g., `Console` is set to `'g2000'`)

- System Type: 1H/13C, Broadband. Sets the basic type of system (`rftype`).
- Proton Frequency: 200, 300, 400. Sets the resonant frequency, in MHz, of ^1H , as determined by magnet field strength (`hlfreq`).
- VT Controller: Not Present, Present. Sets whether a variable temperature controller is present or not on the system (`vttype`).
- Auto Spinner: Not Present, Present. Sets whether spin hardware is present or not on the system (`spinopt`).
- Sample Changer: None, Carousel, ASM/SMS 50 Sample, ASM/SMS 100 Sample. Sets the type of sample changer. Set to None if a sample changer is not present or is to be disabled (`traymax`).
- Pulsed Field Gradients: Not Present, Present. Sets whether the PFG hardware is present or not on the system (`gradtype`).
- Lock Frequency: (number entered directly). Sets the lock frequency of the system. **This value must be set correctly to observe NMR signals** (`lockfreq`).
- Homodecoupler: Not Present, Present. On $^1\text{H}/^{13}\text{C}$ systems, sets whether a homonuclear decoupler board is present or not (`homdec`).
- Homo Dec. Offset: (number entered directly). On $^1\text{H}/^{13}\text{C}$ systems, sets the homonuclear decoupler offset to compensate for differences in individual boards. The installation engineer sets the value, and it should not require changing by the user (`hdofst`).
- BB Atten Type: Slow, Fast. On broadband systems, sets the version of RF Control board (`attens`).
- Max. Decoupler: (number entered directly). On broadband systems, sets maximum power level for CW decoupling (`parmax[9]`).

Arguments: `'display'` is a keyword that the VNMR system administrator can use to make `config` run in the display mode rather than the interactive mode.

Alternate: Hardware button in the Configuration Menu.

Examples: `config`
`config('display')`

See also: *VNMR and Solaris Software Installation*

Related:	<code>amptype</code>	Amplifier type (P)
	<code>apinterface</code>	AP Interface board type (P)
	<code>attens</code>	Fast attenuators present (P)
	<code>audiofilter</code>	Audio filter type (P)
	<code>cattn</code>	Coarse attenuator (P)
	<code>Console</code>	System console type (P)
	<code>fattn</code>	Fine attenuator (P)
	<code>fifolpsize</code>	FIFO loop size (P)
	<code>gradtype</code>	Gradients for X, Y, and Z axes (P)
	<code>hlfreq</code>	Proton frequency of spectrometer (P)
	<code>hdofst</code>	Proton homonuclear decoupler offset (P)
	<code>homdec</code>	Proton homonuclear decoupler present (P)

<code>latch</code>	Frequency synthesizer latching (P)
<code>lockfreq</code>	Lock frequency (P)
<code>maxsw_loband</code>	Maximum spectral width of Input board (P)
<code>numrfch</code>	Number of rf channels (P)
<code>overrange</code>	Frequency synthesizer overrange (P)
<code>parmax</code>	Parameter maximum values (P)
<code>parmin</code>	Parameter minimum values (P)
<code>parstep</code>	Parameter step size values (P)
<code>ptsval</code>	PTS frequency synthesizer value (P)
<code>rfchtype</code>	Type of rf channel (P)
<code>rftype</code>	Type of rf generation (P)
<code>rfg</code>	RF waveform generator (P)
<code>rotorsync</code>	Rotor synchronization (P)
<code>shimset</code>	Type of shim set (P)
<code>spinopt</code>	Spin automation (P)
<code>sysgcoil</code>	System gradient coil (P)
<code>system</code>	System type (P)
<code>traymax</code>	Sample changer tray slots (P)
<code>vttype</code>	Variable temperature controller present (P)

confirm **Confirm message using the mouse (C)**

Syntax: `confirm(message):response`

Description: Displays a dialog box with the specified message and two buttons: Confirm and Cancel. Clicking on the buttons with the mouse produces a return value.

Arguments: `message` is a single-line muticharacter string to be shown in the dialog box.
`response` is 1 if the user clicks the left button of the mouse on the Confirm button or presses the Return key; `response` is 0 if the user clicks the mouse on the Cancel button.

Examples: `confirm('Are you sure you want pw>100?'):$response`

See also: *VNMR User Programming*

Console **System console type (P)**

Description: A global parameter that sets the type of system console: *UNITYINOVA*, *MERCURY-VX*, *MERCURY*, *SISCO Imager*, *GEMINI 2000*, *UNITYplus*, *UNITY*, or *VXR-S*. The value is usually set using the Console label in the CONFIG window (opened from `config`); however, on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems, the value is automatically set.

When `go`, `au`, or `ga` is entered, the value of the `Console` parameter is copied from the `systemglobal` parameter tree to the current experiment and named as the `console` parameter (lowercase c). If `console` does not exist in an old parameter set, `rt` via `fixpar` creates it and sets it to ' '. Both `console` and `Console` are type acquisition. Macros can use `Console` and `console` to take conditional action based on spectrometer type.

Values: `'g2000'` is a *GEMINI 2000* console (Gemini 2000 choice in CONFIG window).

`'inova'` is a *UNITYINOVA* console (UnityInova choice in CONFIG window).

`'mercury'` is a *MERCURY-VX* or *MERCURY* console.

`'sisco'` is a *SISCO imager* console (sisco choice in CONFIG window).

`'unity'` is a *UNITY* console (Unity choice in CONFIG window).

'uplus' is a UNITY*plus* console (UnityPlus choice in CONFIG window).

'vxrs' is a VXR-S console (VXR-S choice in CONFIG window).

See also: *VNMR and Solaris Software Installation*

Related:	au	Submit experiment to acquisition and process data (M)
	config	Display current configuration and possibly change it (M)
	fixpar	Correct parameter characteristics in experiment (M)
	ga	Submit experiment to acquisition and FT the results (M)
	rt	Retrieve FIDs (M)
	go	Submit experiment to acquisition (M)
	system	System type (P)

contact_time MAS cross-polarization spin-lock contact time (M)

Applicability: Systems with solids module.

Syntax: `contact_time`

Description: Processes data obtained using an array of values for a pulse-length parameter. It runs the UNIX program `expfit`, which does an exponential curve fitting that determines the value of *Tch* and *Tlrho*. The output is matched to the equation

$$I = [S_{inf} - (S_{inf} - S_0) * \exp(-T/Tch)] * \exp(-T/Tlrho) + S_0$$

where *Tch* is the time constant of a spin-locked cross-polarization process, and *Tlrho* is relaxation time of ¹³C polarization in the proton rotating field.

The required input is file `fp.out` from the program `fp` and the values of the arrayed parameter. The output table is file `analyze.list` in the current experiment. The file `analyze.out` is used by the `expl` to display the results.

See also: *User Guide: Solid-State NMR*

Related:	expfit	Least-squares fit to polynomial or exponential curve (U)
	expl	Display polynomial/exponential curves (C)
	fp	Find peak heights (C)

conv2ta Convert imaging 3D transform to absolute value (U)

Applicability: Systems with imaging capabilities.

Syntax: (From UNIX) `conv2ta in_file out_file scaling_factor`

Description: Converts a complex 3D transformed data file into a 3D 8-bit absolute value data file suitable for viewing by using `disp3d`. The `conv2ta` command reads the header in the transformed file, typically named `filename.transform`, to determine the dimensions of the data, takes the magnitude of the complex data, scales the data, and writes out only the data (with no header) in 8-bit pixels. It also prints out the dimensions of the file that will be needed by `disp3d`.

Arguments: `in_file` is a valid UNIX file name of the 3D transformed data file.

`out_file` is a valid UNIX file name of the output file in 8-bit bytes.

`scaling_factor` is a value to scale the data so that it is in a range for viewing by `disp3d`. Reasonable values generally range from 1 to 4000. A value of 1000 is typical.

Examples: (From UNIX) `conv2ta kiwi3d.transform kiwi3d.av 1000`

See also: *User Guide: Imaging*

Related:	acqmeter	Open Acqmeter window (M)
	acqstat	Open Acquisition status window (U)

`disp3d` Convert 3D data (U)
`sa` Stop acquisition (C)

convert Convert data set from a VXR-style system (M,U)

Syntax: (From VNMR) `convert(VXR_file)`
 (From UNIX) `cpos_cvt VXR_file`

Description: Converts data stored on a VXR-style system (VXR, XL, or Gemini) to the format used in VNMR software. The VNMR macro `convert` loads the data from `VXR_file` into the current experiment and converts it to the new format. The UNIX command `cpos_cvt` writes the converted data in a subdirectory of the current working directory, using the original name of the data set.

Arguments: `VXR_file` is the name of a VXR-style file to be converted to VNMR style

See also: *Getting Started*

Related: `cpos_cvt` Convert data set from a VXR-style system (C,U)
`decomp` Decompose a VXR-style directory (C)
`unix_vxr` Convert UNIX text files to VXR-style format (M,U)
`vxr_unix` Convert VXR-style text files to UNIX format (M,U)

convertbru Convert Bruker data (M,U)

Syntax: (From UNIX) `convertbru file <options>`
 (From VNMR) `convertbru(file<,options>)`

Description: A C-language program for converting 32-bit Bruker AMX data and 24- and 32-bit Bruker AM data into a 32-bit format compatible with the Varian `sread` program. After converting the Bruker data into the new format, the converted data can be read into VNMR using `sread` and can then be processed normally. The parameters `proc` and `procl` are set appropriately by `sread`, so that `wft` or `wft2da` correctly processes the data.

Bruker AM parameters are converted to Varian parameters as shown in the table "AM Parameter Conversion." Bruker parameter names that do not conflict with a Varian parameter name are converted under the original name: `td`, `fw`, `ds`, `o1`, `o2`, `ns`, `te`, `id`, `sfo1`, `sfo2`, and `ro`. Parameters `proc` and `procl` are set to 'rft' for all spectra (assuming TPPI data in both dimensions).

AM Parameter Conversion

<i>Bruker</i>	<i>Varian</i>	<i>Bruker</i>	<i>Varian</i>
sweeps completed	<code>ct</code>	<code>sp</code>	<code>satdly</code>
<code>td</code>	<code>np</code>	<code>dp</code>	<code>dpwr</code>
<code>dw</code>	<code>dw</code>	<code>te</code>	<code>temp=te-273</code>
<code>fw</code>	<code>fb=1.1*sw/2</code>	<code>id</code>	<code>swl=1/id</code>
<code>ds</code>	<code>ss</code>	<code>sfo1</code>	<code>sfrq=sfo1+o1</code>
<code>sw</code>	<code>sw</code>	<code>sfo2</code>	<code>dfrq=sfo2+o2</code>
experiments done	<code>ni</code>	<code>p#</code>	<code>p#</code>
<code>o1</code>	<code>tof</code>	<code>d#</code>	<code>d#</code>
<code>o2</code>	<code>dof</code>	<code>s#</code>	<code>s#</code>
<code>rd</code> (or <code>d1</code> if <code>rd=0</code>)	<code>rd</code>	<code>ro</code>	<code>spin</code>
<code>pw</code> (or <code>p0</code> if <code>pw=0</code>)	<code>pw</code>	<code>rg</code>	<code>gain</code>
<code>p1</code>	<code>pw90</code>	<code>date</code>	<code>date</code>

<i>Bruker</i>	<i>Varian</i>	<i>Bruker</i>	<i>Varian</i>
de	de	time	time
ns	nt		

Bruker AMX parameters are converted to Varian parameters as shown in the table “AMX Parameter Conversion.” All Bruker parameters are converted under their original names if the name doesn't conflict with the name of a Varian parameter. Arrayed Bruker parameters like P and D are converted to the names P# and D#, where # is the index into the array.

Because `sread` is limited to 8-character parameter names, the parameters `rtwd1#` and `rtwd2#` are converted to `rtwd1#` and `rtwd2#`.

The parameter `proc` is set to 'ft' when the Bruker parameter `aq_mod` is 1, and `proc` is set to 'rft' when `aq_mod` is 2. `procl` is always set to `rft`, assuming TPPI in `t1`.

If there is a file named `info` in the directory with the Bruker data, it is read in and put into the text file for the converted data set.

AMX Parameter Conversion

<i>Bruker</i>	<i>Varian</i>	<i>Bruker</i>	<i>Varian</i>
ns (from <code>acqu</code>)	nt	te	<code>temp=te-273</code>
ns (from <code>acqus</code>)	ct	<code>sfo1</code>	<code>sfrq=sfo1</code>
td (from <code>acqus</code>)	np	<code>sfo2</code>	<code>dfrq=sfo2</code>
td (from <code>acqu2s</code>)	ni	<code>o1</code>	<code>tof</code>
<code>sw_h</code>	<code>sw</code>	<code>o2</code>	<code>dof</code>
<code>sw_h</code>	<code>dw=1.0e6/sw</code>	<code>ro</code>	<code>spin</code>
<code>sw_h</code> (from <code>acqu2s</code>)	<code>sw1</code>	<code>rg</code>	<code>gain</code>
<code>fw</code>	<code>fb=1.1*sw/2</code>	<code>date</code>	<code>date</code>
<code>ds</code>	<code>ss</code>	<code>date</code>	<code>time</code>
<code>rd</code> (or <code>d1</code> if <code>rd=0</code>)	<code>rd</code>	<code>nucleus</code>	<code>tn</code>
<code>de</code>	<code>de</code>	<code>decnuc</code>	<code>dn</code>
<code>pw</code> (or <code>p0</code> if <code>pw=0</code>)	<code>pw</code>	<code>pulprog</code>	<code>pslabel</code>
<code>p1</code>	<code>pw90</code>	<code>pulprog</code>	<code>seqfil</code>

Arguments: `file` is the input file name. For AMX data, `file` should be the name of the directory that contains the `acqus`, `acqu2s`, and `fid` or `ser` files. For AM data, `file` should be the name of the file containing the AM data. The `file` argument is not required to have a `.bru` extension, but if it does, the `.bru` extension is removed before creating the output file. Unless the `-cfile` option is present, the output file will have the same name as the input file, but with a `.cv` extension, and will be written into the current working directory.

options for AMX and AM data are the following, which can be entered in any order as long as `file` comes first (options are usually not necessary, but can be used to override the default actions of `convertbru`):

- `-bam` or `-bamx` specifies whether input is AM or AMX data. The default is determined from name of the input file given.
- `-cfile` specifies that the output file is given the name specified by `file` and is written with `.cv` appended to the name

- `-dxxx`, where `xxx` is the decoupler frequency (it must be a value between 10.0 and 640.0 MHz). The default is to read from data set.
- `-f` specifies that old output file is to be overwritten. The default is to not overwrite old files.
- `-olsb` or `-omsb` specifies whether the data has the least- or most-significant byte first. For AM data, the default is determined from data set. For AMX data, the default is `-olsb`.
- `-pxxx`, where `xxx` is the number of 24- or 32-bit words to skip before converting data. This option is for use with `-t` option to skip the header in AM data without converting it. Typical header sizes are 216 or 256 words. The default is 0.
- `-s3` or `-s4` specifies if AM data is 24-bit (3-byte) or 32-bit (4-byte). All AMX data is 32-bit. The default is determined from the data set.
- `-tall`, `-thdr`, or `-tdata` specifies whether `convertbru` should convert the header and the data, just the header, or just the data. The default is `-tall`.

Examples: Convert AM data from a UNIX shell (in all these examples, the file name is arbitrarily named `br_data`):

- `convertbru br_data` determines the file format and converts the header and data in the file `br_data`.
- `convertbru br_data -d250.0 -cout` determines the file format, converts the header and data in the `br_data`, sets the decoupler frequency to 250.0 MHz, and writes to an output file named `out.cv` in the current working directory.
- `convertbru br_data -thdr` determines file format and converts only the header in the file `br_data`.
- `convertbru br_data -tdata -p256 -s3 -omsb` converts only the data in `br_data` after skipping the 256-word header. The data is converted assuming it is 24-bit AM data words with the most-significant byte first.

Convert AM data from VNMR:

- `convertbru('br_data', '-tdata', '-p256', '-s3', '-omsb')` converts only the data in `br_data` after skipping the 256-word header. The data is converted assuming it is 24-bit AM data words with the most-significant byte first.

Convert AMX data from a UNIX shell:

- `convertbru br_data -f` converts `acqus` and `acqu2s` files to ASCII, if needed, and then converts data and overwrites the existing `br_data.cv` file.

Convert AMX data from VNMR:

- `convertbru('br_data', '-f')` converts `acqus` and `acqu2s` files to ASCII, if needed, and then converts data and overwrites the existing `br_data.cv` file.
- `convertbru('br_data', '-c/home/vnmr1/bdata/data1')` converts `acqus` and `acqu2s` files to ASCII, if needed, and then converts the data and writes it to `/home/vnmr1/bdata/data1.cv`.

See also: *Getting Started*

Related: [readbrutape](#) Read Bruker data files from 9-track tape (U)

`sread` Read converted data into VNMR (C)
`wft2da` Weight and Fourier transform phase-sensitive data (M)

copy Copy a file (C)

Syntax: `copy(<'-r',>from_file,to_file)`

Description: Makes a copy of a file using the UNIX `cp` command. All arguments are passed. `copy` operates the same as the VNMR `cp` command.

Arguments: `'-r'` is a keyword requesting a recursive copy (i.e., copy a directory).

`from_file` is the name of the file (or directory if `'-r'` used) to be copied.

`to_file` is the name of the copy of the file (or directory). If the `from_file` argument has an extension (e.g., `.fid`), be sure the `to_file` argument has the same extension.

Examples: `copy(' -r', '/home/vnmr1/vnmrsys/seqlib', '/vnmr/seqlib')`

`copy('/home/vnmr1/vnmrsys/seqlib/d2pul', \`
`'/vnmr/seqlib/d2pul')`

See also: *Getting Started*

Related: `cp` Copy a file (C)

cos Find cosine value of an angle (C)

Syntax: `cos(angle)<:n>`

Description: Finds the cosine of an angle.

Arguments: `angle` is the angle, given in radians.

`n` is the return value with the cosine of `angle`. The default is to display the cosine value in the status window.

Examples: `cos(.5)`
`cos(val):cos_val`

See also: *VNMR User Programming*

Related: `sin` Find sine value of an angle (C)

cosy Set up parameters to a COSY pulse sequence (M)

Syntax: `cosy`

Description: Sets up for a COSY (correlated spectroscopy) experiment.

Alternate: COSY button in the 2D Pulse Sequence Setup Menu.

See also: *User Guide: Liquids NMR*

Related: `cosyps` Set up parameters for phase-sensitive COSY pulse sequence (M)
`dqcosy` Set up parameters for double-quantum filtered COSY (M)
`relayh` Set up parameters for RELAYH pulse sequence (M)

cosy Change parameters for COSY experiment (M)

Syntax: `COSY<('GLIDE')>`

Description: Converts the current parameter set to a COSY experiment.

Arguments: `'GLIDE'` is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding proton spectrum for the experiment.

cosyps **Set up parameters for phase-sensitive COSY pulse sequence (M)**Syntax: `cosyps`

Description: Sets up a phase-sensitive COSY (homonuclear correlation) experiment.

Alternate: COSYPS button in the 2D Pulse Sequence Setup Menu.

See also: *User Guide: Liquids NMR*

Related: `cosy` Set up parameters for COSY pulse sequence (M)
`dqcosy` Set up parameters for double-quantum filtered COSY (M)
`relayh` Set up parameters for RELAYH pulse sequence (M)

cp **Copy a file (C)**Syntax: `cp(<'-r',>from_file,to_file)`Description: Makes a copy of a file using the UNIX `cp` command. All arguments are passed. `cp` operates the same as the VNMR `copy` command.

Arguments: `'-r'` is a keyword requesting a recursive copy (i.e., copy a directory).
`from_file` is the name of the file (or directory if `'-r'` used) to be copied.
`to_file` is the name of the copy of the file (or directory). If the `from_file` argument has an extension (e.g., `.fid`), be sure the `to_file` argument has the same extension.

Examples: `cp(' /home/vnmr1/vnmrsys/seqlib/d2pul ', \`
`' /vnmr/seqlib/d2pul ')`
`cp(' -r ', ' /home/vnmr1/vnmrsys/seqlib ', ' /vnmr/seqlib ')`

See also: *Getting Started*Related: `copy` Copy a file (C)**cp** **Cycle phase (P)**

Description: Sets the values that real-time variable `oph` is calculated as, either 0,1,2,3 (`cp='y'`) or 0 (`cp='n'`). The only circumstance where setting `cp='n'` may be useful is when displaying an FID with `acqi`. If there is an imbalance between the two receiver channels, the FID displayed for `acqi` may show alternating dc levels. The standard `gf` macro that prepares parameters for the FID display in `acqi` automatically handles this issue.

Values: `'y'` makes `oph` calculate as 0,1,2,3; this is the typical value.
`'n'` makes `oph` calculate as 0.

See also: *VNMR User Programming*

Related: `acqi` Interactive acquisition display process (C)
`go` Submit experiment to acquisition (C)
`gf` Prepare parameters for FID/spectrum display in `acqi` (M)

cpmgt2 **Set up parameters for CPMGT2 pulse sequence (M)**Syntax: `cpmgt2`Description: Macro to set up a CPMGT2 (Carr-Purcell Meiboom-Gill T_2) experiment.See also: *User Guide: Liquids NMR*Related: `t2` T_2 exponential analysis (M)

cpos_cvt Convert data set from a VXR-style system (M,U)

Syntax: (From UNIX) `cpos_cvt VXR_file`
 (From VNMR) `convert(VXR_file)`

Description: Converts data stored on a VXR-style system (Gemini, VXR, or XL) to the format used in VNMR software. `cpos_cvt` writes the converted data in a subdirectory of the current working directory, using the original name of the data set. The VNMR command `convert` loads the data from `VXR_file` into the current experiment and converts it to the new format.

Arguments: `VXR_file` is the file name in the VXR-style format to be converted to the VNMR style.

See also: *Getting Started*

Related: `convert` Convert data set from a VXR-style system (C,U)
`decomp` Decompose a VXR-style directory (C)
`rt` Retrieve FIDs (C)
`unix_vxr` Convert UNIX text files to VXR-style format (M,U)
`vxr_unix` Convert VXR-style text files to UNIX format (M,U)

cptmp Copy experiment data into experiment subfile (M)

Syntax: `cptmp<(file)>`

Description: Copies the data (parameters, FID, and transformed spectrum) from the current experiment into a subdirectory inside `curexp+ '/subexp'`.

Arguments: `file` is the name of the subfile to receive the data. The default is to take the name from the transmitter nucleus (if `seqfil='s2pul'`) or to use the pulse sequence name.

Examples: `cptmp`
`cptmp('cosy')`

See also: *Getting Started; User Guide: Liquids NMR*

Related: `curexp` Current experiment directory (P)
`rttmp` Retrieve experiment data from experiment subfile (M)
`seqfil` Pulse sequence name (P)
`svtmp` Move experiment data into experiment subfile (M)

cpx Create pbox shape file (M)

Syntax: `cpx<(ref_pw90,ref_pwr)>` or `cpx<('g')>`

Description: Calls UNIX command `Pbox`, which generates the specified pulse shape or decoupling/spin locking pattern, as defined by the `shapelib/Pbox.inp` file.

Arguments: `ref_pw90` is the reference 90° pulse width
`ref_pwr` is the reference power level.

'g' is a keyword that is required only when generating gradient shapes and if the file type is not specified otherwise.

Examples: `cpx`
`cpx('g')`
`cpx(pw90*compH, tpwr)`

See also: *User Guide: Liquids NMR*

Related: `Pbox` Pulse shaping software (U)

- cr** **Cursor position in directly detected dimension (P)**
- Description: Contains the current cursor position. The `r1` macro uses `cr` to set the reference line.
- See also: *Getting Started*
- Related: `centersw` Move cursor to center of spectrum (M)
`crf` Current time-domain cursor position (P)
`cr1` Clear ref. line in directly detected dimension (M)
`delta` Difference of two frequency cursors (P)
`r1` Set reference line in directly detected dimension (M)
- cr1** **Cursor position in 1st indirectly detected dimension (P)**
- Description: Contains the current cursor position along the first indirectly detected dimension. Analogous to the `cr` parameter except that `cr1` applies to the first indirectly detected dimension of a multidimensional data set. The `r11` macro uses `cr1` to set the reference line along this dimension.
- See also: *User Guide: Liquids NMR*
- Related: `centersw1` Move cursor to center of spectrum in 1st indirect dimension (M)
`cr` Cursor position in directly detected dimension (P)
`cr2` Cursor position in 2nd indirectly detected dimension (P)
`r11` Set ref. line in 1st indirectly detected dimension (M)
- cr2** **Cursor position in 2nd indirectly detected dimension (P)**
- Description: Contains the current cursor position along the second indirectly detected dimension. Analogous to the `cr` parameter except that `cr2` applies to the second indirectly detected dimension of a multidimensional data set. The `r12` macro uses `cr2` to set the reference line along this dimension.
- See also: *User Guide: Liquids NMR*
- Related: `centersw2` Move cursor to center of spectrum in 2nd indirect dimension (M)
`cr` Cursor position in directly detected dimension (P)
`cr1` Cursor position in 1st indirectly detected dimension (P)
`r12` Set ref. line in 2nd indirectly detected dimension (M)
- crcom** **Create user macro without using text editor (M)**
- Syntax: `crcom(file,actions)`
- Description: Creates a macro file in the user's macro library (`maclib`) with the contents given in the `actions` argument.
- Arguments: `file` is the file name of the user macro to be created. If a macro of the same name already exists, the user is asked whether or not to overwrite it.
- `actions` is a string containing the actions making up the user macro. The string cannot include a carriage return. If a single quote is needed within the string, it must be preceded by a backslash (see second example below).
- Examples: `crcom('plot','pl pscale pap page')`
`crcom('lds','load=\y\' su load=\n\')`
- See also: *VNMR User Programming*
- create** **Create new parameter in a parameter tree (C)**
- Syntax: `create(parameter<,type<,tree>>)`

Description: Creates a parameter in one of the parameter trees. A *parameter tree* is a UNIX file containing the attributes of parameters as formatted text. Refer to the command `paramvi` for a description of the file contents.

Arguments: `parameter` is the name of the parameter to be created.

`type` is the type of values in the parameter to be created and can be one of the following values (default is `'real'`):

- `'real'` is a value with no limits on range and can be positive or negative.
- `'string'` is a value composed of characters. Entry of strings can be limited to selected words by enumerating the possible values with the command `setenumerals`. For example, the enumerated values of `intmod` are `'off'`, `'partial'`, and `'full'`. Therefore, `intmod` can be set only to one of these three string values, such as `intmod='full'`.
- `'delay'` is a value from 0 to 8190, in unit of seconds.
- `'frequency'` is a positive real number value.
- `'flag'`, like `'string'`, is a value composed of characters. Entry of flags can be limited to selected characters by enumerating the possible values with the command `setenumerals`. For example, the enumerated values of `dmm` are `'c'`, `'f'`, `'g'`, `'m'`, `'p'`, `'r'`, `'u'`, `'w'`, and `'x'`. Therefore, `dmm` can only be set to a combinations of these nine characters, such as `dmm='ccw'`. If enumerated values are not set, the `'string'` and `'flag'` types are identical.
- `'pulse'` is a value from 0 to 8190, in units of μs .
- `'integer'` is a value composed of integers (0, 1, 2, 3, ...).

`tree` is one of the following types of parameter trees (default is `'current'`):

- `'current'` contains parameters that are adjusted to set up an experiment. The parameters are from the file `curpar` in the current experiment.
- `'global'` contains user-specific parameters from the file `global` in the `vnmr/sys` directory of the present UNIX user.
- `'processed'` contains parameters with which the data was obtained. These parameters are from the file `procpars` in the current experiment.
- `'systemglobal'` contains instrument-specific parameters from the text file `/vnmr/conpar`. Most of these parameters are defined using the `config` program. All users have the same `systemglobal` tree. Note that `conpar` is not written out when you exit; the only time `conpar` is ever modified is by the `config` program. Thus, any changes you make to `conpar` using `create` (or `destroy`, `setvalue`, etc.) are not permanent. To permanently create a parameter in `conpar`, you must use a text editor to change `/vnmr/conpar`.

Examples: `create('a')`
`create('b','string')`
`create('c','real','global')`

See also: *VNMR User Programming*

Related:	<code>destroy</code>	Destroy a parameter (C)
	<code>display</code>	Display parameters and their attributes (C)
	<code>fread</code>	Read parameters from file and load them into a tree (C)
	<code>fsave</code>	Save parameters from a tree to a file (C)
	<code>paramvi</code>	Edit a parameter and its attributes using <code>vi</code> text editor (M)
	<code>prune</code>	Prune extra parameters from current tree (C)
	<code>setenumerals</code>	Set values of a string variable in a tree (C)

`setgroup` Set group of a parameter in a tree (C)
`setprotect` Set protection mode of a parameter (C)

createtable Generate system gradient table (M)

Syntax: `createtable`

Description: Generates a gradient table in the `$vnmrsystem/imaging/gradtables` directory (`/vnmr/imaging/gradtables`) needed to run an imaging experiment. The system prompts the user for the boresize of the magnet, the maximum gradient strength (`gmax`), and the gradient rise time. The directory `/vnmr/imaging/gradtables` is set up to have group write permission mode for all VNMR users; however, the VNMR administrator, `vnmr1`, may want to set the write permission mode for `vnmr1` only.

Systems with three-axis pulse field gradients (PFGs) or microimaging gradients might not have the same gradient strength on each axis. If the gradient strength varies, `createtable` prompts for the maximum gradient strength for each axis (`gxmax`, `gymax`, and `gzmax`). Additionally, three-axis PFG amplifiers may be limited in their total current output, and hence the gradient strength, when gradients are simultaneously applied on all three axes. If this limitation exists, the user can enter the maximum combined gradient strength, which will be the combination of $x+y+z$, in gauss/cm.

The macro expects gradient strength entered in gauss/cm, risetime in μ s (it is converted to seconds when it is put in the table), and boresize in cm.

Gradient tables are needed when using the obliquing, phase encode, or magic-angle gradient PSG statements.

See also: *User Guide: Imaging*

Related: `gmax` Maximum gradient strength (P)
`gxmax, gymax, gzmax` Maximum gradient strengths for each axis (P)

crf Current time-domain cursor position (P)

Description: Contains current time-domain cursor position. To create `crf` and the other FID display parameters `axisf`, `dotflag`, `vpf`, `vpfi`, and `deltaf` (if the parameter set is older and lacks these parameters), enter `addpar('fid')`.

Values: Number, in seconds.

See also: *Getting Started*

Related: `addpar` Add selected parameters to the current experiment (M)
`cr11` Clear ref. line in 1st indirectly detected dimension (C)
`deltaf` Difference of two time cursors (P)
`fidpar` Add parameters for FID display in current experiment (M)

cr1 Clear reference line in directly detected dimension (M)

Syntax: `cr1`

Description: Clears frequency referencing along the directly detected dimension by setting the reference parameters `rfl` and `rfp` to zero. `cr1` also resets the referencing parameters `refpos` and `reffrq`.

See also: *Getting Started*

Related: `cr11` Clear ref. line in 1st indirectly detected dimension (C)
`cr12` Clear ref. line in 2nd indirectly detected dimension (C)
`r1` Set ref. line in directly detected dimension (M)

<code>reffrq</code>	Reference frequency of reference line (P)
<code>refpos</code>	Position of reference frequency (P)
<code>rfl</code>	Ref. peak position in directly detected dimension (P)
<code>rfp</code>	Ref. peak frequency in directly detected dimension (P)

`cr11` **Clear reference line in 1st indirectly detected dimension (M)**

Syntax: `cr11`

Description: Clears frequency referencing along the first indirectly detected dimension by setting the reference parameters `rfl1` and `rfp1` to zero. `cr11` also resets the referencing parameters `refpos1` and `reffrq1`.

See also: *User Guide: Liquids NMR*

Related:	<code>cr1</code>	Clear ref. line in directly detected dimension (C)
	<code>r11</code>	Set ref. line in 1st indirectly detected dimension (M)
	<code>reffrq1</code>	Ref. frequency of reference line in 1st indirect dimension (P)
	<code>refpos1</code>	Position of reference frequency in 1st indirect dimension (P)
	<code>rfl1</code>	Ref. peak position in 1st indirectly detected dimension (P)
	<code>rfp1</code>	Ref. peak frequency in 1st indirectly detected dimension (P)

`cr12` **Clear reference line in 2nd indirectly detected dimension (M)**

Syntax: `cr12`

Description: Clears frequency referencing along the second indirectly detected dimension by setting the reference parameters `rfl2` and `rfp2` to zero. `cr12` also resets the referencing parameters `refpos2` and `reffrq2`.

See also: *User Guide: Liquids NMR*

Related:	<code>cr1</code>	Clear ref. line in directly detected dimension (C)
	<code>r12</code>	Set ref. line in 2nd indirectly detected dimension (M)
	<code>reffrq2</code>	Ref. frequency of reference line in 2nd indirect dimension (P)
	<code>refpos2</code>	Position of reference frequency in 2nd indirect dimension (P)
	<code>rfl2</code>	Ref. peak position in 2nd indirectly detected dimension (P)
	<code>rfp2</code>	Ref. peak frequency in 2nd indirectly detected dimension (P)

`crmode` **Current state of the cursors in `df`, `ds`, or `dconi` programs (P)**

Description: Stores the current state (box mode or cursor mode) of cursors in the `df`, `ds`, or `dconi` interactive display programs. `crmode` is mostly used by programmable menus to determine the status of the cursors. It is stored in the file `vnmr/sys/global`.

Values: 'b' signifies the box mode, 'c' signifies the cursor mode.

See also: *VNMR User Programming*

Related:	<code>dconi</code>	Interactive 2D data display (C)
	<code>df</code>	Display a single FID (C)
	<code>ds</code>	Display a spectrum (C)

`crof2` **Recalculate `rof2` so that `lp = 0` (M)**

Syntax: `crof2<(alfa)>`

Description: Recalculates a new value for `rof2` (receiver gating time following a pulse) based upon the current `rof2` and `lp` (first-order phase) values, so that `lp` is rendered approximately 0. For `crof2` to work properly, a trial spectrum must be obtained and phased to pure absorption. This spectrum provides the current

`rof2` and `lp` values for `crof2`. The value of the `alfa` delay is left constant, provided `rof2` does not become less than 1 μ s.

`crof2` pertains to processing 2D data. Unless `lp` is approximately 0, `fpmult` affects both the dc offset and the curvature of the spectrum.

Arguments: `alfa` specifies a value for the `alfa` delay before acquisition.

See also: *User Guide: Liquids NMR*

Related: `alfa` Set `alfa` delay before acquisition (P)
`cfpmult` Calculate first point multiplier for 2D experiments (P)
`fpmult` First point multiplier for `np` FID data (P)
`lp` First-order phase along directly detected dimension (P)
`rof2` Receiver gating time following a pulse (P)

ct Completed transients (P)

Description: Stores a nonuser-enterable informational parameter that changes during the course of an experiment to reflect the number of completed transients. During most experiments, an accurate transient counter is displayed in the acquisition status window, updated every five seconds.

The value of `ct` is displayed in the acquisition parameter group by the `dg` command and is only updated when data processing occurs on the FID. In an experiment that is accumulating and not processed until the acquisition is complete, `ct` always indicates 0 until the end of the acquisition.

See also: *Getting Started*

Related: `dg` Display parameters of acquisition/processing group (C)

ctext Clear the text of the current experiment (C)

Syntax: `ctext`

Description: Clears the text from the current experiment text file (a block of text that may be used to describe the sample and experiment).

See also: *Getting Started*

Related: `atext` Append string to the current experiment text (M)
`text` Display text or set new text for current experiment (C)

ctune Tune carbon channel on $^1\text{H}/^{13}\text{C}$ GEMINI 2000 (M)

Applicability: *GEMINI 2000* $^1\text{H}/^{13}\text{C}$ systems.

Syntax: `ctune`

Description: Turns the ^{13}C transmitter on, directing about 0.5 watt of rf to the probe coil. Before entering `ctune`, be sure to move the proper cable on the back of the left-hand magnet leg to the BNC connector labeled TUNE, and also to move the proper cable leading to the probe to the BNC connector labeled TUNE. Enter `tuneoff` to turn off the transmitter. `ctune` cannot be executed while the console is acquiring or interactive acquisition (`acqi`) is connected. For the full tuning procedure, see the manual *Autoswitchable NMR Probes Installation*.

See also: *Getting Started; Autoswitchable NMR Probes Installation*.

Related: `acqi` Interactive acquisition display process (C)
`btune` Tune broadband channel on *MERCURY-Series*, *GEMINI 2000* (M)
`dtune` Tune lock channel on *GEMINI 2000* (M)
`htune` Tune proton channel on *GEMINI 2000* (M)
`sethw` Set values for hardware in acquisition system (C)

`su` Submit a setup experiment to acquisition (M)
`tuneoff` Turn off probe tuning mode, *MERCURY-Series*, *GEMINI 2000* (M)

curecc Name of eddy current compensation file (P)

Applicability: Systems with the imaging capabilities.

Description: A global string parameter containing the name of the file containing the last eddy current compensation file set. `eddysend` updates this parameter from ECC Tool window or from the keyboard.

See also: *User Guide: Imaging*

Related: `eccTool` Pop-up ECC Tool window (M)
`eddysend` Update acquisition eddy current settings (M)

curexp Current experiment directory (P)

Description: Contains the full UNIX path to the currently active experiment. This parameter is useful when accessing text files generated by various commands (e.g., `cat(curexp+' /fp.out ')`).

See also: *Getting Started*

Related: `systemdir` VNMR system directory (P)
`userdir` VNMR user directory (P)

curscan Scan currently in progress (P)

Applicability: Systems with LC-NMR accessory.

Description: Keeps track of which “scan” is currently in progress. If `curscan` does not exist, the `parlc` macro can create it.

See also: *User Guide: Liquids NMR*

Related: `nscans` Number of scout/real scan repetitions (P)
`parlc` Create LC-NMR parameters (M)

curwin Current window (P)

Description: An arrayed global parameter. The first value is the index of the selected window pane in the VNMR graphics window. The second value is the number of window pane rows. The third value is the number of columns.

See also: *Getting Started*

Related: `fontselect` Open FontSelect window (C)
`jwin` Activate current window (M)
`mapwin` List of experiment numbers (P)
`setgrid` Activate selected window (M)
`setwin` Activate selected window (C)

CustomQ Set up a custom queue in automation (M)

Syntax: `CustomQ(solvent)`

Description: Reads the EXPLIST in the `sampleinfo` file, calls appropriate setup macro (`AuHexp` or `AuCexp`), and sets up `explist` and `wexp` parameters appropriate for the selected chain. This macro is called by `auto_au`.

cutoff **Data truncation limit (P)**

Description: Defines the distance above and below the current vertical position `vp` at which spectra and integrals are truncated. By arraying `cutoff` to have two different values, the truncation limits above and below the current vertical position can be controlled independently (e.g., `cutoff=50` truncates data at `vp+50` mm and `vp-50` mm, and `cutoff=50,10` truncates data at `vp+50` mm and `vp-10` mm). `cutoff='n'` disables the action of `cutoff`.

`cutoff` is not active during interactive spectral displays (i.e., for the `ds` command), but is active during non-interactive spectral displays and plots (for the `dss` and `pl` commands).

Values: 'n', number in mm.

See also: *Getting Started*

Related:	<code>ds</code>	Display a spectrum (C)
	<code>dss</code>	Display stacked spectra (C)
	<code>pl</code>	Plot spectra (C)
	<code>vp</code>	Vertical position of spectrum (P)

cyclenoe **Set up parameters for CYCLENOE pulse sequence (M)**

Applicability: Systems in which the observe channel is equipped with direct synthesis rf and a linear amplifier. Sequence is supplied with *MERCURY* and *GEMINI 2000* as `noedif`.

Syntax: `cyclenoe`

Description: Sets up a difference NOE experiment.

See also: *User Guide: Liquids NMR*

Related: `noedif` Convert parameters for NOE difference experiment (M)

cy1br24 **Set up parameters for cycled BR24 pulse sequence (M)**

Applicability: Systems with solids module. Sequence is not supplied with *MERCURY* and *GEMINI 2000*.

Syntax: `cy1br24`

Description: Sets up a BR24 sequence with quadrature detection and prepulse for solids multiple-pulse line narrowing.

See also: *User Guide: Solid-State NMR*

Related: `br24` Set up parameters for BR24 pulse sequence (M)

cy1mrev **Set up parameters for cycled MREV8 pulse sequence (M)**

Applicability: Systems with a solids module. Sequence is not supplied with *MERCURY* and *GEMINI 2000*.

Syntax: `cy1mrev`

Description: Sets up a MREV8 sequence with quadrature detection and prepulse for solids multiple-pulse line narrowing.

See also: *User Guide: Solid-State NMR*

Related: `mrev8` Set up parameters for MREV8 pulse sequence (M)

cz **Clear integral reset points (C)**

Syntax: `cz<(frequency1,frequency2,...)>`

C

Description: Removes currently defined integral reset points.

Arguments: `frequency1`, `frequency2`, . . . are reset points corresponding to specified frequencies to be removed. The default is remove all reset points.

Examples: `cz`
`cz(800,600,250,60)`

See also: *Getting Started*

Related	<code>dli</code>	Display listed integral values (C)
	<code>dlni</code>	Display listed normalized integral values (C)
	<code>nli</code>	Find normalized integral values (C)
	<code>z</code>	Add integral reset point at the cursor position (C)

D

d0 Overhead delay between FIDs (P)

Applicability: ^{UNITY}*INOVA* systems

Description: Defines the extra overhead delay at the start of each FID or array element. Overhead times between increments and transients on the ^{UNITY}*INOVA* are deterministic, i.e., both known and constant. However, the time between increments (typically x) is longer than the time between transients (y , not including times that are actually part of the pulse sequence, such as **d1**). Some experiments may benefit if it is ensured that these two times are not only constant but equal. To ensure that the times are constant and equal, insert the time $d0$ at the start of each transient (before the pulse sequence actually starts); the actual delay is then $y+d0$. However, the overhead time may differ with different system configurations. To keep the $d0$ delay consistent across systems, set $d0$ greater than the overhead delay. The inter-FID delay x is then padded so that $y+d0=x+(d0-(x-y))$.

Currently, $d0$ only takes into account the extra delay at the start of each array element. It does not take into account the overhead delays at the start and end of each scan. It also does not take into account delays when arraying *status* statements, shims, or spinner speeds.

The $d0$ parameter does not exist in any parameter set and must be created by the user. To create $d0$, enter `create('d0','delay')`. If $d0$ is nonexistent, do not insert a delay between transients.

Values: 'n', 'y', or 0 to the maximum delay time (in seconds).

If $d0='n'$, the software calculates the overhead time for an array element and then delays that length of time at the beginning of subsequent transients for every array element. The calculated value of $d0$ can be viewed by entering $d0='y'$ in the VNMR input window.

If $d0$ is set to a value, that value is the length of delay time at the beginning of subsequent transients for every array element. If the value is greater than the array overhead time, the array overhead time is padded to $d0$.

See also: *VNMR User Programming*

Related: [create](#) Create new parameter in parameter tree (C)

d1 First delay (P)

Description: Length of the first delay in the standard two-pulse sequence and most other pulse sequences. This delay is used to allow recovery of magnetization back to equilibrium, if such a delay is desired.

Values: On *MERCURY-VX*, *MERCURY* systems: 0, 0.2 μ s to 150,000 sec. On *GEMINI 2000* systems: 0 to 4095 sec, smallest value possible is 0.2 μ s, finest increment possible is 0.1 μ s. On systems with a Data Acquisition Controller board: 0 to 8190 sec, smallest value possible is 0.1 μ s, finest increment possible is 12.5 ns. On systems with a Pulse Sequence Controller or Acquisition Controller board: 0 to 8190 sec, smallest value possible is 0.2 μ s, finest increment possible is 25 ns. On systems with an Output board: 0 to 8190 sec, smallest value possible is 0.2 μ s, finest increment possible is 0.1 μ s. (Refer to the `acquire` statement in the manual *VNMR User Programming* for a description of these boards.)

See also: *Getting Started*

Related:	alfa	Set alfa delay before acquisition (P)
	d2	Incremented delay in 1st indirectly detected dimension (P)
	d3	Incremented delay in 2nd indirectly detected dimension (P)
	d4	Incremented delay in 3rd indirectly detected dimension (P)
	pad	Preacquisition delay (P)

d2 Incremented delay in 1st indirectly detected dimension (P)

Description: Length of the second delay in the standard two-pulse sequence. The delay is controlled by the parameters **ni** and **sw1** in a 2D experiment.

Values: On *MERCURY-VX* and *MERCURY* systems: 0, 0.2 μ s to 150,000 sec. On *GEMINI 2000* systems: 0 to 4095 sec, smallest value possible is 0.2 μ s, finest increment possible is 0.1 μ s. On systems with a Data Acquisition Controller board: 0 to 8190 sec, smallest value possible is 0.1 μ s, finest increment possible is 12.5 ns. On systems with a Pulse Sequence Controller or Acquisition Controller board: 0 to 8190 sec, smallest value possible is 0.2 μ s, finest increment possible is 25 ns. On systems with an Output board: 0 to 8190 sec, smallest value possible is 0.2 μ s, finest increment possible is 0.1 μ s. (Refer to **acquire** statement in the manual *VNMR User Programming* for a description of these boards.)

See also: *Getting Started; User Guide: Liquids NMR*

Related:	d1	First delay (P)
	ni	Number of increments in 1st indirectly detected dimension (P)
	sw1	Spectral width in 1st indirectly detected dimension (P)

d2pul Set up parameters for D2PUL pulse sequence (M)

Applicability: This pulse sequence is made obsolete on the *GEMINI 2000* because when **tn** is 'H1' and **dn** is not 'H1', the software automatically uses the decoupler as the observe channel and the broadband channel as the decoupler channel. To run the equivalent of **d2pul**, set **tn**= 'H1' and **dn**= 'H1', and then run **s2pul**. **D2PUL** is also not on *MERCURY-VX* and *MERCURY* systems

Syntax: **d2pul**

Description: Sets up a standard two-pulse sequence using the decoupler as transmitter.

Alternate: **D2PUL** button in the 1D Pulse Sequence Setup Secondary Menu.

See also: *User Guide: Liquids NMR*

Related:	dhp	Decoupler high power with class C amplifier (P)
	dn	Nucleus for the first decoupler (P)
	dof	Frequency offset for first decoupler (P)
	dpwr	Power level for first decoupler with linear amplifiers (P)
	homo	Homodecoupling control for first decoupler (P)
	s2pul	Set up parameters for standard two-pulse sequence (M)
	tn	Nucleus for the observe transmitter (P)
	tof	Frequency offset for observe transmitter (P)
	tpwr	Power level of observe transmitter with linear amplifiers (P)

d3 Incremented delay for 2nd indirectly detected dimension (P)

Description: Length of a delay controlled by the parameters **ni2** and **sw2** in a 3D experiment. The **d2** delay, which is controlled by **ni** and **sw1**, is incremented through its entire implicit array first before **d3** is incremented. To create

parameters `d3`, `ni2`, `phase2`, and `sw2` to acquire a 3D data set in the current experiment, enter `addpar(' 3d ')`.

Values: On systems with a Data Acquisition Controller board: 0 to 8190 sec, smallest value possible is 0.1 μ s, finest increment possible is 12.5 ns. On systems with a Pulse Sequence Controller or Acquisition Controller board: 0 to 8190 sec, smallest value possible is 0.2 μ s, finest increment possible is 25 ns. On systems with an Output board: 0 to 8190 sec, smallest value possible is 0.2 μ s, finest increment possible is 0.1 μ s. (Refer to `acquire` statement in the manual *VNMR User Programming* for a description of these boards.)

See also: *User Guide: Liquids NMR*

Related:	<code>addpar</code>	Add selected parameters to the current experiment (M)
	<code>d1</code>	First delay (P)
	<code>ni2</code>	Number of increments in 2nd indirectly detected dimension (P)
	<code>par3d</code>	Create 3D acquisition, processing, display parameters (C)
	<code>phase2</code>	Phase selection for 3D acquisition (P)
	<code>sw2</code>	Spectral width in 2nd indirectly detected dimension (P)

d4 Incremented delay for 3rd indirectly detected dimension (P)

Description: Length of a delay controlled by the parameters `ni3` and `sw3` in a 4D experiment. The `d3` delay, which is controlled by `ni2` and `sw2`, is incremented through its entire implicit array first before `d4` is incremented. To create parameters `d4`, `ni3`, `phase3`, and `sw3` to acquire a 4D data set in the current experiment, enter `addpar(' 4d ')`.

Values: On systems with a Data Acquisition Controller board: 0 to 8190 sec, smallest value possible is 0.1 μ s, finest increment possible is 12.5 ns. On systems with a Pulse Sequence Controller or Acquisition Controller board: 0 to 8190 sec, smallest value possible is 0.2 μ s, finest increment possible is 25 ns. On systems with an Output board: 0 to 8190 sec, smallest value possible is 0.2 μ s, finest increment possible is 0.1 μ s. (Refer to `acquire` statement in the manual *VNMR User Programming* for a description of these boards.)

See also: *User Guide: Liquids NMR*

Related:	<code>addpar</code>	Add selected parameters to the current experiment (M)
	<code>d1</code>	First delay (P)
	<code>ni3</code>	Number of increments in 3rd indirectly detected dimension (P)
	<code>par4d</code>	Create 4D acquisition parameters (C)
	<code>phase3</code>	Phase selection for 4D acquisition (P)
	<code>sw3</code>	Spectral width in 3rd indirectly detected dimension (P)

DAC_to_G Store gradient calibration value in DOSY sequences (P)

Syntax: `DAC_to_G`

Description: `DAG_to_G` is automatically set by the `setup_dosy` macro by retrieving the gradient strength from the probe calibration file if `probe<>' '` and storing it in `DAC_to_G`. If `probe= ' '` (i.e., the probe is not defined), then `DAC_to_G` is set to the current value of the global parameter `gcal`.

See also: *User Guide: Liquids NMR*.

Related:	<code>dosy</code>	Process DOSY experiments (M)
	<code>setup_dosy</code>	Set up gradient levels for DOSY experiments (M)
	<code>setgcal</code>	Set the gradient calibration constant (M)

D

- da** **Display acquisition parameter arrays (C)**
- Syntax: `da<(par1<,par2><,par3...>>`
- Description: Displays arrayed acquisition parameters.
- Arguments: `par1, par2, par3, ...` are names of parameters to be displayed. The default is to display all such parameters.
- Examples: `da`
`da('d2')`
- See also: *User Guide: Liquids NMR*
- Related: `dg` Display parameters of acquisition/processing group (C)
-
- das1p** **Increment for t1 dependent first-order phase correction (P)**
- Applicability: UNITY *INOVA* systems.
- Description: Causes “shearing” of f_1 traces of a 2D dataset and is used to rotate the narrow projection of some solids correlations into the f_1 dimension. Several solids experiments for Dynamic Angle Spinning (DAS) and a triple-quantum filtered 2D MAS experiment require the use of `das1p`. (Note that the command `rotate` shears two traces and is inapplicable for these experiments.)
- When created, the value of `1p` for each increment of a 2D experiment is incremented by the value of `das1p` after the first Fourier transformation. The incremented phase correction is applied to the interferogram created from the coefficient table by `ft1d`, `ft2d`, `wft1d` and `wft2d`, when coefficients are present. `das1p` is also used with `ft1da`, `ft2da`, `wft1da` and `wft2da`.
- Values: Real values, typically similar in size to the value of parameter `1p`.
- See also: *User Guide: Liquids NMR; User Guide: Solid-State NMR*
- Related: `ft1d` Fourier transform along f_2 dimension (C)
`ft1da` Fourier transform phase-sensitive data (M)
`ft2d` Fourier transform 2D data (C)
`ft2da` Fourier transform phase-sensitive data (M)
`1p` First-order phase in directly detected dimension (P)
`rotate` Rotate 2D data (C)
`wft1d` Weight and Fourier transform f_2 for 2D data (C)
`wft1da` Weight and Fourier transform phase-sensitive data (M)
`wft2d` Weight and Fourier transform 2D data (C)
`wft2da` Weight and Fourier transform phase-sensitive data (M)
-
- date** **Date (P)**
- Description: An informational parameter taken from the UNIX-level calendar (which is set by the UNIX system operator only and cannot be entered by the user). Whenever data are acquired, the date is copied from UNIX and written into the acquisition parameters, thus maintaining a record of the date of acquisition.
- See also: *Getting Started*
-
- daxis** **Display horizontal LC axis (M)**
- Applicability: Systems with LC-NMR accessory.
- Syntax: `daxis(time,major_tic,minor_tic)`
- Description: Displays a horizontal LC axis. Horizontal axes are assumed to be used with “LC plots” of an entire LC run and are labeled accordingly.

Arguments: `time` is the time scale, in minutes (decimal values are fine), of the axis.
`major_tic` is spacing, in minutes (decimal values are fine), of major tics.
`minor_tic` is spacing, in minutes (decimal values are fine), of minor tics.

See also: *User Guide: Liquids NMR*

Related: `paxis` Display horizontal LC axis (M)

Dbppste Set up parameters for Dbppste pulse sequence (M)

Syntax: `Dbppste`

Description: Converts a parameter set to Dbppste experiment; replaces the macro `bppste`.

See also: *User Guide: Liquids NMR*

Related: `dosy` Process DOSY experiments (M)
`fiddle` Perform reference deconvolution (M)
`setup_dosy` Set up gradient levels for DOSY experiments (M)

Dbppsteinept Set up parameters for Dbppsteinept pulse sequence (M)

Syntax: `Dbppsteinept`

Description: Converts a parameter set to Dbppsteinept experiment.

See also: *User Guide: Liquids NMR*

Related: `dosy` Process DOSY experiments (M)
`fiddle` Perform reference deconvolution (M)
`setup_dosy` Set up gradient levels for DOSY experiments (M)

dc Calculate spectral drift correction (C)

Syntax: `dc`

Description: Turns on a linear baseline correction. The beginning and end of the straight line to be used for baseline correction are determined from the display parameters `sp` and `wp`. `dc` applies this correction to the spectrum and stores the definition of the straight line in the parameters `lv1` (level) and `tlt` (tilt). The correction is turned off by the command `cdc`.

Care must be taken to ensure that a resonance does not appear too close to either end of the spectrum, or `dc` can produce the opposite effect from that intended; namely, it induces a sloping baseline where none was present!

Alternate: DC button in the 1D Data Manipulation Menu.

See also: *Getting Started*

Related: `bc` 1D and 2D baseline correction (C)
`cdc` Cancel drift correction (C)
`dc` Drift correction group (P)
`lv1` Zero-order baseline correction (P)
`sp` Start of plot (P)
`tlt` First-order baseline correction (P)
`wp` Width of plot (P)

dc2d Apply drift correction to 2D spectra (C)

Syntax: `dc2d('f1' | 'f2')`

Description: Computes a drift correction and applies it to each individual trace.

Arguments: `'f1'` is a keyword to apply drift correction in the f_1 axis direction.

'f2' is a keyword to apply drift correction in the f_2 axis direction.

Examples: `dc2d('f1')`
`dc2d('f2')`

See also: *User Guide: Liquids NMR*

Related: `axis` Axis label for displays and plots (P)
`bc` 1D and 2D baseline correction (C)

dcg Drift correction group (P)

Description: Contains the results of the `dc` or `cdc` command. This parameter cannot be set in the usual way but it can be queried by entering `dcg?` to determine whether drift correction is active.

Values: 'dc' indicates drift correction is active.
'cdc' indicates drift correction is inactive.

See also: *User Guide: Liquids NMR*

Related: `cdc` Cancel drift correction (C)
`dc` Calculate spectral drift correction (C)

dcon Display noninteractive color intensity map (C)

Syntax: `dcon<(options)>`

Description: Produces a “contour plot,” actually a color intensity map, in the graphics window. The parameters `sp` and `wp`, `sp1` and `wp1`, and `sp2` and `wp2` control which portion of the spectrum is displayed. The parameters `sf` and `wf`, `sf1` and `wf1`, and `sf2` and `wf2` control which portion of time-domain data (FIDs and interferograms) is displayed. The parameter `trace` selects which dimension is displayed along the horizontal axis. The parameters `sc`, `wc`, `sc2`, and `wc2` control where on the screen the display occurs. The parameter `th` is active as a threshold to black out all contours whose intensity is below `th`. That is, if `th=7`, the colors 1 to 6 are not used for the display. The parameter `vs` controls the vertical scale of the spectrum.

`dcon` displays either absolute-value mode or phase-sensitive 2D data. In `av` mode, data are shown in 15 different colors (starting with black), with each color representing a factor of two in intensity (a single color is used on monochrome screens). In the `ph` mode, the normal display of colors ranges from -6 to $+6$, each representing a factor of two in intensity, with the color black representing intensity 0 in the center.

Arguments: `options` can be any of the following:

- 'linear' is a keyword to use linear instead of logarithmic increments.
- 'phcolor' is a keyword to use a phased color set with positive and negative peaks.
- 'avcolor' is a keyword to use an absolute-value color set with positive peaks. Negative contours only *cannot* be displayed, but if the data can be rephased, 180° added to `rp1`, and `dcon('avcolor')` entered again, the same thing is accomplished by inverting the phase of all peaks. Alternatively, `dpcon` can display negative peaks only.
- 'gray' is a keyword to use a gray scale color set.
- 'noaxis' is a keyword to omit the display outline and any horizontal or vertical axis.

- 'plot' causes the dcon display to be sent to the plotter instead of being drawn on the graphics window.

Examples: `dcon`
`dcon('gray')`
`dcon('linear','phcolor','plot')`

See also: *User Guide: Liquids NMR*

Related:	<code>dconi</code>	Interactive 2D data display (C)
	<code>dconi</code>	Control display selection for the dconi program (P)
	<code>dconn</code>	Display color intensity map without screen erase (C)
	<code>dpcon</code>	Display plotted contours (C)
	<code>image</code>	Display noninteractive gray scale image (M)
	<code>imageprint</code>	Plot noninteractive gray scale image (M)
	<code>sc</code>	Start of chart (P)
	<code>sc2</code>	Start of chart in second direction (P)
	<code>sf</code>	Start of FID (P)
	<code>sp</code>	Start of plot (P)
	<code>sp1</code>	Start of plot in 1st indirectly detected dimension (P)
	<code>sp2</code>	Start of plot in 2nd indirectly detected dimension (P)
	<code>th</code>	Threshold (P)
	<code>trace</code>	Mode for <i>n</i> -dimensional data display (P)
	<code>wc</code>	Width of chart (P)
	<code>wc2</code>	Width of chart in second direction (P)
	<code>wf</code>	Width of FID (P)
	<code>wp</code>	Width of plot (P)
	<code>wp1</code>	Width of plot in 1st indirectly detected dimension (P)
	<code>wp2</code>	Width of plot in 2nd indirectly detected dimension (P)

dconi **Interactive 2D data display (C)**

Syntax: `dconi<(options)>`

Description: Opens a 2D data display that can be interactively adjusted. The `dconi` program can accommodate any data set that can be displayed by `dcon`, `dpcon`, and `ds2d`, including 2D FIDs, interferograms, 2D spectra, planes from 3D data sets, and images. These data sets are generated by the commands `df2d`, `ft1d`, `ft2d`, and `ft3d`.

Arguments: `options` can be any of the following (note that the `dconi` parameter is also available to control the `dconi` program display):

- 'dcon' is a keyword to display a color intensity map; this is the default mode, but 'dcon' is provided for compatibility with certain macros. If 'dcon' is the first argument, it can be followed by any of the keywords 'linear', 'phcolor', 'avcolor', 'gray', and 'noaxis'; all of these keywords have the same meaning as when used with `dcon`.
- 'dpcon' is a keyword to display a true contour plot. If 'dpcon' is the first argument, it can be followed by any of the keywords 'pos', 'neg', and 'noaxis', and then followed by values for `levels` and `spacing`. All of these options have the same meaning as when used with `dpcon`.
- 'ds2d' is a keyword to display a stacked plot in whitewash mode (after the first spectra, each spectra is blanked out in regions in which it is behind an earlier spectra). If 'ds2d' is the first argument, it can be followed by any of the keywords 'nobase', 'fill', 'fillnb', and 'noaxis'. All of these keywords have the same meaning as used with `ds2d`.

- 'again' is a keyword to make `dcon` identify which display mode is currently being used and redraw the screen in that mode. This option is useful when writing VNMR menus.
- 'restart' is a keyword to activate `dcon` without redrawing the 2D data set. This action causes `dcon` to make sure that 2D data is already displayed.
- 'toggle' is a keyword to toggle between the cursor and box modes.
- 'trace' is a keyword to draw a trace above the spectrum.
- 'expand' is a keyword to toggle between the expand and full views of the spectrum.
- 'plot' is a keyword to plot a projection or a trace.
- 'hproj_max' is a keyword to do a horizontal projection of the maximum trace.
- 'hproj_sum' is a keyword to do a horizontal projection of the sum of all traces.
- 'vproj_max' is a keyword to do a vertical projection of the maximum trace.
- 'vproj_sum' is a keyword to do a vertical projection of the sum of all traces.

Examples: `dcon`
`dcon('dcon','gray','linear')`
`dcon('dpcon')`

See also: *User Guide: Liquids NMR*

Related:	<code>boxes</code>	Draw boxes selected by the mark command (C)
	<code>crmode</code>	Current state of cursors in <code>dfid</code> , <code>ds</code> , or <code>dcon</code> (P)
	<code>dcon</code>	Display noninteractive color intensity map (C)
	<code>dcon</code>	Control display selection for the <code>dcon</code> program (P)
	<code>dconn</code>	Display color intensity map without screen erase (C)
	<code>delta1</code>	Cursor difference in 1st indirectly detected dimension (P)
	<code>df2d</code>	Display FIDs of 2D experiment (C)
	<code>dpcon</code>	Display plotted contours (C)
	<code>ds2d</code>	Display 2D spectra in whitewash mode (C)
	<code>ft1d</code>	Fourier transform along f_2 dimension (C)
	<code>ft2d</code>	Fourier transform 2D data (C)
	<code>ft3d</code>	Perform a 3D Fourier transform on a 3D FID data set (M,U)
	<code>image</code>	Display noninteractive gray scale image (M)
	<code>imcon</code>	Display 2D data in interactive gray-scale mode (M)
	<code>is</code>	Integral scale (P)
	<code>l12d</code>	Automatic and interactive 2D peak picking (C)
	<code>proj</code>	Project 2D data (C)
	<code>sf</code>	Start of FID (P)
	<code>sp</code>	Start of plot (P)
	<code>sp1</code>	Start of plot in 1st indirectly detected dimension (P)
	<code>th</code>	Threshold (P)
	<code>vs2d</code>	Vertical scale for 2D displays (P)
	<code>vsadj</code>	Automatic vertical scale adjustment (M)
	<code>wf</code>	Width of FID (P)
	<code>wp</code>	Width of plot (P)
	<code>wp1</code>	Width of plot in 1st indirectly detected dimension (P)

dconi **Control display selection for the dconi program (P)**

Description: Controls the selection of the 2D display that follows entering the **dconi** command. Because **dconi** is implicitly executed by **ft2d**, the **dconi** parameter also controls the display that follows the **ft2d** or **wft2d** command.

dconi can be a string parameter in the “current” parameter set. Its syntax is similar to an argument string passed to the **dconi** program. For example, if `dconi = 'dpcon, pos, 12, 1.2'`, the **dconi** command displays twelve positive contours with **dpcon**, using a spacing of 1.2. The first component of the **dconi** string must be the name of the display program, such as **dcon**, **dconn**, **dpcon**, **dpconn**, **ds2d**, or **ds2dn**. Subsequent components of the string are arguments appropriate for that display program. Because the entire **dconi** parameter is a string, single quotes around words are not necessary and mixing words and numbers is not a problem, as the example above shows.

If the **dconi** parameter does not exist or is set to the null string (' '), the **dconi** program uses its normal default. If the **dconi** parameter is set to a string (e.g., `dconi= 'dcon, gray, linear'` for image display), and arguments are supplied to the **dconi** program, (e.g., `dconi('dpcon'`), the supplied arguments to the command take precedence. In the case of the examples above, a contour map, not an image, is displayed.

If the **dconi** parameter does not exist in the current experiment, it can be created by the commands `create('dconi', 'string'`)
`setgroup('dconi', 'display'`)

Values: ' ' (two single quotes) indicates that this parameter is ignored.

String `'display_program'` selects the named program for 2D displays.

String `'display_program, option1, option2'` selects the named program for 2D displays with options appropriate to the program.

Examples: `dconi= 'dpcon'` selects contour drawing rather than default color map
`dconi= 'dcon, gray, linear'` selects image display mode.

See also: *User Guide: Liquids NMR*; *User Guide: Imaging*

Related:	dcon	Display noninteractive color intensity map (C)
	dconi	Interactive 2D data display (C)
	dconn	Display color intensity map without screen erase (C)
	dpcon	Display plotted contours (C)
	dpconn	Display plotted contours without screen erase (C)
	ds2d	Display 2D spectra in whitewash mode (C)
	ds2dn	Display 2D spectra in whitewash mode without screen erase (C)
	ft2d	Fourier transform 2D data (C)
	imconi	Display 2D data in interactive gray-scale mode (M)
	wft2d	Weight and Fourier transform 2D data (C)

dconn **Display color intensity map without screen erase (C)**

Syntax: `dconn<(options)>`

Description: Produces a “contour plot,” actually a color intensity map, on the screen the same as the **dcon** command, but without erasing the screen before starting the plot. The options available are the same as the **dcon** command.

See also: *User Guide: Liquids NMR*

Related:	dcon	Display noninteractive color intensity map (C)
	dconi	Control display selection for the dconi program (P)

D

dcrmv Remove dc offsets from FIDs in special cases (P)

Description: If `dcrmv` exists and is set to 'y', hardware information is used to remove the dc offset from the FID providing `ct=1`. This only works on ^{UNITY}*INOVA*, *UNITYplus*, *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems with `sw` less than 100 kHz. If this feature is desired for a particular experiment, create `dcrmv` in that experiment by entering `create('dcrmv','string')` `setgroup('dcrmv','processing')` `dcrmv='y'`

To create image parameters `dcrmv`, `grayctr` and `graysl` in the current experiment, enter `addpar('image')`.

See also: *Getting Started; User Guide: Imaging*

Related:

<code>addpar</code>	Add selected parameters to the current experiment (M)
<code>create</code>	Create new parameter in a parameter tree (C)
<code>ct</code>	Completed transients (P)
<code>dc</code>	Calculate spectral drift correction (C)
<code>setgroup</code>	Set group of a variable in a tree (C)

ddf Display data file in current experiment (C)

Syntax: `ddf<(block_number,trace_number,first_number)>`

Description: Displays the file header of the data file in the current experiment. If entered with arguments, it also displays a block header and part of the data file of that block.

Arguments: `block_number` is the block number. Default is 1.

`trace_number` is the trace number within the block. Default is 1.

`first_number` is the first data element number within the trace. Default is 1.

See also: *VNMR User Programming*

Related:

<code>ddff</code>	Display FID file in current experiment (C)
<code>ddfp</code>	Display phase file in current experiment (C)

ddff Display FID file in current experiment (C)

Syntax: `ddff<(block_number,trace_number,first_number)>`

Description: Displays the file header of the FID file in the current experiment. If entered with arguments, it also displays a block header and part of the FID data of the block.

Arguments: `block_number` is the block number. Default is 1.

`trace_number` is the trace number within the block. Default is 1.

`first_number` is the first data element number within the trace. Default is 1.

See also: *VNMR User Programming*

Related:

<code>ddf</code>	Display data file in current experiment (C)
<code>ddfp</code>	Display phase file in current experiment (C)

ddfp Display phase file in current experiment (C)

Syntax: `ddfp<(block_number,trace_number,first_number)>`

Description: Displays the file header of the phase file in the current experiment. With arguments, it also display a block header and part of the phase file data of that block.

Arguments: `block_number` is the block number. Default is 1.

`trace_number` is the trace number within the block. Default is 1.

`first_number` is the first data element number within the trace. Default is 1.

See also: *VNMR User Programming*

Related: **ddf** Display data file in current experiment (C)
ddff Display FID file in current experiment (C)

ddif Synthesize and show DOSY plot (C)

Syntax: `ddif (<option>, lowerlimit, upperlimit)`

Description: Synthesizes a 2D spectrum from 1D spectra using the information produced by the **dosy** macro. **ddif** takes the 1D spectrum and a table of diffusion data stored in the file `diffusion_display.inp` in the current experiment and synthesizes a 2D DOSY spectrum. It is normally run by **dosy**, but can be directly run, for example, to recalculate a 2D DOSY spectrum with different digitization.

Arguments: `option` is either 'i' or 'c'.

'i' is for a display in which the 2D peak volume is proportional to 1D peak height.

'c' is for a display in which the 2D peak height equals the 1D.

`lowerlimit` is the lower diffusion limit (in units of 10^{-10} m²/s).

`upperlimit` is the upper diffusion limit (in units of 10^{-10} m²/s).

If arguments are not supplied, **ddif** defaults to showing the full range of diffusion coefficients in the file `diffusion_display.inp` in the current experiment. Make sure that the first increment of the DOSY data set has been transformed with the desired `fn2D` before using **ddif**. Digitization of the resultant spectrum is determined by `fn2D` in the spectral (F2) domain and `fn1` in the diffusion (F1) domain. Make sure that the product `fn2D*fn1` is not too large, or memory and processing time problems might result. Typical values are `fn2D=16384` (max: 64k) and `fn1=512`. After **dosy** or **ddif**, 1D data is overwritten by the 2D (the **dosy** macro keeps a copy of the 1D data, which can be retrieved with the command **undosy**). Similarly, after a DOSY spectrum has been calculated, it can be retrieved with the command **redosy**.

See also: *User Guide: Liquids NMR*

Related: **dosy** Process DOSY experiments (M)
fn2d Fourier number to build up 2D DOSY display in frequency domain (P)
redosy Restore the previous 2D DOSY display from the subexperiment (M)
undosy Restore original 1D NMR data from the subexperiment (M)

dds Default display (M)

Syntax: `dds`

Description: Looks for sequence-specific default display macro (`dds_seqfil`) and executes if one is found. If not, the **dds** macro displays 1D, 2D, or array spectrum as the case may be.

Related: **dds_seqfil** Sequence-specific default display (M)
dpl Default plot (M)
dpr Default process (M)

dds_seqfil Sequence-specific default display (M)

Syntax: `dds_seqfil`

Description: Sequence-specific default display. These macros are called by the **dds** macro.

Examples: `dds_NOESY1D`
`dds_TOCSY1D`

Related: `dds` Default display (M)
`dpl` Default plot (M)
`dpr` Default process (M)

debug Trace order of macro and command execution (C)

Syntax: `debug('c' | 'C')`

Description: Controls VNMR command and macro tracing. When turned on, `debug` displays a list of each command and macro in the shell tool from which VNMR was started. If VNMR is started when the user logs in, or if it was started from a drop-down menu or the CDE tool, the output goes to a Console window. If no Console window is present, the output goes into a file in the `/var/tmp` directory. This last option is not recommended. Nesting of the calls is indicated by indentation of the output. This feature is primarily a debugging tool for MAGICAL programming.

Arguments: `'c'` is a keyword to turn on command and macro tracing.
`'C'` is a keyword to turn off command and macro tracing.

Examples: `debug('c')`
`debug('C')`

See also: *VNMR User Programming*

decfrq Interrogate or set first decoupler frequency (obsolete)

Description: This command is no longer in VNMR. Use `dfrq` as the effective replacement.

Related: `dfrq` Transmitter frequency of first decoupler (P)

dec2frq Interrogate or set second decoupler frequency (obsolete)

Description: This command is no longer in VNMR. Use `dfrq2` as the effective replacement.

Related: `dfrq2` Transmitter frequency of second decoupler (P)

dec3frq Interrogate or set third decoupler frequency (obsolete)

Description: This command is no longer in VNMR. Use `dfrq3` as the effective replacement.

Related: `dfrq3` Transmitter frequency of third decoupler (P)

decomp Decompose a VXR-style directory (M)

Syntax: `decomp<(VXR_file)>`

Description: Takes a library, as loaded from a VXR-style system (VXR, XL, or Gemini), and extracts each entry into a separate UNIX file. The file can be obtained from a magnetic tape or over limNET. `decomp` creates a UNIX subdirectory in the current working directory and uses that to write each entry as a UNIX file. The name of the UNIX subdirectory is derived from the library name.

Arguments: `VXR_file` is the name of the original file. It must have an extension in the form `.NNN`, where `NNN` is the number of entries in the original library. A limit of 432 entries is imposed.

See also: *Getting Started*

Related: `convert` Convert data set from a VXR-style system (C,U)
`unix_vxr` Convert UNIX text files to VXR-style format (M,U)
`vxr_unix` Convert VXR-style text files to UNIX format (M,U)

def_osfilt Default value of osfilt parameter (P)

Description: A global parameter that establishes the default type of digital filter, AnalogPlus™ or brickwall, when DSP is configured. The *actual* filter used in any experiment is set by the local parameter `osfilt`. Usually, `def_osfilt` is set to the value for normal use, and then `osfilt` is changed within a given experiment if different filter characteristics are desired.

Values: 'a' or 'A' for the AnalogPlus digital filter. This filter is flatter in the passband and drops off somewhat more sharply than analog filters.

'b' or 'B' for the brickwall digital filter. This filter is extremely flat across the passband and drops off sharply on the edge; however, the enhanced filtering comes at the expense of somewhat reduced baseline performance.

See also: *Getting Started*

Related: `dsp` Type of DSP for data acquisition (P)
`osfilt` Oversampling filter for real-time DSP (P)

defaultdir Default directory for Files menu system (P)

Description: Stores the name to the default directory for use with the Directory Menu in the Files menu system. Initial value for `defaultdir` is the home or login directory of the user. Selecting the Default button in the Directory Menu sets the current directory to the value of `defaultdir`. The opposite action, setting the value of `defaultdir` to the current directory, occurs when the Set Default button in the Directory Menu is selected. If the entry for a directory is marked and the Set Default button is selected, the directory marked becomes the new value of `defaultdir`.

See also: *Getting Started*

delcom Delete a user macro (M)

Syntax: `delcom(file)`

Description: Deletes a macro file in a user's macro library (`maclib`). Note that `delcom` will not delete a macro in the VNMR system macro library or a macro in a macro directory specified by the `maclibpath` parameter.

Arguments: `file` is the file name of the user's macro to be deleted.

Examples: `delcom('lds')`

See also: *VNMR User Programming*

Related: `crcom` Create user macro without using a text editor (C)
`maclibpath` Path to user's macro directory (P)
`macrorm` Remove a user macro (C)

delete Delete a file, parameter directory, or FID directory (C)

Syntax: `delete(file1<,file2,...>)`

Description: Delete files and directories in a somewhat safer manner than the `rm` command. Using `rm` is not recommended in VNMR because `rm` allows wildcard

characters (* and ?) in the file description and recursive file deletion with the `-r` option. The `delete` command does not allow wildcard characters or the `-r` option, but you can still use the `delete` command to delete a file as well as remove `.fid` and `.par` directories, normally the only directories that need to be removed (experiment directories are deleted with the `delexp` macro).

Arguments: `file1`, `file2`, ... are the names of one or more files or directories to be deleted. When the `delete` command is entered, it first searches for `file1`. If it finds that file and it is not a directory, `file1` is deleted. If `file1` is not found, `.fid` is appended to the file name and `delete` searches for the file in that `.fid` directory. If the file is found, it is removed; otherwise, `.par` is appended to the file name and `delete` searches for the file in that `.par` directory. If the file is found, it is removed; otherwise, the command takes no action and continues to the next file name. The process is repeated for each file name given as an argument.

Examples: `delete('/home/vnmr1/memo')`
`delete('/vnmr/fidlib/fid1d')`

See also: *Getting Started*

Related: `delexp` Delete an experiment (M)
`rm` Delete file (C)
`rmdir` Remove directory (C)

delexp Delete an experiment (M)

Syntax: `delexp(experiment_number)`

Description: Deletes an experiment.

Arguments: `experiment_number` is the number (from 2 through 9999) of the experiment to be deleted (experiment 1 cannot be deleted). `delexp` also deletes the corresponding `jexpXXX` macro if necessary.

Examples: `delexp(321)`

Alternate: Delete button in the Workspace Menu.

See also: *User Guide: Liquids NMR*

Related: `cexp` Create an experiment (M)
`jexp` Join existing experiment (C)

dels Delete spectra from T_1 or T_2 analysis (C)

Syntax: `dels(index1<, index2, ...>)`

Description: Deletes the spectra selected from the file `fp.out` (the output file of `fp`) used by the `t1` or `t2` analysis. Spectra may be restored by rerunning `fp`.

Arguments: `index1`, `index2`, ... are the indexes of the spectra to be deleted.

Examples: `dels(7)`
`dels(2, 5)`

See also: *User Guide: Liquids NMR*

Related: `d11` Display listed line frequencies and intensities (C)
`fp` Find peak heights or phases (C)
`get11` Get frequency and intensity of a line (C)
`t1` T_1 exponential analysis (M)
`t2` T_2 exponential analysis (M)

- delta** **Cursor difference in directly detected dimension (P)**
- Description: Difference between two frequency cursors along the directly detected dimension. The value is changed by moving the right cursor, relative to the left, in the `ds` or `dconi` display.
- Values: Positive number, in Hz.
- See also: *Getting Started*
- Related: `dconi` Interactive 2D data display (C)
`delta1` Cursor difference in 1st indirectly detected dimension (P)
`delta2` Cursor difference in 2nd indirectly detected dimension (P)
`ds` Display a spectrum (C)
`split` Split difference between two cursors (M)
- delta1** **Cursor difference in 1st indirectly detected dimension (P)**
- Description: Difference of two frequency cursors along the first indirectly detected dimension. Analogous to the `delta` parameter except that `delta1` applies to the first indirectly detected dimension of a multidimensional data set.
- Values: Positive number, in Hz.
- See also: *User Guide: Liquids NMR*
- Related: `delta` Cursor difference in directly detected dimension (P)
- delta2** **Cursor difference in 2nd indirectly detected dimension (P)**
- Description: Difference of two frequency cursors along the second indirectly detected dimension. Analogous to the `delta` parameter except that `delta2` applies to the second indirectly detected dimension of a multidimensional data set.
- Values: Positive number, in Hz.
- See also: *User Guide: Liquids NMR*
- Related: `delta` Cursor difference in directly detected dimension (P)
- deltaf** **Difference of two time-domain cursors (P)**
- Description: Difference between the two time-domain cursors of the `df` (or `dfid`) display. To create this parameter and the other FID display parameters `axisf`, `dotflag`, `vpf`, `vpfi`, and `crf` (if the parameter set is older and lacks these parameters), enter `addpar('fid')`.
- Values: Number, in seconds.
- See also: *Getting Started*
- Related: `addpar` Add selected parameters to the current experiment (M)
`crf` Current time-domain cursor position (P)
`df` Display a single FID (C)
`dfid` Display a single FID (C)
- dept** **Set up parameters for DEPT pulse sequence (M)**
- Syntax: `dept`
- Description: Macro for the DEPT (Distortionless Enhancement by Polarization Transfer) experiment.
- Alternate: DEPT button in the 1D Pulse Sequence Setup Menu.

See also: *User Guide: Liquids NMR*

Related: **adept** Automatic DEPT analysis and spectrum editing (C)
autodept Automated complete analysis of DEPT data (M)
deptgl Set up parameters for DEPTGL pulse sequence (M)
deptproc Process array of DEPT spectra (M)
padept Plot automatic DEPT analysis (C)
ppcal Proton decoupler pulse calibration (M)

DEPT **Change parameters for DEPT experiment (M)**

Syntax: `DEPT<('GLIDE')>`

Description: Converts the current parameter set to a DEPT experiment.

Arguments: 'GLIDE' is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding carbon spectrum for the experiment.

deptgl **Set up parameters for DEPTGL pulse sequence (M)**

Applicability: Sequence is not supplied with *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Syntax: `deptgl`

Description: Macro for the DEPTGL pulse sequence for spectral editing and polarization transfer experiments.

See also: *User Guide: Liquids NMR*

Related: **dept** Set up parameters for DEPT pulse sequence (M)

deptproc **Process array of DEPT spectra (M)**

Syntax: `deptproc`

Description: Automatically processes arrays of DEPT-type spectra. The FIDs are transformed (using `lb=2.5`), phased, and scaled. In foreground operation, a stacked display is produced. By default, an automatic DEPT analysis (**adept**) is performed.

See also: *User Guide: Liquids NMR*

Related: **adept** Automatically edit DEPT spectra (C)
dept Set up parameters for DEPT pulse sequence (M)
lb Line broadening along the directly detected dimension (P)
pldept Plot DEPT type spectra (M)
procplot Automatically process FIDs (M)

destroy **Destroy a parameter (C)**

Syntax: `destroy(parameter<,tree>)`

Description: Removes a parameter from one of the parameter trees. If the destroyed parameter was an array, the **array** parameter is automatically updated.

Arguments: `parameter` is the name of the parameter to be destroyed.

`tree` is a keyword for the type of parameter tree: 'global', 'current', 'processed', or 'systemglobal'. The default is 'current'. Refer to the **create** command for more information on types of trees.

Examples: `destroy('a')`
`destroy('c','global')`

See also: *VNMR User Programming*

Related:	array	Parameter order and precedence (P)
	create	Create new parameter in a parameter tree (C)
	display	Display parameters and their attributes (C)
	paramvi	Edit a variable and its attributes using <code>vi</code> text editor (C)
	prune	Prune extra parameters from current tree (C)

destroygroup Destroy parameters of a group in a tree (C)

Syntax: `destroygroup (group<, tree>)`

Description: Removes parameters of a group from one of the parameters trees.

Arguments: `group` is a keyword for the type of parameter group: 'all', 'sample', 'acquisition', 'processing', 'display', or 'spin'.
`tree` is a keyword for the type of parameter tree: 'global', 'current', or 'processed'. The default is 'current'. Refer to the **create** command for more information on trees.

Examples: `destroygroup('sample')`
`destroygroup('all', 'global')`

See also: *VNMR User Programming*

Related:	create	Create new parameter in a parameter tree (C)
	destroy	Destroy a parameter (C)
	display	Display parameters and their attributes (C)
	groupcopy	Copy parameters of group from one tree to another (C)
	setgroup	Set group of a variable in a tree (C)

df Display a single FID (C)

Syntax: (1) `df<(index)>`
 (2) `df(options)`

Description: Displays a single FID. Parameter entry after an FID has been displayed causes the display to be updated. The FID is left-shifted by the number of complex data points specified by the parameter `lsfid`. The FID is also phase-rotated (zero-order only) by the number of degrees specified by the parameter `phfid`. Left shifting and phasing can be avoided by setting `lsfid` and `phfid` to 'n'. `df` is identical in function to the `dfid` command.

Arguments: `index` (used with syntax 1) is the number of a particular FID for arrayed 1D experiments or for 2D experiments. Default is 1.

`options` (used with syntax 2) is any of the following:

- 'toggle' is a keyword to switch between box and cursor modes.
- 'restart' is a keyword to redraw the cursor if it has been turned off.
- 'expand' is a keyword to switch between expanded and full views of the FID.
- 'imaginary' is a keyword to switch on and off the display of the imaginary FID.
- 'sfwf' is a keyword to interactively adjust the start and width of the FID display.
- 'phase' is a keyword to enter an interactive phasing mode.
- 'dscale' is a keyword to toggle the scale below the FID on and off.

D

Examples: `df`
`df(4)`
`df('restart')`

Alternate: Display FID button in the 1D Data Processing Menu.

See also: *Getting Started*

Related: `crmode` Current state of cursors in `dfid`, `ds`, or `dconi` (P)
`dfid` Display a single FID (C)
`df2d` Display FIDs of 2D experiment (C)
`dfmode` Current state of display of imaginary part of a FID (P)
`lsfid` Number of complex points to left-shift the `np` FID (P)
`phfid` Zero-order phasing constant for the `np` FID (P)

df2d Display FIDs of 2D experiment (C)

Syntax: `df2d(<'nf', ><array_index> >`

Description: Produces a color intensity map of the raw 2D FIDs as a function of t_1 and t_2 . The display can be modified by subsequent display commands, for example, `df2d dconn` will display the 2D FIDs without clearing the graphics screen.

Arguments: 'nf' is a keyword specifying that the data has been collected in the compressed form using `nf`. In other words, each array element is collected as one 2D FID or image comprised of `nf` FIDs or traces.

`array_index` is the index of the array to be displayed.

Examples: `df2d`
`df2d(1)`

See also: *User Guide: Liquids NMR*

Related: `dconi` Interactive 2D data display (C)
`df` Display a single FID (C)

df2dn Display FIDs of 2D experiment without screen erase (obsolete)

Description: The `df2dn` command is no longer used. Entering `df2d` followed by `dconn` is functionally the same as `df2dn`.

Related: `dconn` Display color intensity map without screen erase (C)
`df2d` Display FIDs of 2D experiment (C)

dfid Display a single FID (C)

Syntax: (1) `dfid<(index)>`
(2) `dfid<(options)>`

Description: Functions the same as the `df` command. See `df` for information.

Alternate: Display FID button in the 1D Data Processing Menu.

See also: *Getting Started*

Related: `df` Display a single FID (C)

dfmode Current state of display of imaginary part of a FID (P)

Description: Holds a string variable that reflects the state of display of the imaginary part of a FID. `dfmode` is primarily used by the programmable menu `dfid` to determine the status of the display of the imaginary part of a FID.

Values: 'r' indicates the current display is real only.

' i ' indicates the current display is imaginary.

' z ' indicates the display is zero imaginary.

See also: *VNMR User Programming*

dfrq Transmitter frequency of first decoupler (P)

Description: Contains the transmitter frequency for the first decoupler. `dfrq` is automatically set when the parameter `dn` is changed and should not be necessary for the user to manually set.

Values: Frequency, in MHz. On *GEMINI 2000* systems, the offset range is ± 50 kHz. On other systems, the value is limited by synthesizer used with the channel.

See also: *Getting Started*

Related:	<code>dfrq2</code>	Transmitter frequency of second decoupler (P)
	<code>dfrq3</code>	Transmitter frequency of third decoupler (P)
	<code>dfrq4</code>	Transmitter frequency of fourth decoupler (P)
	<code>dn</code>	Nucleus for first decoupler (P)
	<code>dof</code>	Frequency offset for first decoupler (P)
	<code>sfrq</code>	Transmitter frequency of observe nucleus (P)
	<code>spcfrq</code>	Display frequencies of rf channels (M)

dfrq2 Transmitter frequency of second decoupler (P)

Applicability: Systems with a second decoupler.

Description: Contains the transmitter frequency for the second decoupler. `dfrq2` is automatically set when parameter `dn2` is changed and should not be necessary for the user to manually set.

Values: Frequency, in MHz. Value is limited by synthesizer used with the channel. If `dn2= ' '` (two single quotes with no space in between) and a second decoupler is present in the console, `dfrq2` is internally set to 1 MHz.

See also: *Getting Started*

Related:	<code>dn2</code>	Nucleus for second decoupler (P)
	<code>dof2</code>	Frequency offset for second decoupler (P)

dfrq3 Transmitter frequency of third decoupler (P)

Applicability: Systems with a third decoupler.

Description: Contains the transmitter frequency for the third decoupler. `dfrq3` is automatically set when the parameter `dn3` is changed and should not be necessary for the user to manually set.

Values: Frequency, in MHz. Value is limited by synthesizer used with the channel. If `dn3= ' '` (two single quotes with no space in between) and a third decoupler is present in the console, `dfrq3` is internally set to 1 MHz.

See also: *Getting Started*

Related:	<code>dn3</code>	Nucleus for third decoupler (P)
	<code>dof3</code>	Frequency offset for third decoupler (P)

dfrq4 Transmitter frequency of fourth decoupler (P)

Applicability: Systems with a deuterium decoupler channel as the fourth decoupler.

Description: Contains the transmitter frequency for the fourth decoupler. `dfrc4` is automatically set when the parameter `dn4` is changed and should not be necessary for the user to manually set.

Values: Frequency, in MHz. Value is limited by a synthesizer used with the channel. If `dn4= ' '` (two single quotes with no space in between) and a fourth decoupler is present in the console, `dfrc4` is internally set to 1 MHz.

See also: *Getting Started*

Related: `dn4` Nucleus for fourth decoupler (P)
`dof4` Frequency offset for fourth decoupler (P)
`spcfrq` Display frequencies of rf channels (M)
`rftype` type of rf generation

dfs Display stacked FIDs (C)

Syntax: `dfs(<start><,finish><,step><,'all'|'imag'><,color>)>`

Description: Displays one or more FIDs. The position of the first FIDs is governed by the parameters `wc`, `sc`, and `vpf`. A subsequent FID is positioned relative to the preceding FID by the parameters `vo` and `ho`.

Arguments: `start` is the index number of the first FID for multiple FIDs. It can also be the index number of a particular FID for arrayed 1D or 2D data sets.

`finish` is the index number of the last FID for multiple FIDs. To include all FIDs, set `start` to 1 and `finish` to `arraydim` (see example below).

`step` is the increment for the FID index. The default is 1.

'all' is a keyword to display all of the FIDs. This is the default.

'imag' is a keyword to display only the imaginary FID channel.

`color` is the color of the display: 'red', 'green', 'blue', 'cyan', 'magenta', 'yellow', 'black', or 'white'.

Examples: `dfs(1,arraydim,3)`
`dfs('imag')`

See also: *Getting Started*

Related: `arraydim` Dimension of experiment (P)
`dfsa` Display stacked FIDs automatically (C)
`dfsan` Display stacked FIDs automatically without screen erase (C)
`dfsh` Display stacked FIDs horizontally (C)
`dfshn` Display stacked FIDs horizontally without screen erase (C)
`dfsn` Display stacked FIDs without screen erase (C)
`dfww` Display FIDs in whitewash mode (C)
`ho` Horizontal offset (P)
`plfid` Plot FID (C)
`pfww` Plot FIDs in whitewash mode (C)
`sc` Start of chart (P)
`vo` Vertical offset (P)
`vpf` Current vertical position of FID (P)
`wc` Width of chart (P)

dfsa Display stacked FIDs automatically (C)

Syntax: `dfsa(<start><,finish><,step><,'all'|'imag'><,color>)>`

Description: Displays one or more FIDs automatically by adjusting the parameters `vo` and `ho` to fill the screen in a lower left to upper right presentation (`wc` must be set

to less than full screen width for this to work). The position of the first FID is governed by parameters **wc**, **sc**, and **vpf**.

Arguments: **start** is the index number of the first FID for multiple FIDs. It can also be the index number of a particular FID for arrayed 1D or 2D data sets.

finish is the index number of the last FID for multiple FIDs.

step is the increment for the FID index. The default is 1.

'**all**' is a keyword to display all of the FIDs. This is the default.

'**imag**' is a keyword to display only the imaginary FID channel.

color is the color of the display: 'red', 'green', 'blue', 'cyan', 'magenta', 'yellow', 'black', or 'white'.

See also: *Getting Started*

Related: **dfs** Display stacked FIDs (C)
dfsan Display stacked FIDs automatically without screen erase (C)

dfsan Display stacked FIDs automatically without screen erase (C)

Syntax: `dfsan(<start><,finish><,step><,'all'|'imag'><,color>)>`

Description: Functions the same as the command **dfsa** except the graphics window is not erased before starting the display. This allows composite displays of many FIDs to be created. The arguments are the same as **dfsa**.

See also: *Getting Started*

Related: **dfsa** Display stacked FIDs automatically (C)

dfsh Display stacked FIDs horizontally (C)

Syntax: `dfsh(<start><,finish><,step><,'all'|'imag'><,color>)>`

Description: Displays one or more FIDs horizontally by setting **vo** to zero and adjusting **ho**, **sc**, and **wc** to fill the screen from left to right with the entire array. The position of the first FID is governed by parameters **wc**, **sc**, and **vpf**.

Arguments: **start** is the index number of the first FID for multiple FIDs. It can also be the index number of a particular FID for arrayed 1D or 2D data sets.

finish is the index number of the last FID for multiple FIDs. To display all FIDs, set **finish** to the parameter **arraydim**.

step is the increment for the FID index. The default is 1.

'**all**' is a keyword to display all of the FIDs. This is the default.

'**imag**' is a keyword to display only the imaginary FID channel.

color is the color of the display: 'red', 'green', 'blue', 'cyan', 'magenta', 'yellow', 'black', or 'white'.

See also: *Getting Started*

Related: **dfs** Display stacked FIDs (C)
dfshn Display stacked FIDs horizontally without screen erase (C)

dfshn Display stacked FIDs horizontally without screen erase (C)

Syntax: `dfshn(<start><,finish><,step><,'all'|'imag'><,color>)>`

Description: Functions the same as the command **dfsh** except the graphics window is not erased before starting the display. This allows composite displays of many FIDs to be created. The arguments are the same as **dfsh**.

See also: *Getting Started*

Related: **dfsh** Display stacked FIDs horizontally (C)

dfsn Display stacked FIDs without screen erase (C)

Syntax: `dfsn(<start><,finish><,step><,'all'|'imag'><,color>)>`

Description: Functions the same as the command **dfs** except the graphics window is not erased before starting the display. This allows composite displays of many FIDs to be created. The arguments are the same as **dfs**.

See also: *Getting Started*

Related: **dfs** Display stacked FIDs (C)

dfww Display FIDs in whitewash mode (C)

Syntax: `dfww(<start><,finish><,step><,'all'|'imag'><,color>)>`

Description: Displays FIDs in whitewash mode (after the first FID, each FID is blanked out in regions in which it is behind an earlier FID). The position of the first FIDs is governed by parameters **wc**, **sc**, and **vpf**.

Arguments: **start** is the index number of the first FID for multiple FIDs. It can also be the index number of a particular FID for arrayed 1D or 2D data sets.

finish is the index number of the last FID for multiple FIDs.

step is the increment for the FID index. The default is 1.

'all' is a keyword to display all of the FIDs. This is the default.

'imag' is a keyword to display only the imaginary FID channel.

color is the color of the display: 'red', 'green', 'blue', 'cyan', 'magenta', 'yellow', 'black', or 'white'.

See also: *Getting Started*

Related: **dfs** Display stacked FIDs (C)
pfww Plot FIDs in whitewash mode (C)

dg Display group of acquisition/processing parameters (C)

Syntax: `dg(template)>`

Description: Displays the group of acquisition and 1D/2D processing parameters. To display an individual parameter, enter the name of the parameter followed by a question mark (e.g., **sw?**). Parameters do not have to be displayed in order to be entered or changed. The **dg** display is controlled by the string parameter **dg**.

Arguments: **template** is the name of the template parameter. The default is 'dg'. See the manual *VNMR User Programming* for rules on constructing a template.

Commands such as **dg1**, **dg2**, and **dgs** (but not **da**) are macros that activate **dg** with the appropriate **template** argument ('dg1', 'dg2', 'dgs', etc.).

Examples: `dg`
`dg('dgexp')`

See also: *Getting Started; VNMR User Programming*

Related: **?** Display individual parameter value (C)
da Display acquisition parameter arrays (C)
dg Control dg parameter group display (P)
dg1 Display group of display parameters (M)
dg2 Display group of 3rd and 4th rf channel/3D parameters (M)

<code>dg1p</code>	Display group of linear prediction parameters (M)
<code>dgs</code>	Display group of special/automation parameters (M)
<code>da</code>	Display acquisition parameter arrays (C)

dg Control dg parameter group display (P)

Description: Controls the display of the `dg` command for the group of acquisition and 1D/2D processing parameters. `dg`, a string parameter, can be modified with the command `paramvi('dg')`.

See also: *Getting Started*

Related: `dg` Display group of acquisition/processing parameters (C)
`paramvi` Edit a parameter and its attributes with `vi` text editor (C)

dg1 Display group of display parameters (M)

Syntax: `dg1`

Description: Displays the group of display parameters. To display an individual parameter, enter the name of the parameter followed by a question mark (e.g., `sp?`). Parameters do not have to be displayed in order to be entered or changed. The `dg1` display is controlled by the string parameter `dg1`.

See also: *Getting Started*

Related: `?` Display individual parameter value (C)
`dg1` Control `dg1` parameter group display (P)
`dg` Display group of acquisition/processing parameters (C)

dg1 Control dg1 parameter group display (P)

Description: Controls the display of the `dg1` command for the group of display parameters. `dg1`, a string parameter, can be modified with `paramvi('dg1')`.

See also: *Getting Started*

Related: `dg1` Display group of display parameters (M)
`paramvi` Edit a parameter and its attributes with `vi` text editor (C)

dg2 Display group of 3rd and 4th rf channel/3D parameters (M)

Applicability: All systems except *GEMINI 2000*.

Syntax: `dg2`

Description: Displays the group of acquisition parameters associated with a second decoupler channel on a system with a third rf channel. It also displays the group of parameters associated with selective 2D processing of 3D data sets. To display an individual parameter, enter the name of the parameter followed by a question mark (e.g., `sw?`). Parameters do not have to be displayed in order to be entered or changed. The `dg2` display is controlled by the string parameter `dg2`.

See also: *User Guide: Liquids NMR*

Related: `dg` Display group of acquisition/processing parameters (C)
`dg2` Control `dg2` parameter group display (P)

dg2 Control dg2 parameter group display (P)

Applicability: All systems except *GEMINI 2000*.

Description: Controls the display of the `dg2` command for the group of 3rd and 4th rf channel/3D parameters. `dg2`, a string parameter, can be modified with the command `paramvi('dg2')`. To retrieve the `dg2` and `ap` display templates for the current experiment, enter `addpar('3rf')`.

See also: *Getting Started*

Related: `addpar` Add selected parameters to the current experiment (M)
`dg2` Display group of 3rd and 4th rf channel/3D parameters (M)
`paramvi` Edit a parameter and its attributes with `vi` text editor (M)

dga Display group of spin simulation parameters (M)

Syntax: `dga`

Description: Displays the file of spin simulation parameters (Group A). There is one such group of parameters in the data system, not one per experiment as with normal NMR parameters.

Alternate: Show Params button in the Spin Simulation Main Menu.

See also: *User Guide: Liquids NMR*

Related: `dg` Display group of acquisition/processing parameters (C)
`dla` Display spin simulation parameter arrays (C)

DgcsteSL Set up parameters for DgcsteSL pulse sequence (M)

Syntax: `DgcsteSL`

Description: Converts a parameter set to DgcsteSL experiment.

See also: *User Guide: Liquids NMR*

Related: `dosy` Process DOSY experiments (M)
`fiddle` Perform reference deconvolution (M)
`setup_dosy` Set up gradient levels for DOSY experiments (M)

Dgcstecosy Set up parameters for Dgcstecosy pulse sequence (M)

Syntax: `Dgcstecosy`

Description: Converts a parameter set to Dgcstecosy experiment

See also: *User Guide: Liquids NMR*

Related: `dosy` Process DOSY experiments (M)
`makeslice` Synthesize 2D projection of a 3D DOSY spectrum (C)
`setup_dosy` Set up gradient levels for DOSY experiments (M)
`showoriginal` Restore first 2D spectrum in 3D DOSY spectrum (M)

Dgcstehmqc Set up parameters for Dgcstehmqc pulse sequence (M)

Syntax: `Dgcstehmqc`

Description: Converts a parameter set to Dgcstehmqc experiment

See also: *User Guide: Liquids NMR*

Related: `dosy` Process DOSY experiments (M)
`makeslice` Synthesize 2D projection of 3D DOSY spectrum (C)
`setup_dosy` Set up gradient levels for DOSY experiments (M)
`showoriginal` Restore first 2D spectrum in 3D DOSY spectrum (M)

- dg1c** **Display group of LC-NMR parameters (M)**
 Applicability: Systems with LC-NMR accessory.
 Syntax: dg1c
 Description: Displays parameters related to LC-NMR on a separate screen. This macro is equivalent to the command dg('dg1c').
 See also: *User Guide: Liquids NMR*
 Related: **dg1c** Control LC-NMR parameter display (P)
- dg1c** **Control dg1c parameter group display (P)**
 Applicability: Systems with LC-NMR accessory.
 Description: Controls the display of the LC-NMR parameters by the macro **dg1c** and the equivalent command dg('dg1c'). If this parameter does not exist, the **par1c** macro can create it.
 See also: *User Guide: Liquids NMR*
 Related: **dg1c** Display LC-NMR parameters (M)
par1c Create LC-NMR parameters (M)
- dg1p** **Display group of linear prediction parameters (M)**
 Syntax: dg1p
 Description: Displays the group of parameters associated with linear prediction. To display an individual parameter, enter the name of the parameter followed by a question mark (e.g., lppopt?). Parameters do not have to be entered or changed.
 See also: *User Guide: Liquids NMR*
 Related: **dg** Display group of acquisition/processing parameters (C)
- dgm** **Display menu to view parameter screens (C)**
 Applicability: Systems with imaging capabilities.
 Syntax: dgm
 Description: Displays a menu for selecting and viewing a list of parameter screens.
 See also: *User Guide: Imaging*
- dgs** **Display group of shims and automation parameters (M)**
 Syntax: dgs
 Description: Displays the group of shims and automation parameters. To display an individual parameter, enter name of the parameter followed by a question mark (e.g., sw?). Parameters do not have to be displayed in order to be entered or changed. The dgs display is controlled by the parameter **dgs**.
 See also: *User Guide: Liquids NMR*
 Related: **dg** Display group of acquisition/processing parameters (C)
dgs Control dgs parameter group display (P)
- dgs** **Control dgs parameter group display (P)**
 Description: Controls display of the **dgs** command for the group of shims and automation parameters. dgs, a string parameter, can be modified by **paramvi**('dgs').

See also: *Getting Started*

Related: **dgs** Display group of special/automation parameters (M)
paramvi Edit a parameter and its attributes with `vi` text editor (C)

dhp Decoupler high-power control with class C amplifier (P)

Applicability: System with a class C amplifier.

Description: On *GEMINI 2000* $^1\text{H}/^{13}\text{C}$ systems, controls decoupler high power for proton heteronuclear decoupling. `dhp` is ignored if protons are being observed (`tn=1.0`). (*GEMINI 2000* broadband systems use `dpwr` to adjust decoupler power for both homonuclear and heteronuclear decoupling.)

On systems other than the *GEMINI 2000*, `dhp` selects a decoupler high-power level for systems with class C amplifiers on the decoupler channel. Specific values of `dhp` should be calibrated periodically for any particular instrument and probe combination. As a rough guide, `dhp=75` corresponds to approximately 2 watts at 200 MHz.

CAUTION: Decoupler power greater than 2 watts in a switchable probe will damage the probe. Always carefully calibrate high-power decoupling to avoid exceeding 2 watts of power.

For systems equipped with a linear amplifier on the decoupler channel, `dhp` is nonfunctional and is replaced by the parameter `dpwr`.

Note that `dhp` runs in the opposite direction from `dlp` (i.e., for `dhp` a higher number means more power, for `dlp` a higher number means less power).

Values: On *GEMINI 2000* $^1\text{H}/^{13}\text{C}$ systems, 0.5 or 1.0, in watts. On other systems, 0 to 255 (where 255 is maximum power) in uncalibrated, non-linear units.

'n' selects low-power decoupling under the control of the parameter `dlp`.

See also: *Getting Started*

Related: **dlp** Decoupler low power with class C amplifier (P)
dpwr Power level for first decoupler with linear amplifier (P)
tn Nucleus for observe transmitter (P)

dialog Display a dialog box from a macro (C)

Syntax: `dialog(definition_file,output_file<,'nowait'>)`

Description: Opens a dialog box from a macro. The output is written to a file that can be read by the macro using the `lookup` command.

Arguments: `definition_file` is the name of the file (specified by an absolute path) that defines the layout of the dialog box. The structure of the file is the same as the definition files for *GLIDE*.

`output_file` is the name of the file (specified by an absolute path) where the results of the dialog box are written.

'nowait' is a keyword to return immediately, without waiting for input into the dialog box.

Examples: `dialog(userdir+' /dialoglib/array', '/tmp/array')`

See also: *VNMR User Programming*

Related: **lookup** Look up words and lines from a text file (C)

diffshims Compare two sets of shims (M,U)

Syntax: (From VNMR) `diffshims(shimfile1, shimfile2)`
 (From UNIX) `diffshims shimfile1 shimfile2`

Description: Compares values for room-temperature shims stored in two separate files.

Arguments: `shimfile1` and `shimfile2` are names of separate files containing shim values. Both files must have been written using the `svs` command.

See also: *Getting Started*

Related: `svs` Save shim coil settings (C)

digfilt Write digitally filtered FIDs to another experiment (M)

Syntax: `digfilt(exp_number<, option>)`

Description: Saves digitally filtered FIDs to another experiment.

Arguments: `exp_number` specifies the number of the experiment, from 1 to 9, for saving the FIDs.

`option` is one of the keywords 'nodc', 'zero', 'lfs', 'zfs', or 't2dc'. Use a keyword for an option if the same option was used when processing the data with `ft`, `wft`, `ft2d`, or `wft2d`.

See also: *Getting Started*

Related: `downsamp` Sampling factor applied after digital filtering (P)
`ft` Fourier transform 1D data (C)
`ft2d` Fourier transform 2D data (C)
`wft` Weight and Fourier transform 1D data (C)
`wft2d` Weight and Fourier transform 2D data (C)

dir List files in directory (C)

Syntax: `dir<(string)>`

Description: Displays files in a directory on the text window. The `dir` command is identical to the `ls` and `lf` commands.

Arguments: `string` is a string argument containing the options and/or directory names used if this were the UNIX `ls` command (e.g., `dir('-l *.fid'`) requests a long listing (-l) of all files ending with `.fid` (`*.fid`)). If no argument is entered, `dir` lists all files in the current working directory.

Examples: `dir`
`dir('data'`)
`dir('-l *.fid'`)

See also: *Getting Started*

Related: `lf` List files in directory (C)
`ls` List files in directory (C)

disp3d Display 3D data (U)

Applicability: Systems with imaging capabilities.

Syntax: (From UNIX) `disp3d <fdf_file>`

Description: Displays a 3D FDF (Flexible Data Format) file or a raw 8-bit 3D data file with no header. Compatible FDF files are produced by `ft3d` with the 'fdf' option (or by default if `appmode= 'imaging'`).

FDF data can also be loaded either by entering the file name as an argument to `disp3d` or by typing the file name into the File field in the `disp3d` control panel and clicking the Load button. If the FDF data word size is larger than 8 bits, the data are scaled and truncated to 8 bits for display. Raw data files can only be loaded from the control panel.

Besides the file name, the user must enter the size of the data matrix in the fast, medium, and slow dimensions in the Data size field. Typically, these would be the values `fn/2`, `fn1/2`, and `fn2/2`, respectively.

Furthermore, the desired size of the image in screen pixels—also in the fast, medium, and slow dimensions—must be entered in the Display size fields. Typically, these values would be near 100 and the relative ratio of the parameters `lro`, `lpe`, and `lpe2`, respectively.

After loading the data, a 3D volume appears in the display panel.

Arguments: `fdf_file` is the name of a file containing FDF data.

See also: *User Guide: Imaging*

Related:	<code>appmode</code>	Application mode (P)
	<code>fn</code>	Fourier number in directly detected dimension (P)
	<code>fn1</code>	Fourier number in 1st indirectly detected dimension (P)
	<code>fn2</code>	Fourier number in 2nd indirectly detected dimension (P)
	<code>ft3d</code>	Perform a 3D Fourier transform on a 3D FID data set (M,U)
	<code>lpe</code>	Field of view size for phase encode axis (P)
	<code>lpe2</code>	Field of view size for 2nd phase-encode axis (P)
	<code>lro</code>	Field of view size for readout axis (P)

display **Display parameters and their attributes (C)**

Syntax: `display(parameter | '*' | '**' <, tree>)`

Description: Displays one or more parameters and their attributes from a parameter tree.

Arguments: Three levels of display are available: `parameter`, `'*'`, and `**'`.

- `parameter` is the name of a single parameter and the display is of its attributes (e.g., `display('a')` displays the attributes of parameter `a` in the (default) current tree).
- `'*'` is a keyword to display the name and values of all parameters in a tree (e.g., `display('*', 'global')` displays all parameter names and values in the global tree).
- `**'` is a keyword to display the attributes of all parameters in a tree (e.g., `display('**', 'processed')` displays the attributes of all parameters in the processed tree).

`tree` is the type of parameter tree and can be `'global'`, `'current'`, `'processed'`, or `'systemglobal'`. The default is `'current'`. Refer to the `create` command for more information on types of trees.

Examples: `display('a')`
`display('*', 'global')`
`display('**', 'processed')`

See also: *VNMR User Programming*

Related:	<code>create</code>	Create new parameter in a parameter tree (C)
	<code>destroy</code>	Destroy a parameter (C)
	<code>paramvi</code>	Edit a parameter and its attributes with the <code>vi</code> text editor (C)
	<code>prune</code>	Prune extra parameters from current tree (C)

dla **Display spin simulation parameter arrays (M)**

Syntax: `dla<('long')>`

Description: Displays the parameters containing the line assignments for spin simulation iteration (matching simulated spectra to actual data). A `clindex` value of a calculated transition gives the index of the assigned measured line. The value is zero for unassigned transitions.

Arguments: 'long' is a keyword to display the parameters containing the line assignments for spin simulation iteration (matching simulated spectra to actual data) and put the line assignments into the file `spini.la`. This option is most useful when the `dla` display is too large to display all the calculated transitions in the VNMR text window. The `dlalong` command operates the same as the `dla('long')` command.

Examples: `dla`
`dla('long')`

See also: *User Guide: Liquids NMR*

Related:	<code>assign</code>	Assign transitions to experimental lines (M)
	<code>clindex</code>	Index of experimental frequency of a transition (P)
	<code>dga</code>	Display parameters of spin simulation group (C)
	<code>dlalong</code>	Long display of spin simulation parameter arrays (C)

dlalong **Long display of spin simulation parameter arrays (C)**

Syntax: `dlalong`

Description: Puts line assignments into the file `spini.la` in a more complete form, then displays this file in the text window. It is most useful when the `dla` display is too large to display all the calculated transitions in the VNMR text window. The `dla('long')` command operates the same as `dlalong`.

See also: *User Guide: Liquids NMR*

Related:	<code>dla</code>	Display spin simulation parameter arrays (M)
----------	------------------	--

dli **Display list of integrals (C)**

Syntax: `dli`

Description: Displays a list of integrals at the integral reset points. The frequency units of the displayed list of integrals is controlled by the parameter `axis`. The reset points may be defined with the `z` command and these frequencies are stored in `lifrq`. The calculated amplitudes of the integral region are stored in `liamp`. The reset points are stored as hertz and are not referenced to `rfl` and `rfp`. The amplitudes are stored as the actual value; they are not scaled by `ins` or by `insref`. When the integral blanking mode is used (i.e., `intmod='partial'`), only the integrals corresponding to the displayed integral regions are listed.

The displayed integral value can be scaled with the `setint` macro. The integral is scaled by the parameters `ins` and `insref`.

Alternate: Integrals button in the 1D Data Display Secondary Menu.

See also: *Getting Started*

Related:	<code>axis</code>	Axis label for displays and plots (P)
	<code>cz</code>	Clear integral reset points (C)
	<code>dlni</code>	Display list of normalized integrals (M)
	<code>ins</code>	Integral normalization scale (P)
	<code>insref</code>	Fourier number scaled value of an integral (P)

<code>liamp</code>	Amplitudes of integral reset points (P)
<code>lifrq</code>	Frequencies of integral reset points (P)
<code>nli</code>	Find integral values (C)
<code>rfl</code>	Reference peak position in directly detected dimension (P)
<code>rfp</code>	Reference peak frequency in directly detected dimension (P)
<code>setint</code>	Set value of an integral (M)
<code>z</code>	Add integral reset point at cursor position (C)

dlivast Produce text file and process wells (M)

Syntax: `dlivast<(last)>`

Description: Produces a text file containing the integral of the partial regions and processes the wells.

Arguments: `last` is the number of the last well. The default is 96.

See also: *User Guide: Liquids NMR*

Related: `combiplate` View a color map for visual analysis of VAST microtiter plate (U)
`combishow` Display regions as red, green, and blue in CombiPlate window (M)

d11 Display listed line frequencies and intensities (C)

Syntax: `d11<('pos'<,noise_mult)>><:number_lines, scale>`

Description: Displays a list of line frequencies and amplitudes that are above a threshold defined by `th`. Frequency units are defined by the parameter `axis`. The results of this calculation are stored in `llfrq` and `llamp`. The frequencies are stored as Hz and are not referenced to `rfl` and `rfp`. Amplitudes are stored as the actual data point value; they are not scaled by `vs`.

Arguments: `'pos'` is a keyword to list only positive lines.

`noise_mult` is a numerical value that determines the number of noise peaks listed for broad, noisy peaks. The default value is 3. A smaller value results in more peaks, a larger value results in fewer peaks, and a value of 0.0 results in a line listing containing all peaks above the threshold `th`. Negative values of `noise_mult` are changed to 3.

`number_lines` is a return argument with the number of lines above the threshold.

`scale` is a return argument with a scaling factor for line amplitudes. This scaling factor accounts for `vs` and whether the lines are listed in absolute intensity mode or normalized mode.

Examples: `d11`
`d11('pos')`
`d11(2.5)`
`d11:rl,sc`

Alternate: Lines button in the 1D Data Display Secondary menu.

See also: *Getting Started; User Guide: Liquids NMR*

Related: `axis` Axis label for displays and plots (P)
`dels` Delete spectra from T_1 or T_2 analysis (C)
`fp` Find peak heights (C)
`get11` Get frequency and intensity of a line (C)
`llamp` List of line amplitudes (P)
`llfrq` List of line frequencies (P)
`nl` Position the cursor at the nearest line (C)
`n11` Find line frequencies and intensities (C)

<code>rfl</code>	Reference peak position in directly detected dimension (P)
<code>rfp</code>	Reference peak frequency in directly detected dimension (P)
<code>th</code>	Threshold (P)
<code>vs</code>	Vertical scale (P)

dlni **Display list of normalized integrals (M)**

Syntax: `dlni`

Description: Displays integrals in a normalized format. The parameter `ins` represents the value of the sum of all the integrals. When the integral blanking mode is used (i.e., `intmod= 'partial'`), only the integrals corresponding to the displayed integral regions are listed and are used in the summation.

See also: *Getting Started*

<code>cz</code>	Clear integral reset points (C)
<code>dli</code>	Display list of integrals (C)
<code>ins</code>	Integral normalization scale (P)
<code>nli</code>	Find integral values (C)
<code>z</code>	Add integral reset point at cursor position (C)

d1p **Decoupler low-power control with class C amplifier (P)**

Applicability: Systems with a class C amplifier.

Description: On *GEMINI 2000* $^1\text{H}/^{13}\text{C}$ systems, `d1p` controls the proton homodecoupler power level, if present.

On *GEMINI 2000* broadband systems with a relay switching version of the RF Control board, `d1p` has no meaning (refer to the description of the parameter `attens` for information on RF Control boards).

On *GEMINI 2000* broadband systems with a diode switching version of RF Control board, `d1p` controls a fine attenuator over a range of approximately 14 dB. In line with this attenuator is a coarse attenuator controlled by `dpwr` and `pp1vl`. Unless fine control is necessary, `d1p=1023` (maximum power) is recommended. `d1p` affects pulse and CW decoupler power; therefore, it affects both the γH_2 of the 90° decoupler pulse and `dmf`.

On systems other than *GEMINI 2000*, `d1p` controls the decoupler power level for systems with a class C decoupler amplifier in the low-power mode, generally used for homonuclear decoupling. `d1p` specifies dB of attenuation of the decoupler, below a nominal 1 watt value. `d1p` is active only if `dhp= 'n'`.

On systems with a decoupler linear amplifier, `d1p` is nonfunctional and `dpwr` controls decoupler power.

Values: On *GEMINI 2000* $^1\text{H}/^{13}\text{C}$ systems, 0 to 2047 in arbitrary units (2047 is full power). On *GEMINI 2000* broadband systems with the diode switching version of the RF Control board, 0 to 1023 in arbitrary units (1023 is full power). On systems other than the *GEMINI 2000*, 0 to 39 (in dB of attenuation, 0 is maximum power).

See also: *Getting Started*

Related:	<code>attens</code>	Fast attenuators present (P)
	<code>dhp</code>	Decoupler high-power control with class C amplifier (P)
	<code>dm</code>	Decoupler mode for first decoupler (P)
	<code>dmf</code>	Decoupler modulation frequency for first decoupler (P)
	<code>dpwr</code>	Power level for first decoupler with linear amplifier (P)
	<code>hdofst</code>	Proton homonuclear decoupler offset (P)

`homdec` Proton homonuclear decoupler present (P)
`pplvl` Proton pulse power level (P)

`dm` Decoupler mode for first decoupler (P)

Description: Determines the state of first decoupler during different status periods within a pulse sequence (refer to the manual *VNMR User Programming* for a discussion of status periods). Pulse sequences may require one, two, three, or more different decoupler states. The number of letters that make up the `dm` parameter vary appropriately, with each letter representing a status period (e.g., `dm= 'yNy'` or `dm= 'ns'`). If the decoupler status is constant for the entire pulse sequence, it can be entered as a single letter (e.g., `dm= 'n'`).

Values: 'n', 'y', 'a', or 's' (or a combination of these values), where:

'n' specifies no decoupler rf.

'y' specifies the asynchronous mode. In this mode, the decoupler rf is gated on and modulation is started at a random places in the modulation sequence.

'a' specifies the asynchronous mode, the same as 'y'. The 'a' value is not available on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems.

's' specifies the synchronous mode in which the decoupler rf is gated on and modulation is started at the beginning of the modulation sequence. This value has meaning only on *UNITYINOVA* and *UNITYplus* systems. On *UNITY* and *VXR-S* systems it is equivalent to 'y'. The 's' value is not available on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

See also: *Getting Started*

Related: `dm2` Decoupler mode for second decoupler (P)
`dm3` Decoupler mode for third decoupler (P)
`dm4` Decoupler mode for fourth decoupler (P)
`dmf` Decoupler modulation frequency for first decoupler (P)
`dmm` Decoupler modulation mode for first decoupler (P)
`dn` Nucleus for first decoupler (P)

`dm2` Decoupler mode for second decoupler (P)

Applicability: Systems with a second decoupler.

Description: Determines the state of second decoupler during different status periods within a pulse sequence. It functions analogously to `dm`.

Values: Same as `dm`, except that if `dn2= ''` (two single quotes with no space in between) and a second decoupler is present in the console, `dm2` assumes a default value of 'n' when `go` is executed.

See also: *Getting Started*

Related: `dm` Decoupler mode of first decoupler (P)
`dmf2` Decoupler modulation frequency for second decoupler (P)
`dmm2` Decoupler modulation mode for second decoupler (P)
`dn2` Nucleus for second decoupler (P)

`dm3` Decoupler mode for third decoupler (P)

Applicability: Systems with a third decoupler.

Description: Determines the state of third decoupler during different status periods within a pulse sequence. It functions analogously to `dm`.

Values: Same as `dm`, except that if `dn3= ' '` (two single quotes with no space in between) and a third decoupler is present in the console, `dm3` assumes a default value of 'n' when `go` is executed.

See also: *Getting Started*

Related: `dm` Decoupler mode of first decoupler (P)
`dmf3` Decoupler modulation frequency for third decoupler (P)
`dmm3` Decoupler modulation mode for third decoupler (P)
`dn3` Nucleus for third decoupler (P)

`dm4` Decoupler mode for fourth decoupler (P)

Applicability: Systems with a deuterium decoupler channel as the fourth decoupler.

Description: Determines the state of fourth decoupler during different status periods within a pulse sequence. It functions analogously to `dm`.

Values: Same as `dm`, except that if `dn4= ' '` (two single quotes with no space in between) and a fourth decoupler is present in the console, `dm4` assumes a default value of 'n' when `go` is executed.

See also: *Getting Started*

Related: `dm` Decoupler mode of first decoupler (P)
`dmf4` Decoupler modulation frequency for fourth decoupler (P)
`dmm4` Decoupler modulation mode for fourth decoupler (P)
`dn4` Nucleus for fourth decoupler (P)

`dmf` Decoupler modulation frequency for first decoupler (P)

Description: Controls modulation frequency of the first decoupler. It specifies 1/`pw90` at the particular power level used. After calibrating the decoupler field strength γH_2 (expressed in units of Hz), `dmf` should be set equal to $4 \cdot \gamma H_2$ for WALTZ, MLEV16, GARP, and XY32 (when available).

`dmf` is inactive for CW mode decoupling (`dmm= ' c '`).

On *UNITYINOVA*, *MERCURY-VX*, *MERCURY*, and *UNITYplus*, `dmf` is also active for square wave mode decoupling (`dmm= ' r '`) and fm-fm mode (`dmm= ' f '`) decoupling. For `dmm= ' f '`, the modulation frequency is swept back and forth between about 0.5% and 5% of the `dmf` frequency (e.g., if `dmf` is 100 kHz, the modulation is swept between approximately 500 Hz and 5 kHz). A reasonable optimum value for `dmf` when `dmm= ' f '` is the decoupler frequency divided by 4000.

On *GEMINI 2000*, *UNITY*, and *VXR-S*, `dmf` is fixed at 75 kHz for fm-fm mode decoupling (`dmm= ' f '`) and noise mode decoupling (`dmm= ' n '`).

Values: On *UNITYINOVA*, *MERCURY-VX*, *MERCURY*, and *UNITYplus*: 5 Hz to 2 MHz in steps of 5 Hz (steps are actually approximately 4.768 Hz). On *GEMINI 2000*: 100 to 25000 Hz, in steps of 100 Hz. On *UNITY* and *VXR-S*: 100 Hz to 990 or 32000 Hz, in steps of 100 Hz.

For GARP modulation, the `dmf` value is internally multiplied by 45, making the limit of possible `dmf` values to 5 Hz to 44.4 kHz when `dmm= ' g '`.

See also: *Getting Started*

Related: `dmf2` Decoupler modulation frequency for second decoupler (P)
`dmf3` Decoupler modulation frequency for third decoupler (P)
`dmf4` Decoupler modulation frequency for fourth decoupler (P)
`dmm` Decoupler modulation mode for first decoupler (P)
`pw90` 90° pulse width (P)

dmf2 Decoupler modulation frequency for second decoupler (P)

Applicability: Systems with a second decoupler.

Description: Controls the modulation frequency of the second decoupler. It functions analogously to the parameter `dmf`.

Values: Same as `dmf` except that if `dn2=' '` (two single quotes with no space in between) and a second decoupler is present in the console (`numrfch` greater than 2), `dmf2` assumes a default value of 1000 Hz when `go` is executed.

See also: *Getting Started*

Related: `dm2` Decoupler mode for second channel (P)
`dmf` Decoupler modulation frequency for first decoupler (P)
`dmm2` Decoupler modulation mode for second decoupler (P)
`dn2` Nucleus for second decoupler (P)
`numrfch` Number of rf channels (P)

dmf3 Decoupler modulation frequency for third decoupler (P)

Applicability: Systems with a third decoupler.

Description: Controls the modulation frequency of the third decoupler. It functions analogously to the parameter `dmf`.

Values: Same as `dmf` except that if `dn3=' '` (two single quotes with no space in between) and a third decoupler is present in the console (`numrfch` equals 4), `dmf3` assumes a default value of 1000 Hz when `go` is executed.

See also: *Getting Started*

Related: `dm3` Decoupler mode for third channel (P)
`dmf` Decoupler modulation frequency for first decoupler (P)
`dmm3` Decoupler modulation mode for third decoupler (P)
`dn3` Nucleus for third decoupler (P)
`numrfch` Number of rf channels (P)

dmf4 Decoupler modulation frequency for fourth decoupler (P)

Applicability: Systems with a deuterium decoupler channel as the fourth decoupler.

Description: Controls the modulation frequency of the fourth decoupler. It functions analogously to the parameter `dmf`.

Values: Same as `dmf` except that if `dn4=' '` (two single quotes with no space in between) and a fourth decoupler is present in the console (`numrfch` equals 5), `dmf4` assumes a default value of 1000 Hz when `go` is executed.

See also: *Getting Started*

Related: `dm4` Decoupler mode for fourth channel (P)
`dmf` Decoupler modulation frequency for first decoupler (P)
`dmm4` Decoupler modulation mode for fourth decoupler (P)
`dn4` Nucleus for fourth decoupler (P)
`numrfch` Number of rf channels (P)

dmfadj Adjust tip-angle resolution time for first decoupler (M)

Applicability: All systems except *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Syntax: `dmfadj<(tipangle_resolution)>`

Description: Adjusts the parameter `dmf` so that time associated with the first decoupler tip-angle resolution is an integral multiple of 50 ns (*UNITYINOVA* and *UNITYplus*)

or 100 ns (UNITY and VXR-S). This eliminates time truncation error in execution of programmable decoupling or spin-locking sequence by the waveform generator. For example, the tip-angle resolution for an MLEV-16 decoupling sequence should be 90.0° since every pulse in that sequence can be represented as an integral multiple of 90.0°; however, the tip-angle resolution for a GARP decoupling sequence should be 1.0°.

Arguments: `tipangle_resolution` specifies the necessary tip-angle resolution for the programmable decoupling or spin-locking sequence to be executed. The default value is the current value of the parameter `dres`.

Examples: `dmfadj`
`dmfadj(90.0)`

See also: *Getting Started*

Related: `dmf` Decoupler modulation frequency for first decoupler (P)
`dmf2adj` Adjust tip-angle resolution time for second decoupler (M)
`dmf3adj` Adjust tip-angle resolution time third decoupler (M)
`dmf4adj` Adjust tip-angle resolution time fourth decoupler (M)
`dres` Tip angle resolution for programmable decoupling (P)

`dmf2adj` Adjust tip-angle resolution time for second decoupler (M)

Applicability: Systems with a second decoupler.

Syntax: `dmf2adj<(tipangle_resolution)>`

Description: Adjusts the parameter `dmf2` to make time associated with the second decoupler tip-angle resolution an integral multiple of 50 ns (*UNITYINOVA* and *UNITYplus*) or 100 ns (UNITY and VXR-S). `dmf2adj` functions analogously to the macro `dmfadj`.

Arguments: `tipangle_resolution` specifies the necessary tip-angle resolution for the programmable decoupling or spin-locking sequence to be executed. The default value is the current value of the parameter `dres2`.

Examples: `dmf2adj`
`dmf2adj(90.0)`

See also: *Getting Started*

Related: `dmf2` Decoupler modulation frequency for second decoupler (P)
`dmfadj` Adjust decoupler tip-angle resolution time (M)
`dres2` Tip angle resolution for second decoupler (P)

`dmf3adj` Adjust tip-angle resolution time for third decoupler (M)

Applicability: Systems with a third decoupler.

Syntax: `dmf3adj<(tipangle_resolution)>`

Description: Adjusts the parameter `dmf3` to make time associated with the third decoupler tip-angle resolution an integral multiple of 50 ns (*UNITYINOVA* and *UNITYplus*) or 100 ns (UNITY and VXR-S). `dmf3adj` functions analogously to the macro `dmfadj`.

Arguments: `tipangle_resolution` specifies the necessary tip-angle resolution for the programmable decoupling or spin-locking sequence to be executed. The default value is the current value of the parameter `dres3`.

Examples: `dmf3adj`
`dmf3adj(90.0)`

See also: *Getting Started*

Related: **dmf3** Decoupler modulation frequency for third decoupler (P)
dres3 Tip-angle resolution for third decoupler (P)

dmf4adj Adjust tip-angle resolution time for fourth decoupler (M)

Applicability: Systems with a deuterium decoupler as the fourth decoupler.

Syntax: `dmf4adj<(tipangle_resolution)>`

Description: Adjusts the parameter **dmf4** to make time associated with the fourth decoupler tip-angle resolution an integral multiple of 50 ns (^{UNITY}INNOVA). `dmf4adj` functions analogously to the macro `dmfadj`.

Arguments: `tipangle_resolution` specifies the necessary tip-angle resolution for the programmable decoupling or spin-locking sequence to be executed. The default value is the current value of the parameter **dres4**.

Examples: `dmf4adj`

See also: *Getting Started*

Related: **dmf4** Decoupler modulation frequency for fourth decoupler (P)
dres4 Tip-angle resolution for fourth decoupler (P)

dmg Data display mode in directly detected dimension (P)

Description: Controls the mode of data display along the directly detected dimension. `dmg` is in the display group and can be set manually or by executing the commands **ph**, **av**, **pwr**, or **pa** for the values 'ph', 'av', 'pwr', or 'pa', respectively.

Values: 'ph' sets the *phased mode* in which each real point in the displayed spectrum is calculated from a linear combination of real and imaginary points comprising each respective complex data point.

'av' sets the *absolute-value mode* in which each real point in the displayed spectrum is calculated as the square root of the sum of squares of the real and imaginary points comprising each respective complex data point.

'pwr' sets the *power mode* in which each real point in the displayed spectrum is calculated as the sum of squares of the real and imaginary points comprising each respective complex data point.

'pa' sets the *phase angle mode* in which each real point in the displayed spectrum is calculated as the phase angle from the arc tangent of the real and imaginary points comprising each respective complex data point.

See also: *User Guide: Liquids NMR*

Related: **aig** Absolute intensity group (P)
av Set absolute-value mode in directly detected dimension (C)
dcg Drift correction group (P)
dmg1 Data display mode in 1st indirectly detected dimension (P)
dmg2 Data display mode in 2nd indirectly detected dimension (P)
ft Fourier transform 1D data (C)
ft1d Fourier transform along f_2 dimension (C)
ft2d Fourier transform 2D data (C)
pa Set phase angle mode in directly detected dimension (C)
ph Set phased mode in directly detected dimension (C)
pmode Processing mode for 2D data (P)
pwr Set power mode in directly detected dimension (C)
wft Weigh and Fourier transform 1D data (C)

`wft1d` Weigh and Fourier transform of 2D data (C)
`wft2d` Weigh and Fourier transform 2D data (C)

`dmg1` **Data display mode in 1st indirectly detected dimension (P)**

Description: Controls the mode of data display along the first indirectly detected dimension of a multidimensional data set. `dmg1` is in the display group and can be set manually or by executing the commands `ph1`, `av1`, `pwr1`, or `pa1` for the values 'ph1', 'av1', 'pwr1', or 'pa1', respectively. If `dmg1` does not exist or if it is set to the empty string (`dmg1= ''`), VNMR uses the value of `dmg` to decide the display mode along the first indirectly detected dimension.

Values: 'ph1' sets phased mode.
 'av1' sets absolute-value mode.
 'pwr1' sets power mode.
 'pa1' sets phase angle mode.

See also: *User Guide: Liquids NMR*

Related: `av1` Set absolute-value mode in 1st indirectly det. dim. (C)
`dmg` Data display mode in directly detected dimension (P)
`pa1` Set phase angle mode in 1st indirectly detected dimension (C)
`ph1` Set phased mode in 1st indirectly detected dimension (C)
`pwr1` Set power mode in 1st indirectly detected dimension (C)

`dmg2` **Data display mode in 2nd indirectly detected dimension (P)**

Applicability: All systems except *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Description: Controls the mode of data display along the second indirectly detected dimension of a multidimensional data set. `dmg2` is in the display group and can be set manually or by executing the commands `ph2`, `av2`, or `pwr2` for the values 'ph2', 'av2', or 'pwr2', respectively. If `dmg2` does not exist or if it is set to the empty string (`dmg2= ''`), VNMR uses the value of the parameter `dmg` instead of `dmg2` to decide the display mode along the second indirectly detected dimension.

Values: 'ph2' sets phased mode.
 'av2' sets absolute-value mode.
 'pwr2' sets power mode.

See also: *User Guide: Liquids NMR*

Related: `av2` Set absolute-value mode in 2nd indirectly det. dim. (C)
`dmg` Data display mode in directly detected dimension (P)
`ph2` Set phased mode in 2nd indirectly det. dim. (C)
`pwr2` Set power mode in 2nd indirectly det. dim. (C)

`dmgf` **Absolute-value display of FID data or spectrum in `acqi` (P)**

Description: If the parameter `dmgf` exists and is set to 'av', the FID display in the `acqi` program is set to the absolute-value mode, which displays the square root of the sum of the squares of the real and imaginary channels. `dmgf` has no function outside of the `acqi` program. This display mode may cause the displayed FID to exceed the displayed ADC limits in `acqi` by as much as a factor of the square root of 2.

See also: *Getting Started*

Related: **acqi** Interactive acquisition display process (C)
av Set absolute-value mode in directly detected dimension (C)
gf Prepare parameters for FID/spectrum display in **acqi** (M)

dmi **Display multiple images (M)**

Applicability: Systems with imaging capabilities.

Syntax: **dmi**

Description: Displays a series of multiple images from a single arrayed and/or multislice/multiecho experiment in the graphics window. The resulting display is noninteractive. The layout and size of the images are optimized to maximize the image display size.

See also: *User Guide: Imaging*

Related: **svib** Generate and save images as ImageBrowser FDF files (M)

dmm **Decoupler modulation mode for first decoupler (P)**

Description: Sets the modulation modes for the first decoupler. In the standard two-pulse sequence, **dmm** typically has a single state because the decoupler modulation is normally not changed during the pulse sequence, but this is not fixed. For example, **dmm**= 'ccw' gives single-frequency CW decoupling during the first part of the sequence and WALTZ-16 decoupling during acquisition.

In pulse sequences using the decoupler for pulsing (INEPT, DEPT, HETCOR, etc.), decoupler modulation must be set to 'c' during periods of the pulse sequence when the decoupler is to be pulsed.

Values: On ^{UNITY}*INOVA* and *UNITYplus*, 'c', 'f', 'g', 'm', 'p', 'r', 'u', 'w', and 'x' are available; on *MERCURY-VX* and *MERCURY*, 'c', 'f', 'g', 'm', 'r', 'w', and 'x' are available; on *VXR-S* and *UNITY*, 'c', 'f', 'n', 'p', and 'w' are available; and on *GEMINI 2000*, 'c', 'f', 'r', and 'w' are available, where:

- 'c' sets continuous wave (CW) modulation.
- 'f' sets fm-fm modulation (swept-square wave).
- 'g' sets GARP modulation.
- 'm' sets MLEV-16 modulation.
- 'n' sets noise modulation.
- 'p' sets programmable pulse modulation using the **dseq** parameter to specify the decoupling sequence.
- 'r' sets square-wave modulation.
- 'u' sets user-supplied modulation using external hardware.
- 'w' sets WALTZ-16 modulation.
- 'x' sets XY32 modulation.

See also: *Getting Started*

Related: **dm** Decoupler mode for first decoupler (P)
dmf Decoupler modulation frequency for first decoupler (P)
dmm2 Decoupler modulation mode for second decoupler (P)
dmm3 Decoupler modulation mode for third decoupler (P)
dmm4 Decoupler modulation mode for fourth decoupler (P)
dseq Decoupler sequence for the first decoupler (P)

dmm2 Decoupler modulation mode for second decoupler (P)

Applicability: Systems with a second decoupler.

Description: Sets the type of decoupler modulation for the second decoupler during different status periods within a pulse sequence. It functions analogously to **dmm**.

Values: For ^{UNITY}*INOVA* and *UNITYplus*, 'c', 'f', 'g', 'm', 'p', 'r', 'u', 'w', and 'x' are available. For VXR-S and UNITY, 'c' and 'p' are available. Refer to **dmm** for the definition of these values (note that if the mode 'p' is selected, **dseq2** specifies the decoupling sequence). If **dn2**=' ' (two single quotes) and a second decoupler is present in the console (**numrfch** greater than 2), **dmm2** is internally set to 'c' when **go** is executed.

See also: *Getting Started*

Related: **dm2** Decoupler modulation for the second decoupler (P)
dmf2 Decoupler modulation frequency for the second decoupler (P)
dmm Decoupler modulation mode for first decoupler (P)
dn2 Nucleus for the second decoupler (P)
dseq2 Decoupler sequence for the second decoupler (P)
numrfch Number of rf channels (P)

dmm3 Decoupler modulation mode for third decoupler (P)

Applicability: Systems with a third decoupler.

Description: Sets type of decoupler modulation for the third decoupler during different status periods within a pulse sequence. It functions analogously to **dmm**.

Values: For ^{UNITY}*INOVA* and *UNITYplus*, 'c', 'f', 'g', 'm', 'p', 'r', 'u', 'w', and 'x' are available. Refer to **dmm** for the definition of these values (note that if the mode 'p' is selected, **dseq3** specifies the decoupling sequence). If **dn3**=' ' (two single quotes) and a third decoupler is present in the console (**numrfch** equal to 4), **dmm3** is internally set to 'c' when **go** is executed.

See also: *Getting Started*

Related: **dm3** Decoupler modulation for third decoupler (P)
dmf3 Decoupler modulation frequency for third decoupler (P)
dmm Decoupler modulation mode for first decoupler (P)
dn3 Nucleus for the third decoupler (P)
dseq3 Decoupler sequence for the third decoupler (P)
numrfch Number of rf channels (P)

dmm4 Decoupler modulation mode for fourth decoupler (P)

Applicability: Systems with a deuterium decoupler channel as the fourth decoupler.

Description: Sets type of decoupler modulation for the fourth decoupler during different status periods within a pulse sequence. It functions analogously to **dmm**.

Values: For ^{UNITY}*INOVA*, 'c', 'f', 'g', 'm', 'r', 'u', 'w', and 'x' are available. Refer to **dmm** for the definition of these values. If **dn4**=' ' (two single quotes) and a fourth decoupler is present in the console (**numrfch** greater than 4), **dmm4** is internally set to 'c' when **go** is executed.

See also: *Getting Started*

Related: **dm4** Decoupler modulation for the fourth decoupler (P)
dmf4 Decoupler modulation frequency for the fourth decoupler (P)
dmm Decoupler modulation mode for first decoupler (P)
dn4 Nucleus for the fourth decoupler (P)

dseq4 Decoupler sequence for the fourth decoupler (P)
numrfch Number of rf channels (P)

dn Nucleus for first decoupler (P)

Description: Changing the value of **dn** causes a macro (named `_dn`) to be executed that extracts values for **dfrq** and **dof** from lookup tables. The tables, stored in the directory `/vnmr/nuctables`, are coded by atomic weights.

Values: In the lookup tables, typically 'H1', 'C13', 'P31', etc.

See also: *Getting Started*

Related: **dfrq** Transmitter frequency of first decoupler (P)
dn2 Nucleus for second decoupler (P)
dn3 Nucleus for third decoupler (P)
dn4 Nucleus for fourth decoupler (P)
dof Frequency offset for first decoupler (C)
tn Nucleus for observe transmitter (P)

dn2 Nucleus for second decoupler (P)

Applicability: Systems with a second decoupler.

Description: Changing the value of **dn2** causes a macro (named `_dn2`) to be executed that extracts values for **dfrq2** and **dof2** from lookup tables. Otherwise, **dn2** functions analogously to the parameters **tn** and **dn**. If an experiment does not use the second decoupler channel, the channel can be disabled by setting `dn2=' '` (two single quotes with no space in between). This sets **dm2**='n', **dmm2**='c', **dmf2**=1000 (in Hz), **dfrq2**=1 (in MHz), **dof2**=0, **dpwr2**=0, **homo2**='n', **dseq2**=' ', and **dres2**=1.

See also: *Getting Started*

Related: **dfrq2** Transmitter frequency of second decoupler (P)
dn Nucleus for first decoupler (P)
dof2 Frequency offset for second decoupler (C)
numrfch Number of rf channels (P)
tn Nucleus for observe transmitter (P)

dn3 Nucleus for third decoupler (P)

Applicability: Systems with a third decoupler.

Description: Changing the value of **dn3** causes a macro (named `_dn3`) to be executed that extracts values for **dfrq3** and **dof3** from lookup tables. Otherwise, **dn3** functions analogously to the parameters **tn** and **dn**. If an experiment does not use the third decoupler channel, the channel can be disabled by setting `dn3=' '` (two single quotes with no space in between). This sets **dm3**='n', **dmm3**='c', **dmf3**=1000 (in Hz), **dfrq3**=1 (in MHz), **dof3**=0, **dpwr3**=0, **homo3**='n', **dseq3**=' ', and **dres3**=1.

See also: *Getting Started*

Related: **dn** Nucleus for first decoupler (P)
dfrq3 Transmitter frequency of third decoupler (P)
dof3 Frequency offset for third decoupler (C)
numrfch Number of rf channels (P)
tn Nucleus for observe transmitter (P)

dn4 Nucleus for fourth decoupler (P)

Applicability: Systems with a deuterium decoupler channel as the fourth decoupler.

Description: Changing the value of `dn4` causes a macro (named `_dn4`) to be executed that extracts values for `dfrq4` and `dof4` from lookup tables. Otherwise, `dn4` functions analogously to the parameters `tn` and `dn` except that the only valid value for `dn4` is 'H2'. If an experiment does not use the fourth decoupler channel, the channel can be disabled by setting `dn4=' '` (two single quotes with no space in between). This sets `dm4='n'`, `dmm4='c'`, `dmf4=1000` (in Hz), `dfrq4=1` (in MHz), `dof4=0`, `dpwr4=0`, `homo4='n'`, `dseq4=' '`, and `dres4=1`.

See also: *Getting Started*

Related: `dfrq4` Transmitter frequency of fourth decoupler (P)
`dn` Nucleus for first decoupler (P)
`dof4` Frequency offset for fourth decoupler (C)
`numrfch` Number of rf channels (P)
`tn` Nucleus for observe transmitter (P)

dnode Display list of valid limNET nodes (M,U)

Applicability: Systems with limNET.

Syntax: `dnode`

Description: Displays the contents of the user's limNET node database (i.e., all remote nodes available to limNET). Each node is listed by name, Ethernet address (6 hexadecimal bytes), and burst size

See also: *Getting Started*

Related: `eaddr` Display Ethernet address (M,U)

dnuc Retrieve nucleus table parameters for first decoupler (obsolete)

Description: This command is no longer part of VNMR. Use `setfrq` as the replacement.

Related: `setfrq` Set frequencies of rf channels in system (C)

dnuc2 Retrieve nucleus table parameters for second decoupler (obsolete)

Description: This command is no longer part of VNMR. Use `setfrq` as the replacement.

Related: `setfrq` Set frequencies of rf channels in system (C)

dnuc3 Retrieve nucleus table parameters for third decoupler (obsolete)

Description: This command is no longer part of VNMR. Use `setfrq` as the replacement.

Related: `setfrq` Set frequencies of rf channels in system (C)

doautodialog Start a dialog window using def file (M)

Applicability: Systems with automation.

Syntax: `doautodialog`

Description: Internal macro used by `enter` to start a dialog window using the `def` file for an experiment in the `dialoglib` directory.

Related: `enter` Enter sample information for automation run (M,U)

D

dodialog Start a dialog window with dialoglib file (M)

Syntax: `dodialog`

Description: Internal macro that starts a dialog window using a dialog file in the `dialoglib` directory.

doexpdialog Start a dialog window with glide/exp/experiment def file (M)

Syntax: `doexpdialog`

Description: Internal macro that starts a dialog window using a def file in the directory `/glide/exp/experiment`.

dof Frequency offset for first decoupler (P)

Description: Controls the frequency offset of the first decoupler. Higher numbers move the decoupler to higher frequency (toward the left side of the spectrum). The frequency accuracy of the decoupler offset is generally 0.0745 Hz on *GEMINI 2000* systems and 0.1 Hz on other systems. The value is specified in the `config` program.

Values: On *GEMINI 2000* systems, -50000 to 50000 Hz, in steps of 0.0745 Hz.
On systems other than the *GEMINI 2000*, -100000 to 100000 Hz (approximate, depends on frequency), in steps of 0.1 Hz.

See also: *Getting Started*

Related: `config` Display current configuration and possible change it (M)
`dof2` Frequency offset for second decoupler (P)
`dof3` Frequency offset for third decoupler (P)
`dof4` Frequency offset for fourth decoupler (P)
`tof` Frequency offset for observe transmitter (P)

dof2 Frequency offset for second decoupler (P)

Applicability: Systems with a second decoupler.

Description: Controls the frequency offset for the second decoupler. `dof2` functions analogously to the parameters `tof` and `dof`.

Values: -100000 to 100000 Hz (approximate, depends on frequency), in steps of 0.1 Hz.
If `dn2= ' '` (two single quotes with no space in between) and a second decoupler channel is present in the console, `dof2` assumes a default value of 0 when `go` is executed.

See also: *Getting Started*

Related: `dn2` Nucleus for second decoupler (P)
`dof` Frequency offset for first decoupler (P)
`tof` Frequency offset for observe transmitter (P)

dof3 Frequency offset for third decoupler (P)

Applicability: Systems with a third decoupler.

Description: Controls the frequency offset for the third decoupler. `dof3` functions analogously to the parameters `tof` and `dof`.

Values: -100000 to 100000 Hz (approximate, depends on frequency), in steps of 0.1 Hz.
If `dn3= ' '` (two single quotes with no space in between) and a third decoupler channel is present in the console, `dof3` assumes a default value of 0 when `go` is executed.

See also: *Getting Started*

Related: **dn3** Nucleus for third decoupler (P)
dof Frequency offset for first decoupler (P)
tof Frequency offset for observe transmitter (P)

dof4 Frequency offset for fourth decoupler (P)

Applicability: Systems with a deuterium decoupler channel as the fourth decoupler.

Description: Controls the frequency offset for the fourth decoupler. **dof4** functions analogously to the parameters **tof** and **dof**.

Values: -100000 to 100000 Hz (approximate, depends on frequency), in steps of 2.384 Hz. If **dn4**= ' ' (two single quotes with no space in between) and a fourth decoupler channel is present in the console, **dof4** assumes a default value of 0 when **go** is executed.

See also: *Getting Started*

Related: **dn4** Nucleus for fourth decoupler (P)
dof Frequency offset for first decoupler (P)
tof Frequency offset for observe transmitter (P)

Doneshot Set up parameters for Doneshot pulse sequence (M)

Syntax: Doneshot

Description: Converts a parameter set to Doneshot experiment.

See also: *User Guide: Liquids NMR*

Related: **dosy** Process DOSY experiments (M)
fiddle Perform reference deconvolution (M)
setup_dosy Set up gradient levels for DOSY experiments (M)

dopardialog Start a dialog with **dialoglib/experiment def file (M)**

Syntax: dopardialog

Description: Internal macro that starts a dialog window using a **def** file in the directory **dialoglib/experiment**.

do_pcsc Calculate proton chemical shifts spectrum (C)

Syntax: **do_pcsc**<(<threshold><,max_cc><,max_width) >

Description: Strips a high-resolution proton spectrum down to a list of chemical shifts. The list is saved in the file **pcsc.outpar**. If no argument is given, **do_pcsc** automatically calculates the threshold and uses default values for the maximum allowable coupling constant and the maximum width of a spin multiplet.

Arguments: **threshold** sets the level whether a point belongs to a peak or is noise.
max_cc is the maximum allowable coupling constant in the spectrum. Default is 20 Hz.
max_width is the maximum width of a spin multiplet in the spectrum. Default is 60 Hz.

Examples: **do_pcsc**
do_pcsc(10)
do_pcsc(9,20,80)

See also: *User Guide: Liquids*

Related: `pcss` Calculate and show proton chemical shifts spectrum (M)

dosy Process DOSY experiments (M)

Syntax: `dosy(<'prune'>, <lowerlimit, upperlimit>)`

Description: Performs a DOSY (diffusion ordered spectroscopy) analysis of the data in an array of spectra.

`dosy` uses the commands `dll` and `fp` to determine the heights of all signals above the threshold defined by the parameter `th` and then fits the decay curve for each signal to a Gaussian using the program `dosyfit`. It stores a summary of all diffusion coefficients and their estimated standard errors and various other results as follows:

- In the directory `$HOME/vnmrsys/Dosy`: `diffusion_display.inp`, `general_dosy_stats`, `calibrated_gradients`, `fit_errors`, and `diffusion_spectrum`
- In the current experiment: a second copy of `diffusion_display.inp`.

The command `showdosy` has been incorporated into `dosy`.

Arguments: `prune` starts a dialog to allow one or more spectra to be omitted from the analysis.

`lowerlimit` is the lower diffusion limit (in units of 10^{-10} m²/s) to be displayed.

`upperlimit` is the upper diffusion limit (in units of 10^{-10} m²/s) to be displayed.

Without arguments, `dosy` uses all the experimental spectra and covers the whole diffusion range seen in the experimental peaks.

See also: *User Guide: Liquids NMR*

Related: `ddif` Synthesize and display DOSY plot (C)
`fiddle` Perform reference deconvolution (M)
`setup_dosy` Set up gradient levels for DOSY experiments (M)

dosyfrq Larmor frequency of phase encoded nucleus in DOSY (P)

Syntax: `dosyfrq`

Description: Stores the NMR frequency of the phase encoded nucleus in DOSY experiments. It is directly set by the DOSY sequences.

See also: *User Guide: Liquids NMR*

Related: `dosy` Process DOSY experiments (M)

dosygamma Gyromagnetic constant of phase encoded nucleus in DOSY (P)

Syntax: `dosygamma`

Description: Stores the gyromagnetic constant of the phase encoded nucleus in DOSY experiments. It is automatically set by the DOSY sequences and used by the `dosy` macro.

See also: *User Guide: Liquids NMR*

Related: `dosy` Process DOSY experiments (M)

dosytimecubed Gyromagnetic constant of phase encoded nucleus in DOSY (P)

Syntax: `dosytimecubed`

Description: Timecubed factor in the expression for diffusional attenuation. It is automatically set by the DOSY sequences and used by the `dosy` macro.

See also: *User Guide: Liquids NMR*

Related: `dosy` Process DOSY experiments (M)

dot1 Set up a T_1 experiment (M)

Syntax: `dot1<(min_T1_estimate,max_T1_estimate,time)>`

Description: Sets up all parameters to perform a T_1 experiment, including `d1`, `pw`, `p1`, `nt`, and an array of `d2` values, based on information entered you enter. Make sure that the parameter `pw90` is set properly and contains the correctly calibrated 90° pulse width because `dot1` uses this information. If you have not done a pulse width calibration recently, you may wish to do so now.

Minimum and maximum T_1 for the peaks of interest are estimates. Do the best you can. Your estimates are used to select optimum values of `d2`. If the T_1 does not fall between your two guesses, your experiment may not be optimum, but it should still be usable unless your estimates are extremely far off. When you are satisfied with the parameters, enter `ga` or `au` to acquire the data.

Arguments: `min_T1_estimate` is the estimated minimum expected T_1 . The default is the system prompts the user for the value.

`max_T1_estimate` is the estimated maximum expected T_1 . The default is the system prompts the user for the value.

`time` is the total time in hours that the experiment should take. The default is the system prompts the user for the value.

Examples: `dot1`
`dot1(1,2,.5)`

See also: *User Guide: Liquids NMR*

Related: `d1` First delay (P)
`d2` Incremented delay in 1st indirectly detected dimension (P)
`ga` Submit experiment to acquisition and FT the result (C)
`go` Submit experiment to acquisition (C)
`nt` Number of transients (P)
`p1` First pulse width (P)
`pw` Pulse width (P)
`pw90` 90° pulse width (P)

dotflag Display FID as connected dots (P)

Description: When sparse FID data points are displayed, they are displayed as unconnected dots. If `dotflag` exists and is set to 'n', the FID dots will be connected. To create `dotflag`, enter `create('dotflag','flag')`. To create `dotflag` and the FID display parameters `axisf`, `vpf`, `vpfi`, `crf`, and `deltaf` (if the parameter set is older and lacks these parameters), enter `addpar('fid')`.

Values: 'n' sets connecting the dots. 'y' sets not connecting the dots.

See also: *Getting Started*

Related: `addpar` Add selected parameters to the current experiment (M)

`create` Create new parameter in a parameter tree (C)
`df` Display a single FID (C)

`downsamp` **Downsampling factor applied after digital filtering (P)**

Description: Specifies the downsampling factor applied after digital filtering. The spectral width of the data set after digital filtering and downsampling is `sw` divided by `downsamp`, where `sw` is the acquired spectral width. If `downsamp` does not exist in the current experiment, enter `addpar('downsamp')` to add it. `addpar('downsamp')` creates the digital filtering and downsampling parameters `downsamp`, `dscoef`, `dsfb`, `dsfsfrq`, and `filtfile`.

Values: Number for the downsampling factor. 1 sets digital filtering with a filter bandwidth specified by `dsfb` without downsampling.

'n' sets normal data processing in VNMR without digital filtering.

See also: *Getting Started*

Related: `addpar` Add selected parameters to current experiment (M)
`digfilt` Write digitally filtered FID to another experiment (M)
`dscoef` Digital filter coefficients for downsampling (P)
`dsfb` Digital filter bandwidth for downsampling (P)
`dsfsfrq` Bandpass filter offset for downsampling (P)
`filtfile` File of FIR digital filter coefficients (P)
`pards` Create additional parameters used by downsampling (M)
`sw` Spectral width in directly detected dimension (P)

`dp` **Double precision (P)**

Description: Sets whether data are acquired in a 16-bit or 32-bit integer format.

Values: 'n' sets 16-bit format, 'y' sets 32-bit format. If the 200-kHz receiver option is installed (Max. Narrowband Width set to 200 kHz in the CONFIG window), `dp` is forced to 'n' if $120000 < sw \leq 200000$. If $sw > 200000$, `dp` is forced to 'y'. On wideline systems, `dp='y'` is required when $sw > 100000$. On *MERCURY-VX* and *MERCURY* systems, `dp='y'` only.

See also: *Getting Started*

Related: `sw` Spectral width in directly detected dimension (P)

`dpcon` **Display plotted contours (C)**

Syntax: `dpcon(<options,><levels,spacing>)`

Description: Produces a true contour plot display.

Arguments: `options` must precede `levels` and `spacing` in the argument list and can be one or more of the following:

- 'pos' is a keyword to limit the display to positive peaks only in phased spectra. The default is both positive and negative peaks.
- 'neg' is a keyword to limit the display to negative peaks only in phased spectra.
- 'noaxis' is a keyword to omit outlining the display and drawing the horizontal or vertical axis.

`levels` is the maximum number of contours to be shown. The default is 4.

`spacing` is the spacing by relative intensity of successive contour levels. The default is 2.

Examples: `dpcon`
`dpcon('pos', 6)`
`dpcon(15, 1.4)`

See also: *User Guide: Liquids NMR*

Related: `dcon` Display noninteractive color intensity map (C)
`dconi` Control display selection for the `dconi` program (P)
`dpconn` Display plotted contours without screen erase (C)
`pcon` Plot contours on plotter (C)

dpconn Display plotted contours without screen erase (C)

Syntax: `dpconn(<options>, <levels>, <spacing>)`

Description: Produces a true contour plot display exactly the same as the `dpcon` command, but without erasing the screen before drawing. The arguments are entered the same as `dpcon`.

See also: *User Guide: Liquids NMR*

Related: `dpcon` Display plotted contours (C)

dpf Display peak frequencies over spectrum (C)

Syntax: (1) `dpf(<'noll'><,<'pos'><,<noise_mult><,<'top'>>>`
(2) `dpf(<'noll'><,<'pos'><,<noise_mult><,<'leader'><`
`<,<length>>>`

Description: Displays peak frequencies in the graphics window, with units specified by the `axis` parameter. Only those peaks greater than `th` high are selected. If the interactive command `ds` is active, `dpf` deactivates it.

Two basic modes of label positioning are available: labels placed at the top, with *long leaders* extending down to the tops of the lines (syntax 1 using `'top'` keyword) or labels positioned just above each peak, with *short leaders* (syntax 2 using `'leader'` keyword). The default is short leaders.

Arguments: `'noll'` is a keyword to display frequencies using last previous line listing.

`'pos'` (or `'noneg'`) is a keyword to display positive peaks only.

`noise_mult` is a numerical value that determines the number of noise peaks displayed for broad, noisy peaks. The default is 3. A smaller value results in more peaks, a larger value results in fewer peaks, and a value of 0.0 results in a line listing containing all peaks above the threshold `th`. Negative values of `noise_mult` are changed to a value of 3. The `noise_mult` argument is inactive when the `'noll'` keyword is specified.

`'top'` is a keyword to display peak labels at the top with long leaders. In this mode, the height of labels is varied by changing the parameter `wc2`.

`'leader'` is a keyword to display labels positioned just above each peak.

`length` specifies the leader length, in mm, if labels are positioned just above each peak. The default is 20.

Examples: `dpf('pos')`
`dpf('leader', 30)`
`dpf('top', 'noll')`
`dpf('pos', 0.0, 'leader', 30)`

See also: *Getting Started*

Related: `axis` Axis label for displays and plots (P)
`dpir` Display integral amplitudes below spectrum (C)

<code>dpirn</code>	Display normalized integral amplitudes below spectrum (M)
<code>pir</code>	Plot integral amplitudes below spectrum (C)
<code>pirn</code>	Plot normalized integral amplitudes below spectrum (M)
<code>ppf</code>	Plot peak frequencies over spectrum (M)
<code>th</code>	Threshold (P)
<code>vp</code>	Vertical position of spectrum (P)
<code>wc2</code>	Width of chart in second direction (P)

dpir **Display integral amplitudes below spectrum (C)**

Syntax: `dpir`

Description: Displays integral amplitudes below the appropriate spectral regions.

See also: *Getting Started*

Related:	<code>dpf</code>	Display peak frequencies over spectrum (C)
	<code>dpirn</code>	Display normalized integral amplitudes below spectrum (M)
	<code>pir</code>	Plot integral amplitudes below spectrum (C)
	<code>pirn</code>	Plot normalized integral amplitudes below spectrum (M)
	<code>ppf</code>	Plot peak frequencies over spectrum (M)

dpirn **Display normalized integral amplitudes below spectrum (M)**

Syntax: `dpirn`

Description: Equivalent to the command `dpir` except that the sum of the integrals is normalized to the value of the parameter `ins`.

See also: *Getting Started*

Related:	<code>dpir</code>	Display integral amplitudes below spectrum (C)
	<code>ins</code>	Integral normalization scale (P)
	<code>pirn</code>	Plot normalized integral amplitudes below spectrum (M)

dpl **Default plot (M)**

Syntax: `dpl`

Description: Looks for sequence-specific default plot macro (`dpl_seqfil`) and executes if one is found.

Related:	<code>dpl_seqfil</code>	Sequence-specific default plot (M)
	<code>dpr</code>	Default process (M)
	<code>dds</code>	Default display (M)

dpl_seqfil **Sequence-specific default plot (M)**

Syntax: `dpl_seqfil`

Description: Sequence-specific default plot. These macros are called by the `dpl` macro.

Examples: `dpl_NOESY1D`
`dpl_TOCSY1D`

Related:	<code>dpl</code>	Default plot (M)
	<code>dpr</code>	Default process (M)
	<code>dds</code>	Default display (M)

dplane	Display a 3D plane (M)																
Applicability:	All systems; however, although <code>dplane</code> is available on <i>MERCURY-VX</i> , <i>MERCURY</i> , and <i>GEMINI 2000</i> , such systems can only process 3D data and cannot acquire 3D data.																
Syntax:	<code>dplane (<plane_type , >plane_number)</code>																
Description:	Displays the 2D color map of a particular data plane from a 3D spectral data set. The 3D parameters are loaded into VNMR each time <code>dplane</code> is executed. The parameter <code>path3d</code> specifies the absolute path to the directory (without the <code>.extr</code> file extension) where the 2D planes extracted from the 3D spectral data set reside.																
Arguments:	<code>plane_type</code> is one of the keywords 'f1f3', 'f2f3', and 'f1f2' for the f ₁ f ₃ , f ₂ f ₃ , and f ₁ f ₂ planes, respectively. If <code>plane_type</code> is specified, the parameter <code>plane</code> is updated with that new value. <code>plane</code> is then used to determine the type of 3D plane to be displayed. <code>plane_number</code> specifies which plane of a particular type is to be displayed: <ul style="list-style-type: none"> • For plane f₁f₃, the range of <code>plane_number</code> is 1 to <code>fn2/2</code> • For plane f₂f₃, the range of <code>plane_number</code> is 1 to <code>fn1/2</code> • For plane f₁f₂, the range of <code>plane_number</code> is 1 to <code>fn/2</code> 																
Examples:	<code>dplane (3)</code> <code>dplane (' f1f2 ' , 2)</code>																
See also:	<i>User Guide: Liquids NMR</i>																
Related:	<table> <tr> <td><code>dsplanes</code></td> <td>Display a series of 3D planes (M)</td> </tr> <tr> <td><code>dproj</code></td> <td>Display a 3D plane projection (M)</td> </tr> <tr> <td><code>getplane</code></td> <td>Extract planes from a 3D spectral data set (M)</td> </tr> <tr> <td><code>nextpl</code></td> <td>Display the next 3D plane (M)</td> </tr> <tr> <td><code>path3d</code></td> <td>Path to currently displayed 2D planes from a 3D data set (P)</td> </tr> <tr> <td><code>plane</code></td> <td>Currently displayed 3D plane type (P)</td> </tr> <tr> <td><code>prevpl</code></td> <td>Display the previous 3D plane (M)</td> </tr> <tr> <td><code>plplanes</code></td> <td>Plot a series of 3D planes (M)</td> </tr> </table>	<code>dsplanes</code>	Display a series of 3D planes (M)	<code>dproj</code>	Display a 3D plane projection (M)	<code>getplane</code>	Extract planes from a 3D spectral data set (M)	<code>nextpl</code>	Display the next 3D plane (M)	<code>path3d</code>	Path to currently displayed 2D planes from a 3D data set (P)	<code>plane</code>	Currently displayed 3D plane type (P)	<code>prevpl</code>	Display the previous 3D plane (M)	<code>plplanes</code>	Plot a series of 3D planes (M)
<code>dsplanes</code>	Display a series of 3D planes (M)																
<code>dproj</code>	Display a 3D plane projection (M)																
<code>getplane</code>	Extract planes from a 3D spectral data set (M)																
<code>nextpl</code>	Display the next 3D plane (M)																
<code>path3d</code>	Path to currently displayed 2D planes from a 3D data set (P)																
<code>plane</code>	Currently displayed 3D plane type (P)																
<code>prevpl</code>	Display the previous 3D plane (M)																
<code>plplanes</code>	Plot a series of 3D planes (M)																

dpr	Default process (M)						
Syntax:	<code>dpr</code>						
Description:	Looks for sequence-specific default plot macro (<code>dpr_seqfil</code>) and executes if one is found.						
Related:	<table> <tr> <td><code>dpr_seqfil</code></td> <td>Sequence-specific default process (M)</td> </tr> <tr> <td><code>dpl</code></td> <td>Default plot (M)</td> </tr> <tr> <td><code>dds</code></td> <td>Default display (M)</td> </tr> </table>	<code>dpr_seqfil</code>	Sequence-specific default process (M)	<code>dpl</code>	Default plot (M)	<code>dds</code>	Default display (M)
<code>dpr_seqfil</code>	Sequence-specific default process (M)						
<code>dpl</code>	Default plot (M)						
<code>dds</code>	Default display (M)						
dpr_seqfil	Sequence-specific default process (M)						
Syntax:	<code>dpr_seqfil</code>						
Description:	Sequence-specific default plot. These macros are called by the <code>dpr</code> macro.						
Examples:	<code>dpr_NOESY1D</code> <code>dpr_TOCSY1D</code>						
Related:	<table> <tr> <td><code>dpr</code></td> <td>Default process (M)</td> </tr> <tr> <td><code>dpl</code></td> <td>Default plot (M)</td> </tr> <tr> <td><code>dds</code></td> <td>Default display (M)</td> </tr> </table>	<code>dpr</code>	Default process (M)	<code>dpl</code>	Default plot (M)	<code>dds</code>	Default display (M)
<code>dpr</code>	Default process (M)						
<code>dpl</code>	Default plot (M)						
<code>dds</code>	Default display (M)						

dprofile **Display pulse excitation profile (M)**

Syntax: `dprofile<(axisflag<,profile<,shapefile>>>>`

Description: Displays the X, Y and Z excitation (inversion) profile for a pulse shape generated by the Pbox software. If `shapefile` is not provided, the last simulation data stored in the `shapelib/pbox.sim` file are displayed.

Arguments: The `axisflag` and `profile` arguments can be given in any order.

`axisflag` is 'y' to display the full spectrum and a frequency scale, or 'n' to suppress the scale and spectrum. The default is 'n'.

`profile` is a character string identifying the desired profile. 'xyz' selects X, Y, and Z (inversion) profiles; 'xy' selects only the excitation (transverse) profiles; 'x' selects only the X transverse excitation profile; and 'z' selects only the inversion profile. The default is 'xyz'.

`shapefile` is the name of a *.RF or *.DEC file, including the extension.

Examples: `dprofile`
`dprofile('y','xy')`
`dprofile('xy','n','softpls.RF')`

See also: *User Guide: Liquids NMR*

Related: `pprofile` Plot pulse excitation profile (M)
`Pbox` Pulse shaping software (U)

dproj **Display a 3D plane projection (M)**

Applicability: All systems; however, although `dproj` is available on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*, such systems can only process 3D data and cannot acquire 3D data.

Syntax: `dproj<(plane_type)>`

Description: Displays 2D color map of the 2D projection plane from a 3D spectral data set. The projection is a skyline projection. The 3D parameters are loaded into VNMR each time `dproj` is executed. For this macro, the parameter `path3d` specifies the directory (without the `.extr` extension) where the 2D projection resides that has been created from the 3D spectral data set.

Arguments: `plane_type` is one of the keywords 'f1f3', 'f2f3', and 'f1f2' for the f₁f₃, f₂f₃, and f₁f₂ planes, respectively. If `plane_type` is specified, the parameter `plane` is updated with that value. `plane` is then used to determine the type of 2D projection to be displayed.

Examples: `dproj`
`dproj('f1f2')`

See also: *User Guide: Liquids NMR*

Related: `dplane` Display a 3D plane (M)
`dsplanes` Display a series of 3D planes (M)
`getplane` Extract planes from a 3D spectral data set (M)
`nextpl` Display the next 3D plane (M)
`path3d` Path to currently displayed 2D planes from a 3D data set (P)
`plane` Currently displayed 3D plane type (P)
`plplanes` Plot a series of 3D planes (M)
`prevpl` Display the previous 3D plane (M)

dps **Display pulse sequence (C)**

Syntax: `dps<(file),x,y,width,height>`

Description: Displays a picture of pulse sequences consisting of three to five parts. The top part is the transmitter pulse sequence (Tx). The second part is the decoupler pulse sequence (Dec). The third part might be the second or third decoupler (Dec2 or Dec3) pulse sequence or gradients (X, Y, or Z), depending on the program. The lowest part is the status.

The pulse parameters are displayed if there is enough space and if the length of the parameter name is less than thirty letters. The value of each pulse is also displayed. If the value delay or width is less than zero, a question mark (?) is displayed. The time units are displayed in color (on a color monitor). The height of pulses is scaled according to their power level.

`dps` also displays spin lock, transmitter gating, observe transmitter power, and other information.

Arguments: `file` specifies the name of the file containing the pulse sequences. The default is the file `seqfil`.

`x, y` specifies the start of the position with respect to the lower-left corner of the window.

`width, height` are in proportion to `wcmax` and `wc2max`.

See also: *Getting Started*

Related:	<code>pps</code>	Plot pulse sequence (C)
	<code>seqfil</code>	Pulse sequence name (P)
	<code>wc</code>	Width of chart (P)
	<code>wcmax</code>	Maximum width of chart (P)
	<code>wc2max</code>	Maximum width of chart in second direction (P)

`dpwr` Power level for first decoupler with linear amplifier (P)

Applicability: Systems with a linear amplifier.

Description: On *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* broadband systems equipped with a linear amplifier, a 63-dB attenuator between the decoupler transmitter board and the amplifier controls the power level. On systems other than *GEMINI 2000*, the configuration is the same except the attenuator can be 63 dB or 79 dB.

The system value for the attenuator upper safety limit is set in the CONFIG window (opened by `config`). For the *GEMINI 2000*, the label Max. Decoupler in the CONFIG window sets this value; for other systems, Upper Limit sets it. For broadband decoupling of ^1H nuclei, typical values range from 36 to 49 dB. For homonuclear decoupling, typical values range from 5 to 15 dB.

Values: On *MERCURY* systems, 0 to 63 dB, in steps of 1 dB.

GEMINI 2000 systems, 0 to 63 (63 is maximum power), steps of 0.5 dB.

On systems other than *GEMINI 2000*: 0 to 63 (63 is maximum power), in units of dB, if the 63-dB attenuator is installed or -16 to 63 (63 is maximum power), in units of dB, if the 79-dB attenuator is installed.

CAUTION: **Decoupler power greater than 2 watts in a switchable probe will damage the probe. Always carefully calibrate decoupling to avoid exceeding 2 watts. The maximum value for `dpwr` on a 200-, 300-, or 400-MHz system with a linear amplifier on the decoupler channel has been set to 49, corresponding to about 2 watts of power. Before using `dpwr=49` for continuous decoupling, ensure safe operation by measuring the output power. This should be done during system installation and checked periodically by the user.**

See also: *Getting Started*

Related:	<code>cattn</code>	Coarse attenuator (P)
	<code>config</code>	Display current configuration and possible change it (M)
	<code>dpwrf</code>	First decoupler fine power (P)
	<code>dpwr2</code>	Power level for second decoupler (P)
	<code>dpwr3</code>	Power level for third decoupler (P)
	<code>dpwr4</code>	Power level for fourth decoupler (P)
	<code>fattn</code>	Fine attenuator (P)
	<code>tpwr</code>	Power level of observe transmitter with linear amplifiers (P)
	<code>tpwrf</code>	Observe transmitter fine power (P)

`dpwr2` Power level for second decoupler with linear amplifier (P)

Applicability: Systems with a linear amplifier as the second decoupler.

Description: Controls the coarse attenuator (63 dB or 79 dB) that resides between the transmitter board and the linear amplifier associated with the second decoupler. The system value for the attenuator upper safety limit is set in the CONFIG window (opened by `config`).

Values: If 63-dB attenuator installed: 0 to 63 (63 is max. power), in units of dB. If 79-dB attenuator installed: -16 to 63 (63 is max. power), in units of dB. If `dn2= ' '` (two single quotes) and a second decoupler channel is present in the console, `dpwr2` assumes a default value of 0 when `go` is executed.

CAUTION: Decoupler power greater than 2 watts in a switchable probe will damage the probe. Always carefully calibrate decoupling to avoid exceeding 2 watts. The maximum value for `dpwr2` on a 200-, 300-, or 400-MHz system with a linear amplifier on the decoupler channel has been set to 49, corresponding to about 2 watts of power. Before using `dpwr2=49` for continuous decoupling, ensure safe operation by measuring the output power. This should be done during system installation and checked periodically by the user.

See also: *Getting Started*

Related:	<code>cattn</code>	Coarse attenuator type (P)
	<code>config</code>	Display current configuration and possible change it (M)
	<code>dn2</code>	Nucleus for second decoupler (P)

`dpwr3` Power level for third decoupler with linear amplifier (P)

Applicability: Systems with a linear amplifier as the third decoupler.

Description: Controls the coarse attenuator (63 dB or 79 dB) that resides between the transmitter board and the linear amplifier associated with the third decoupler. The system value for the attenuator upper safety limit is set in the CONFIG window (opened by `config`).

Values: If 63-dB attenuator installed: 0 to 63 (63 is max. power), in units of dB. If 79-dB attenuator installed: -16 to 63 (63 is max. power), in units of dB. If `dn3= ' '` (two single quotes) and a third decoupler channel is present in the console, `dpwr3` assumes a default value of 0 when `go` is executed.

CAUTION: Decoupler power greater than 2 watts in a switchable probe will damage the probe. Always carefully calibrate decoupling to avoid exceeding 2 watts. The maximum value for `dpwr3` on a 200-, 300-, or 400-MHz system with a linear amplifier on the decoupler channel has been set to 49, corresponding to about 2 watts of power. Before using `dpwr3=49` for continuous decoupling, ensure safe operation by

measuring the output power. This should be done during system installation and checked periodically by the user.

See also: *Getting Started*

Related: `cattn` Coarse attenuator type (P)
`config` Display current configuration and possible change it (M)
`dn3` Nucleus for third decoupler (P)

`dpwr4` Power level for fourth decoupler amplifier (P)

Applicability: Systems with deuterium decoupler channel as the fourth decoupler.

Description: Controls the coarse attenuator (45 dB range) that resides on the Lock Transceiver board and the amplifier associated with the fourth decoupler. The system value for the attenuator upper safety limit is set in the CONFIG window (opened by `config`).

Values: 48-dB attenuator: 15 to 63 (63 is max. power), in units of dB.
 If `dn4= ' '` (two single quotes) and a third decoupler channel is present in the console, `dpwr4` assumes a default value of 0 when `go` is executed.

CAUTION: Decoupling power greater than 5 watts applied to a triple-resonance probe will damage the probe. The maximum value for `dpwr4` is 63, corresponding to about 35 watts to the probe. A value of `dpwr4` equal to 52 corresponds to about 5 watts and will produce approximately a 1 kHz decoupling field. Always carefully calibrate decoupling power to avoid exceeding 5 watts. Before using `dpwr4=52` continuous decoupling, ensure safe operation by measuring the output power. Measurement should be taken during system installation and checked periodically by the user.

See also: *Getting Started*

Related: `cattn` Coarse attenuator type (P)
`config` Display current configuration and possible change it (M)
`dn3` Nucleus for third decoupler (P)

`dpwrf` First decoupler fine power (P)

Applicability: Systems with an optional fine attenuator on the decoupler channel.

Description: Controls the first decouple fine attenuator on ^{UNITY}INOVA and UNITYplus systems, on solids systems, or on UNITY systems where an optional second attenuator is in series with the standard attenuator on the decouple channel. Systems with this attenuator are designated within the CONFIG window (opened by `config`) by the status of the Fine Attenuator entry. The fine attenuator is linear and spans 60 dB (^{UNITY}INOVA or UNITYplus) or 6 dB (other systems).

Values: 0 to 4095 (where 4095 is maximum power). If `dpwrf` does not exist in the parameter table, a value of 4095 is assumed.

See also: *VNMR User Programming*

Related: `config` Display current configuration and possibly change it (M)
`dpwr` Power level for first decoupler with linear amplifiers (P)
`dpwrf2` Second decoupler fine power (P)
`dpwrf3` Third decoupler fine power (P)
`dpwrm` First decoupler linear modulator power (P)
`fattn` Fine attenuator (P)
`tpwr` Power level of observe transmitter with linear amplifiers (P)
`tpwrf` Transmitter fine power (P)

dpwrf2 Second decoupler fine power (P)

Applicability: Systems with an optional fine attenuator on the second decoupler channel.

Description: Controls the second decoupler fine attenuator, functioning analogously to [dpwrf](#).

Values: 0 to 4095 (where 4095 is maximum power). If [dpwrf2](#) does not exist in the parameter table, a value of 4095 is assumed.

See also: *VNMR User Programming*

Related: [dpwrf](#) First decoupler fine power (P)

dpwrf3 Third decoupler fine power (P)

Applicability: Systems with an optional fine attenuator on the third decoupler channel.

Description: Controls the third decoupler fine attenuator, functioning analogously to [dpwrf](#).

Values: 0 to 4095 (where 4095 is maximum power). If [dpwrf3](#) does not exist in the parameter table, a value of 4095 is assumed.

See also: *VNMR User Programming*

Related: [dpwrf](#) First decoupler fine power (P)

dpwrm First decoupler linear modulator power (P)

Applicability: ^{UNITY}*INOVA* or *UNITYplus* systems with a first decoupler linear modulator.

Description: Controls the first decoupler linear modulator on *UNITYplus* systems.

Values: 0 to 4095 (where 4095 is maximum power). If [dpwrm](#) does not exist in the parameter table, a value of 4095 is assumed.

See also: *VNMR User Programming*

Related: [dpwrm2](#) Second decoupler linear modulator power (P)
[dpwrm3](#) Third decoupler linear modulator power (P)

dpwrm2 Second decoupler linear modulator power (P)

Applicability: ^{UNITY}*INOVA* or *UNITYplus* systems with a second decoupler linear modulator.

Description: Controls the second decoupler linear modulator on *UNITYplus* systems.

Values: 0 to 4095 (where 4095 is maximum power). If [dpwrm2](#) does not exist in the parameter table, a value of 4095 is assumed.

See also: *VNMR User Programming*

Related: [dpwrm](#) First decoupler linear modulator power (P)

dpwrm3 Third decoupler linear modulator power (P)

Applicability: ^{UNITY}*INOVA* or *UNITYplus* systems with a third decoupler linear modulator.

Description: Controls the third decoupler linear modulator on *UNITYplus* systems.

Values: 0 to 4095 (where 4095 is maximum power). If [dpwrm3](#) does not exist in the parameter table, a value of 4095 is assumed.

See also: *VNMR User Programming*

Related: [dpwrm](#) First decoupler linear modulator power (P)

dqcosy **Set up parameters for double-quantum filtered COSY (M)**Syntax: `dqcosy`

Description: Macro to set up a double-quantum filtered COSY (homonuclear correlation) experiment.

Alternate: DQCOSY button in the 2D Pulse Sequence Setup Secondary Menu.

See also: *User Guide: Liquids NMR*Related: `cosyps` Set up parameters for phase-sensitive COSY (M)
`relayh` Set up parameters for COSY pulse sequence (M)**DQCOSY** **Change parameters for DQCOSY experiment (M)**Syntax: `DQCOSY<('GLIDE')>`

Description: Converts the current parameter set to a DQCOSY experiment.

Arguments: 'GLIDE' is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding carbon spectrum for the experiment.**draw** **Draw line from current location to another location (C)**Syntax: `draw(<'keywords'>x,y)`

Description: Draws a line from the current location to the absolute location with coordinates given by the arguments.

Arguments: 'keywords' identifies the output device ('graphics' | 'plotter'), drawing mode ('xor' | 'normal'), and drawing capability ('newovly' | 'ovly' | 'ovlyC').

- 'graphics' | 'plotter' is a keyword for the output device. The default is 'plotter'. The output selected is passed to subsequent `pen`, `move`, or `draw` commands and remains active until a different output is specified.
- 'xor', 'normal' is a keyword for the drawing mode when using the 'graphics' output device. The default is 'normal'. In the 'xor' mode, if a line is drawn such that one or more points of the line are in common with a previous 'xor' line, the common points are erased. In the normal mode, the common points remain. The mode selected is passed to subsequent `draw`, `pen`, and `move` commands and remains active until a different mode is specified.
- 'newovly', 'ovly', and 'ovlyC' are keywords that specify an interactive drawing capability that is slightly slower than the 'xor' mode but more consistent in color. 'newovly' clears any previous draws, boxes, and writes made with the 'ovly' modes and draws the figure. 'ovly' draws without clearing so that multisegment figures can be created. 'ovlyC' clears without drawing.

`x`, `y` are the absolute coordinates, in mm, of the endpoint of the line to be drawn. The range of `x` is 0 at the left edge of the chart and `wcmax` at the right edge. The range of `y` is `-20` at the bottom of the chart and `wc2max` at the top.

Examples: `draw('graphics', 'xor'.wcmax-sc, vp+th)`
`draw(wcmax-sc-wc*(cr-delta-sp)/wp, wc2max)`

See also: *Getting Started*

Related: `gin` Return current mouse position and button values (C)
`move` Move to an absolute location (C)
`pen` Select a pen or color for drawing (C)

`wcmax` Maximum width of chart (P)
`wc2max` Maximum width of chart in second direction (P))

drawslice Display target slices (M)

Applicability: Systems with imaging capabilities.

Syntax: `drawslice`

Description: Displays target slices defined by the file `curexp+ '/mark2d.out'`. The program shows graphically the position and orientation of the selected target slices on a scout image. This macro is also called by the Show Target button in the slice planner menu. See the `plan` macro for more details.

See also: *User Guide: Imaging*

Related: `curexp` Current experiment directory (P)
`drawvox` Display target voxels (M)
`plan` Display menu for planning a target scan (M)
`ssplan` Set slice parameters for target slice (M)
`voxplan` Set voxel parameters for voxel defined by 2D box cursor (M)

drawvox Display target voxels (M)

Applicability: Systems with imaging capabilities.

Syntax: `drawvox`

Description: Displays target voxels defined by the file `curexp+ '/mark2d.out'`. This program shows graphically the position of the selected target voxels on the scout image. The user can plan and then display more than one voxel with this macro. This macro is also called by the Show Target button in the voxel planner menu. See the `plan` macro for more details.

See also: *User Guide: Imaging*

Related: `curexp` Current experiment directory (P)
`drawslice` Display target slices (M)
`plan` Display menu for planning a target scan (M)
`planlock` Planner lock out (P)
`ssplan` Set slice parameters for target slice (M)
`voxplan` Set voxel parameters for voxel defined by 2D box cursor (M)

dres Measure linewidth and digital resolution (C)

Syntax: `dres(<freq,<fractional_height>>)>`
`:linewidth,digital_resolution`

Description: Analyzes the line defined by the current cursor position for its linewidth (width at half-height) and digital resolution.

Arguments: `freq` is the frequency of the line. The default is the parameter `cr`. This overrides using the current cursor position as the frequency.

`fractional_height` is the linewidth is measured at this height.

`linewidth` is the value returned for the linewidth of the line.

`digital_resolution` is the value returned for the digital resolution of the line.

Examples: `dres:$width,$res`
`dres(cr,0.55)`

See also: *Getting Started; VNMR User Programming*

Related: `cr` Current cursor position (P)
`dsn` Measure signal-to-noise (C)

dres Tip-angle resolution for first decoupler (P)

Applicability: Systems with waveform generators.

Description: Controls the tip-angle resolution to be used within a waveform generator decoupling sequence on the first decoupler. The optimum value is a function of the decoupling sequence to be used: for WALTZ-16, `dres=90.0`; for MLEV16-240, `dres=30.0`; and for GARP1, `dres=1.0`.

Values: 1.0 to 90.0, in units of degrees. In reality, `dres` can assume values as small of 0.7 (but no smaller) and can be specified in units of 0.1°. To use this capability, change the limits of `dres` by using `destroy('dres')`
`create('dres','real')` `setlimit('dres',360,0.7,0.1)`.
 Making corresponding changes within the `fixpar` macro ensures that `dres` is created in the desired way with each new parameter set.

See also: *Getting Started*

Related: `dmfadj` Adjust decoupler tip-angle resolution time (M)
`dres2` Tip angle resolution for second decoupler (P)
`dres3` Tip angle resolution for third decoupler (P)
`fixpar` Correct parameter characteristics in experiment (M)

dres2 Tip-angle resolution for second decoupler (P)

Applicability: Systems with waveform generators.

Description: Controls the tip-angle resolution to be used within a waveform generator decoupling sequence on the second decoupler. The optimum value is a function of the decoupling sequence to be used: for WALTZ-16, `dres2=90.0`; for MLEV16-240, `dres2=30.0`; and for GARP1, `dres2=1.0`.

Values: 1.0 to 90.0, in units of degrees.

See also: *Getting Started*

Related: `dmf2adj` Adjust second decoupler tip-angle resolution time (M)
`dres` Tip-angle resolution for first decoupler (P)

dres3 Tip-angle resolution for third decoupler (P)

Applicability: Systems with waveform generators.

Description: Controls the tip-angle resolution to be used within a waveform generator decoupling sequence on the third decoupler. The optimum value is a function of the decoupling sequence to be used: for WALTZ-16, `dres3=90.0`; for MLEV16-240, `dres3=30.0`; and for GARP1, `dres3=1.0`.

Values: 1.0 to 90.0, in units of degrees.

See also: *Getting Started*

Related: `dmf3adj` Adjust third decoupler tip-angle resolution time (M)
`dres` Tip-angle resolution for first decoupler (P)

dres4 Tip-angle resolution for fourth decoupler (P)

Applicability: Systems with deuterium decoupler channel as the fourth decoupler.

Description: Controls the tip-angle resolution to be used for the decoupling sequence on the fourth decoupler. The optimum value is a function of the decoupling sequence to be used: for WALTZ-16, `dres4=90.0`; for MLEV16-240, `dres4=30.0`; and for GARP1, `dres4=1.0`.

Values: 1.0 to 90.0, in units of degrees.

See also: *Getting Started*

Related: `dmf4adj` Adjust fourth decoupler tip-angle resolution time (M)
`dres` Tip-angle resolution for first decoupler (P)

ds Display a spectrum (C)

Syntax: (1) `ds<(index)>`
 (2) `ds<(options)>`

Description: Displays a single spectrum. Parameter `intmod` controls integral display:

- `intmod= 'off'` turns off the integral display
- `intmod= 'full'` displays the entire integral
- `intmod= 'partial'` displays every other integral region

Parameter entry after a spectrum has been displayed with the `ds` command causes the spectrum to be updated.

Two additional parameters control the behavior of the `ds` command:

- The parameter `phasing` (in the “global” parameter set) controls the percentage of the spectrum updated during interactive phasing. This parameter can be set in the range of 10 to 100. A value of 100 causes the entire spectrum to be updated. A value of 20 causes the area between the two horizontal cursors to be updated.
- The parameter `lvltlt` (in the “current” parameter set) controls the sensitivity of the interactive `lv1` and `tlt` adjustments. `lvltlt` can be set to any positive real number. It is basically a multiplier for the sensitivity. The default value is 1.0. Larger values make the adjustments larger. Smaller values make the adjustments smaller.

For arrayed 1D spectra or for 2D spectra, a particular trace can be viewed by supplying the index number as an argument. For 2D data sets, spectra can be displayed from either the `f1` or `f2` domain by setting the parameter `trace` equal to `'f1'` or `'f2'`, respectively. After entering `ftld`, interferograms can be viewed by setting `trace= 'f1'` and then typing `ds`.

Spectra are scaled according to the number of completed transients `ct`. If `nt` is arrayed (`nt=1, 2, 4, 8`), each spectrum is scaled by its own `ct`.

Arguments: `index` (used with syntax 1) is the index number of a particular trace to be displayed in arrayed 1D spectra or in 2D spectra (syntax 1).

`options` (used with syntax 2) is any of the following keywords:

- `'toggle'` switches between the box and the cursor modes.
- `'restart'` redraws the cursor if it has been turned off.
- `'expand'` toggles between expanded and full view of the spectrum.
- `'spwp'` interactively adjusts start and width of the spectrum display.
- `'phase'` enters an interactive phasing mode.
- `'thresh'` interactively adjusts the threshold.
- `'z'` interactively sets integral resets.
- `'dscale'` toggles the scale below the spectrum on and off.

- 'lvltlt' interactively adjusts the `lv1` and `tlt` parameters.
- 'scwc' interactively adjusts the start and width of chart.

Examples: `ds`
`ds(7)`
`ds('restart')`

Alternate: Interactive button in the 1D Data Display Menu.

See also: *Getting Started; User Guide: Liquids NMR*

Related:	<code>crmode</code>	Current state of cursors in <code>dfid</code> , <code>ds</code> , or <code>dconi</code> (P)
	<code>ct</code>	Completed transients (P)
	<code>ft1d</code>	Fourier transform along f_2 dimension (C)
	<code>intmod</code>	Integral display mode (P)
	<code>lp</code>	First-order phase in directly detected dimension (P)
	<code>lv1</code>	Zero-order baseline correction (P)
	<code>lvltlt</code>	Control sensitivity of <code>lv1</code> and <code>tlt</code> adjustments (P)
	<code>nt</code>	Number of transients (P)
	<code>phasing</code>	Control update region during <code>ds</code> phasing (P)
	<code>rp</code>	Zero-order phase in directly detected dimension (P)
	<code>select</code>	Select a spectrum without displaying It (C)
	<code>tlt</code>	First-order baseline correction (P)
	<code>trace</code>	Mode for n-dimensional data display (P)
	<code>wft1d</code>	Weight and Fourier transform f_2 for 2D data (C)

ds2d **Display 2D spectra in whitewash mode (C)**

Syntax: `ds2d<(options)>`

Description: Displays a stacked plot of 2D spectra in whitewash mode (after the first spectra, each spectra is blanked out in regions in which it is behind an earlier spectra). Color does not represent intensity (unlike `dcon`), because intensity can be seen visually, but instead successive traces are displayed in different colors so that color represents frequency.

Arguments: `options` can be any of the following keywords:

- 'nobase' is a keyword to activate the `th` parameter to suppress all intensity below the `th` level.
- 'fill' is a keyword to fill in the peaks. When using 'fill', `th` operates linearly and not logarithmically (factors of 2) as it does in the contour or color intensity displays.
- 'fillnb' is a keyword to combine base suppression and peak filling. When using 'fillnb', `th` operates linearly and not logarithmically (factors of 2) as it does in the contour or color intensity displays.
- 'noaxis' is a keyword to omit outlining the display and drawing the horizontal and vertical axis.

Examples: `ds2d`
`ds2d('fillnb')`

See also: *User Guide: Liquids NMR*

Related:	<code>dcon</code>	Display noninteractive color intensity map (C)
	<code>dconi</code>	Control display selection for the <code>dconi</code> program (P)
	<code>ds2dn</code>	Display 2D spectra in whitewash mode without screen erase (C)
	<code>pl2d</code>	Plot 2D spectra in whitewash mode (C)
	<code>th</code>	Threshold (P)

ds2dn **Display 2D spectra in whitewash mode without screen erase (C)**

Syntax: `ds2dn<(options)>`

Description: Displays a stacked plot of 2D spectra in whitewash mode (after the first spectra, each spectra is blanked out in regions in which it is behind an earlier spectra) the same as `ds2d` but without erasing the screen before drawing. The arguments are the same as `ds2d`.

Examples: `ds2dn`
`ds2dn('fillnb')`

See also: *User Guide: Liquids NMR*

Related: `ds2d` Display 2D spectra in whitewash mode (C)

dscale **Display scale below spectrum or FID (C)**

Syntax: `dscale(<axis><,vert_start><,display_start><,color>)>`

Description: Displays a scale under a spectrum or FID.

Arguments: `axis` is a letter to be used to label the axis. For a spectrum scale, if 'p', 'h', 'k', 'c', 'm', 'u', etc. is supplied, the letter within the single quotes is used instead of the current value of `axis`. For an FID scale, if 's', 'm', or 'u' is supplied, it is used instead of the current value of `axisf`.

`vert_start` is a real number that sets the vertical position where the scale is drawn. The default is 5 mm below the current value of the parameter `vp`.

`display_start` is a real number that modifies the start of a display. For example, if the display is from 347 to 447 Hz, but a scale of 0 to 100 Hz is desired, `display_start` would be 0.

`color` is one of the keywords 'red', 'green', 'blue', 'cyan', 'magenta', 'yellow', 'black', or 'white' for the color of the scale.

Examples: `dscale`
`dscale(20)`
`dscale('h',0,'green')`
`dscale('h',vp-10,0)`

See also: *Getting Started*

Related: `axis` Axis label for displays and plots (P)
`axisf` Axis label for FID displays and plots (P)
`pscale` Plot scale below spectrum or FID (C)
`vp` Vertical position of spectrum (P)

dscoef **Digital filter coefficients for downsampling (P)**

Description: Specifies the number of coefficients used in the digital filter. This parameter does not need to be changed as the parameter `downsamp` is changed, because `dscoef` is automatically adjusted by VNMR to give filter cutoffs that are the same, regardless of the value of `downsamp`. This is done by using $dscoef * downsamp / 2$ coefficients in the digital filter. VNMR always rounds $dscoef * downsamp / 2$ to an odd number. If `dscoef` does not exist in the current experiment, enter `addpar('downsamp')` to add it. Entering `addpar('downsamp')` creates the digital filtering and downsampling parameters `downsamp`, `dscoef`, `dsfb`, `dslsfrq`, and `filtfile`.

Values: Number of digital filter coefficients. The default is 61. A larger number of coefficients gives a filter with sharper cutoffs; a smaller number gives a filter with more gradual cutoffs.

See also: *Getting Started*

Related:	<code>addpar</code>	Add selected parameters to current experiment (M)
	<code>downsamp</code>	Downsampling factor applied after digital filtering (P)
	<code>dsfb</code>	Digital filter bandwidth for downsampling (P)
	<code>ds1sfrq</code>	Bandpass filter offset for downsampling (P)
	<code>filtfile</code>	File of FIR digital filter coefficients (P)
	<code>pards</code>	Create additional parameters used for downsampling (M)

dseq **Decoupler sequence for first decoupler (P)**

Applicability: Systems with waveform generators.

Description: Specifies the decoupling sequence (without the `.DEC` file extension) to be used during any period of programmable decoupling on the first decoupler under status control (i.e., `dmm='p'`). The decoupling sequence must be located in the user's `shapelib` directory or in the VNMR system's `shapelib` directory.

See also: *Getting Started*

Related:	<code>dmm</code>	Decoupler modulation mode for first decoupler (P)
	<code>dseq2</code>	Decoupler sequence for second decoupler (P)
	<code>dseq3</code>	Decoupler sequence for third decoupler (P)

dseq2 **Decoupler sequence for second decoupler (P)**

Applicability: Systems with waveform generators.

Description: Specifies the decoupling sequence (without the `.DEC` file extension) to be used during any period of programmable decoupling on the second decoupler under status control (i.e., `dmm2='p'`). The decoupling sequence must be located in the user's `shapelib` directory or in the VNMR system `shapelib` directory.

See also: *Getting Started*

Related:	<code>dmm2</code>	Decoupler modulation mode for second decoupler (P)
	<code>dseq</code>	Decoupler sequence for first decoupler (P)

dseq3 **Decoupler sequence for third decoupler (P)**

Applicability: Systems with waveform generators.

Description: Specifies the decoupling sequence (without the `.DEC` file extension) to be used during any period of programmable decoupling on the third decoupler under status control (i.e., `dmm3='p'`). The decoupling sequence must be located in the user's `shapelib` directory or in the VNMR system's `shapelib` directory.

See also: *Getting Started*

Related:	<code>dmm3</code>	Decoupler modulation mode for third decoupler (P)
	<code>dseq</code>	Decoupler sequence for first decoupler (P)

dsfb **Digital filter bandwidth for downsampling (P)**

Description: Specifies the bandwidth of the digital filter used for downsampling. If `dsfb` does not exist in the current experiment, enter `addpar('downsamp')` to add it. `addpar('downsamp')` creates the digital filtering and downsampling parameters `downsamp`, `dscoef`, `dsfb`, `ds1sfrq`, and `filtfile`.

Values: Number, in Hz. A smaller value rejects frequencies at the spectrum edges; a larger value aliases noise and signals at frequencies outside of $\pm sw/2$.

'n' makes `dsfb` default to the final `sw/2`.

See also: *Getting Started*

Related: **addpar** Add selected parameters to current experiment (M)
downsamp Downsampling factor applied after digital filtering (P)
dscoef Digital filter coefficients for downsampling (P)
dslsfrq Bandpass filter offset for downsampling (P)
filtfile File of FIR digital filter coefficients (P)
pards Create additional parameters used for downsampling (M)
sw Spectral width in directly detected dimension (P)

dshape Display pulse shape or modulation pattern (M)

Syntax: `dshape<(pattern.ext)>`

Description: Displays the real (X) and imaginary (Y) components of a shaped pulse. Any type of waveform (.RF, .DEC or .GRD) can be displayed.

Arguments: `pattern` is the name of a shape or pattern file specified by an absolute file name, relative file name, or a simple pattern file name. `ext` is a file name extension that specifies the file type. In the case of a simple file name, `dshape` searches for the file in the local directory, then in the user's `shapelib`, and finally in the directory `/vnmr/shapelib`. If `pattern.ext` is not given, `dshape` displays the last created waveform stored in the `pbox.fid` file.

Examples: `dshape`
`dshape('Pbox.RF')`

See also: *User Guide: Liquids NMR*

Related: **Pbox** Pulse shaping software (U)
pshape Plot pulse shape or modulation pattern (M)

dshapef Display last generated pulse shape (M)

Syntax: `dshapef`

Description: Displays the real (X) and imaginary (Y) components of last generated shaped pulse, stored in `pbox.fid` file.

See also: *User Guide: Liquids NMR*

Related: **Pbox** Pulse shaping software (U)
pshapef Plot last generated pulse shape (M)

dshapei Display pulse shape or modulation pattern interactively (M)

Syntax: `dshapei<(pattern.ext)>`

Description: Displays the real (X) and imaginary (Y) components of a pulse shape, modulation pattern or gradient shape interactively. `dshapei` overwrites the existing data (FID) after the permission is granted by the user. It also asks for the duration of the waveform and displays the timescale.

Arguments: `pattern` is the name of a shape or pattern file specified by an absolute file name, relative file name, or a simple pattern file name. `ext` is a file name extension that specifies the file type. In the case of a simple file name, `dshapei` searches for the file in the local directory, then in the user's `shapelib`, and finally in the directory `/vnmr/shapelib`. If no file name is given, `dshapei` displays the last created waveform stored in the `pbox.fid` file.

Examples: `dshapei`
`dshapei('myfile.DEC')`

See also: *User Guide: Liquids NMR*

Related: [Pbox](#) Pulse shaping software (U)

dshim Display a shim “method” string (M)

Syntax: (1) `dshim<(file)>`
 (2) `dshim('method' | 'help')`

Description: Looks in the user's `shimmethods` directory and then in the VNMR system `shimmethods` directory for a file and displays the file (syntax 1) or displays information about `method` strings (syntax 2).

Arguments: `file` is the name of a file to be searched for in the `shimmethods` directories. The default is to display the contents of the `shimmethods` directories.

'`method`' is a keyword to explain the structure of `method` strings.

'`help`' is a keyword to describe the `method` strings in the VNMR system's `shimmethods` directory.

Examples: `dshim`
`dshim('method')`
`dshim('help')`

See also: *Getting Started*

Related: [method](#) Autoshim method (P)
[newshm](#) Interactively create a shim “method” with options (M)
[shim](#) Submit an Autoshim experiment to acquisition (C)
[stdshm](#) Interactively create a shim “method” (M)

dslice Display multiple images from multislice data set (M)

Applicability: Systems with imaging capabilities.

Syntax: `dslice<(start, end)>`

Description: Displays multiple images in a 4×3 (row \times column) format, up to a maximum of 12 images from a multislice data set. `dslice` is not valid with a data set acquired using the compressed 2D format.

Arguments: `start` is the number of the beginning slice.

`end` is the number of the ending slice.

Examples: `dslice`
`dslice(2, 3)`

See also: *User Guide: Imaging*

Related: [dmuli](#) Display multiple images (M)

dslsfrq Bandpass filter offset for downsampling (P)

Description: For downsampling, selects a bandpass filter that is not centered about the transmitter frequency. In this way, `dslsfrq` works much like `lsfrq`. If `dslsfrq` does not exist in the current experiment, add it by entering `addpar('downsamp')`. The command `addpar('downsamp')` creates the digital filtering and downsampling parameters `downsamp`, `dscoef`, `dsfb`, `dslsfrq`, and `filtfile`.

Values: A number, in Hz. A positive value selects a region upfield from the transmitter frequency; a negative value selects a downfield region.

See also: *Getting Started*

Related:	addpar	Add selected parameters to current experiment (M)
	downsamp	Downsampling factor applied after digital filtering (P)
	dscoef	Digital filter coefficients for downsampling (P)
	dsfb	Digital filter bandwidth for downsampling (P)
	filtfile	File of FIR digital filter coefficients (P)
	lsfrq	Frequency shift of the fn spectrum in Hz (P)
	movedssw	Set parameters for digital filtering and downsampling (M)
	pards	Create additional parameters used by downsampling (M)

dsn **Measure signal-to-noise (C)**

Syntax: `dsn<(low_field,high_field)>:signal_to_noise,noise`

Description: Measures the signal-to-noise ratio of the spectrum by first measuring the intensity of the largest peak in the spectral range defined by **sp** and **wp**, and then measuring the noise in the spectral region defined by the position of the two cursors. The noise value returned from **dsn** is not scaled by **vs**. The interrelations between the signal-to-noise ratio, the noise, and peak intensities can be illustrated by comparing `dsn:$sn`, `$noise` and `peak:$signal`. In this case, `$sn` is equal to $(\$signal / \$noise) / vs$.

Arguments: `low_field` and `high_field` are the upper and lower frequencies of the noise region to be measured. The default is the position of the two cursors.
`signal_to_noise` is the calculated value of signal-to-noise ratio.
`noise` is the noise value measured within the defined spectral region.

Examples: `dsn:$ston`
`dsn(sp+sp,sp+wp-100)`
`dsn(10000,8000):r1`

See also: *VNMR User Programming*

Related:	dres	Measure linewidth and digital resolution (C)
	peak	Find tallest peak in specified region (C)
	sp	Start of plot (P)
	vs	Vertical scale (P)
	wp	Width of plot (P)

dsnmax **Calculate maximum signal-to-noise (M)**

Syntax: `dsnmax<(noise_region)>`

Description: Finds the best signal-to-noise in a specified region.

Arguments: `noise_region` is the size, in Hz, of the region. The default is the region between the cursors as defined by the parameter **delta**.

Examples: `dsnmax`
`dsnmax(400)`

See also: *VNMR User Programming*

Related:	delta	Cursor difference in directly detected dimension (P)
----------	--------------	--

dsp **Display calculated spectrum (C)**

Syntax: `dsp<(file<,'nods'>)>`

Description: Using the current table of transitions and intensities, **dsp** recalculates the simulated spectrum (using the current value for the linewidth **slw**) and displays the spectrum. **dsp** can only be used after the **spins** program has been run. If

only the linewidth `slw` or vertical scale `svs` have been changed, `dsp` can be used to redisplay the spectrum. If a chemical shift or coupling constant has been changed, however, `dsp` will not display a spectrum reflecting the changes in the parameter; `spins` must be run again to recalculate the new spectrum.

The number of points in the calculated spectrum is $fn/2$. To increase the number of points, change `fn` and rerun `dsp` without doing a transform.

To display a synthetic spectrum, prepare a file in the following format:

```
Freq1, Intens1, LineWidth1, GaussFrac1
Freq2, Intens2, LineWidth2, GaussFrac2
...
FreqN, IntensN, LineWidthN, GaussFracN
```

The units for frequency and line width are Hz. The Gaussian fraction, which is the percentage of the line shape that is Gaussian (the rest is Lorentzian) should be between 0 and 1 (i.e., 0 is pure Lorentzian, 1 is pure Gaussian). Units for intensity are not particularly important. Given numbers in a file `myshape`, it is only necessary to enter `dsp('myshape')` to display the synthetic spectrum. This approach is often preferred over deconvolution for quantifying small shoulders on large peaks.

Arguments: `file` is the name of a file containing spectral information that displays the result of a spectrum deconvolution. Any file in the proper format can be used to generate a display. The default is the file `spins.outdata` in the experiment directory. This file contains information about frequencies, intensities, line widths, and Gaussian/Lorentzian fractions.

'`nodes`' is a keyword for `dsp` to recalculate the simulated spectrum but not to display the spectrum. The spectrum can be displayed with the `ds` or `dss` command.

Examples: `dsp`
`dsp('fitspec.outpar')`

See also: *User Guide: Liquids NMR*

Related:	<code>ds</code>	Display a spectrum (C)
	<code>dss</code>	Display stacked spectra (C)
	<code>fn</code>	Fourier number in directly detected dimension (P)
	<code>slw</code>	Spin simulation linewidth (P)
	<code>spins</code>	Perform spin simulation calculation (C)
	<code>svs</code>	Spin simulation vertical scale (P)

dsp **Type of DSP for data acquisition (P)**

Description: Selects the type of DSP (digital signal processing) for data acquisition:

- *Inline DSP* performs digital filtering and downsampling on the workstation immediately after each oversampled FID is transferred from the console. `sw` and `at` should be set to the values desired for the final spectrum. Only the digital filtered and downsampled data is written to the disk. Selective detection of a region of a spectrum is available using the `moveossw` macro.
- *Real-time DSP* uses optional hardware (not available on all systems) to filter the data prior to summing to memory. Real-time DSP is not compatible with pulse sequences that use explicit acquisition to acquire less than the full number of data points (`np`) in a single acquire statement (e.g., solids sequences such as BR24 and FLIPFLOP).

If either type is active, the filter bandwidth parameter `fb` is not active. The actual analog filter *is* active and is automatically set by the software to a value that matches $(sw/2) * oversamp$ as closely as possible.

Another type of DSP is available that allows post-processing of data. See the description of the `pard`s macro for details.

Values: `'i'` selects inline DSP and calls `addpar('oversamp')` to create the DSP parameters `def_osfilt`, `filtfile`, `oscoef`, `osfb`, `osfilt`, `oslsfrq`, and `oversamp`. A value of `oversamp` greater than 1 causes the next experiment run to be oversampled, digitally filtered, and downsampled back to the selected `sw` prior to saving it to disk. On systems other than `UNITYINOVA`, inline DSP is not possible if interleaving is active (`il='y'`). Also, the command `sa` can be used to stop acquisition, but `ra` cannot be used to resume it. On `UNITYINOVA`, inline DSP is completely compatible with interleaving and with stopping and restarting on acquisition with `sa` and `ra`. Set `fsq='y'` to use frequency-shifted quadrature detection on `UNITYINOVA`.

`'r'` selects real-time DSP and calls the macro `addpar('oversamp')` to create the DSP parameters `def_osfilt`, `filtfile`, `oscoef`, `osfb`, `osfilt`, `oslsfrq`, and `oversamp` (although only `oversamp` and `osfilt` are user adjustable for real-time DSP). Use `dsp='r'` only if the optional DSP hardware is present in the system. On `UNITYINOVA` systems, set `fsq='y'` to use frequency-shifted quadrature detection.

`'n'` (or parameter `dsp` is not present) disables both types of DSP. Set `dsp='n'` if you wish to turn off DSP on a permanent or semi-permanent basis. To turn off DSP within just a single experiment, set `oversamp='n'`.

See also: *Getting Started*

Related:	<code>addpar</code>	Add selected parameters to current experiment (M)
	<code>at</code>	Acquisition time (P)
	<code>def_osfilt</code>	Default value of <code>osfilt</code> (P)
	<code>fb</code>	Filter bandwidth (P)
	<code>filtfile</code>	File of FIR digital filter coefficients (P)
	<code>fsq</code>	Frequency-shifted quadrature detection (P)
	<code>il</code>	Interleave arrayed and 2D experiments (P)
	<code>moveossw</code>	Set oversampling parameters for selected spectral region (M)
	<code>np</code>	Number of data points (P)
	<code>oscoef</code>	Digital filter coefficients for oversampling (P)
	<code>osfb</code>	Digital filter bandwidth for oversampling (P)
	<code>osfilt</code>	Oversampling filter for real-time DSP (P)
	<code>oslsfrq</code>	Bandpass filter offset for oversampling (P)
	<code>oversamp</code>	Oversampling factor for acquisition (P)
	<code>pard</code> s	Create additional parameters used by downsampling (M)
	<code>paros</code>	Create additional parameters used by oversampling (M)
	<code>ra</code>	Resume acquisition stopped with <code>sa</code> command (C)
	<code>sa</code>	Stop acquisition (C)
	<code>sw</code>	Spectral width in the directly detected dimension (P)

dsplanes **Display a series of 3D planes (M)**

Applicability: All systems; however, although `dsplanes` is available on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems, such systems can only process 3D data and cannot acquire 3D data.

Syntax: `dsplanes(start_plane, stop_plane)`

Description: Produces a graphical 2D color or contour map for a subset of 3D planes. The `dconi` program is used to display the planes.

Arguments: `start_plane` specifies the number of the 3D plane with which display is to begin. It must be greater than 0.

`stop_plane` specifies the number of the 3D plane with which the display is to end. If `start_plane` is greater than `stop_plane`, only the first plane, whose number is `start_plane`, is plotted. The range of `stop_plane` depends on the value of the parameter `plane` as follows:

- If `plane` = 'f1f3', range of `stop_plane` is between 0 and `fn2/2`
- If `plane` = 'f2f3', range of `stop_plane` is between 0 and `fn1/2`
- If `plane` = 'f1f2', range of `stop_plane` is between 0 and `fn/2`

Examples: `dsplanes(1,3)`

See also: *User Guide: Liquids NMR*

Related:	<code>dconi</code>	Interactive 2D data display (C)
	<code>dplane</code>	Display a 3D plane (M)
	<code>dproj</code>	Display a 3D plane projection (M)
	<code>getplane</code>	Extract planes from 3D spectral data set (M)
	<code>nextpl</code>	Display the next 3D plane (M)
	<code>plane</code>	Currently displayed 3D plane type (P)
	<code>plplanes</code>	Plot a series of 3D planes (M)
	<code>prevpl</code>	Display the previous 3D plane (M)

dss **Display stacked spectra (C)**

Syntax: `dss(<start, finish<, step>><, options>)>`

Description: Displays one or more spectra on the screen, but not interactively like the command `ds`. When a single spectrum is displayed, integral display is controlled by the parameter `intmod`, which has the following values:

- `intmod` = 'off' turns off the integral display.
- `intmod` = 'full' displays the entire integral.
- `intmod` = 'partial' displays every other integral region.

For arrayed 1D spectra or for 2D spectra, a particular trace can be viewed by supplying the index number as an argument. For 2D data sets, spectra can be displayed from either the `f1` or `f2` domain by setting the parameter `trace` equal to 'f1' or 'f2', respectively. After entering `ft1d`, interferograms can be viewed by setting `trace` = 'f1' and then entering `dss`. Multiple spectra can be displayed by supplying indexes of the first and last spectra.

The position of the first spectrum is governed by the parameters `wc`, `sc`, and `vp`. For 1D data, subsequent spectra are positioned relative to the preceding spectrum by the parameters `vo` (vertical offset) and `ho` (horizontal offset). For 2D data, `ho` defines the total horizontal offset between the first and last spectrum. Also for 2D data, `vo` is inactive while the parameter `wc2` defines the total vertical offset between the first and last spectrum.

The parameter `cutoff`, if it exists and is active, defines the distance above and below the current vertical position `vp` at which peaks are truncated. By arraying `cutoff` to have two different values, the truncation limits above and below the current vertical position can be controlled independently. For example, `cutoff` = 50 truncates peaks at `vp` + 50 mm and `vp` - 50 mm. `cutoff` = 50, 10 truncates peaks at `vp` + 50 mm and `vp` - 10 mm.

Arguments: `start` is the index of the first spectra when displaying multiple spectra. It is also the index number of a particular trace to be viewed when displaying arrayed 1D spectra or 2D spectra.

`finish` is the index of the last spectra when displaying multiple spectra. Since the parameter `arraydim` is automatically set to the total number of spectra, it can be used to set `finish` to include all spectra (e.g., `dss(1, arraydim, 3)`).

`step` is the increment for the spectral index when displaying multiple spectra. The default is 1.

options can be any of the following:

- 'all' is a keyword to display all of the spectra.
- 'int' is a keyword to only display the integral, independently of the value of the parameter `intmod`
- 'top' or 'side' are keywords that cause the spectrum to be displayed either above or at the left edge, respectively, of a contour plot. This assumes that the parameters `sc`, `wc`, `sc2`, and `wc2` are those used to position the contour plot.
- 'dodc' is a keyword for all spectra to be drift corrected independently.
- 'red', 'green', 'blue', 'cyan', 'magenta', 'yellow', 'black', and 'white' are keywords that select a color.

Examples: `dss(1, 3)`
`dss(1, 12, 3, 'green')`

See also: *User Guide: Liquids NMR*

Related:	<code>cutoff</code>	Data truncation limit (P)
	<code>dssa</code>	Display stacked spectra automatically (C)
	<code>dssan</code>	Display stacked spectra automatically without erasing (C)
	<code>dssh</code>	Display stacked spectra horizontally (C)
	<code>dsshn</code>	Display stacked spectra horizontally without erasing (C)
	<code>dssn</code>	Display stacked spectra without screen erase (C)
	<code>dsww</code>	Display spectra in whitewash mode (C)
	<code>ft1d</code>	Fourier transform along f_2 dimension (C)
	<code>ho</code>	Horizontal offset (P)
	<code>intmod</code>	Integral display mode (P)
	<code>pl</code>	Plot spectra (C)
	<code>plww</code>	Plot spectra in whitewash mode (C)
	<code>sc</code>	Start of chart (P)
	<code>sc2</code>	Start of chart in second direction (P)
	<code>trace</code>	Mode for 2D data display (P)
	<code>vo</code>	Vertical offset (P)
	<code>vp</code>	Vertical position of spectrum (P)
	<code>wc</code>	Width of chart (P)
	<code>wc2</code>	Width of chart in second direction (P)

dssa **Display stacked spectra automatically (C)**

Syntax: `dssa(<start, finish<, step>><, options>)>`

Description: Displays one or more spectra automatically. When a single spectrum is displayed, integral display is controlled by the parameter `intmod`, which has the following values:

- `intmod='off'` turns off the integral display.
- `intmod='full'` displays the entire integral.
- `intmod='partial'` displays every other integral region.

For arrayed 1D spectra or for 2D spectra, a particular trace can be viewed by supplying the index number. For 2D data sets, spectra can be displayed from either the f_1 or f_2 domain by setting the parameter `trace` equal to 'f1' or 'f2', respectively. Following the command `ft1d`, interferograms may be viewed by setting `trace='f1'` and then entering `dssa`. Multiple spectra can be displayed by supplying indexes of the first and last spectra.

The position of the first spectrum is governed by the parameters `wc`, `sc`, and `vp`. For 1D data, subsequent spectra are positioned relative to the preceding spectrum by the parameters `vo` (vertical offset) and `ho` (horizontal offset). For 2D data, `ho` defines the total horizontal offset between the first and last spectrum. Also for 2D data, `vo` is inactive while the parameter `wc2` defines the total vertical offset between the first and last spectrum. To display spectra “automatically,” the command `dssa` adjusts the parameters `vo` and `ho` to fill the screen in a lower left to upper right presentation (`wc` must be set to less than full screen width for this to work)

The parameter `cutoff`, if it exists and is active, defines the distance above and below the current vertical position `vp` at which peaks are truncated. By arraying `cutoff` to have two different values, the truncation limits above and below the current vertical position can be controlled independently. For example, `cutoff=50` truncates peaks at `vp+50` mm and `vp-50` mm. `cutoff=50,10` truncates peaks at `vp+50` mm and `vp-10` mm.

Arguments: `start` is the index of the first spectra when displaying multiple spectra. It is also the index number of a particular trace to be viewed when displaying arrayed 1D spectra or 2D spectra.

`finish` is the index of the last spectra when displaying multiple spectra.

`step` is the increment for the spectral index when displaying multiple spectra. The default is 1.

`options` can be any of the following:

- 'all' is a keyword to display all of the spectra.
- 'int' is a keyword to only display the integral, independently of the value of the parameter `intmod`
- 'dodc' is a keyword for all spectra to be drift corrected independently.

Examples: `dssa(1,3)`

See also: *User Guide: Liquids NMR*

Related:	<code>cutoff</code>	Data truncation limit (P)
	<code>dss</code>	Display stacked spectra (C)
	<code>dssan</code>	Display stacked spectra automatically without erasing (C)
	<code>dssh</code>	Display stacked spectra horizontally (C)
	<code>dsshn</code>	Display stacked spectra horizontally without erasing (C)
	<code>dssn</code>	Display stacked spectra without screen erase (C)
	<code>dsww</code>	Display spectra in whitewash mode (C)
	<code>ft1d</code>	Fourier transform along f_2 dimension (C)
	<code>ho</code>	Horizontal offset (P)
	<code>intmod</code>	Integral display mode (P)
	<code>pl</code>	Plot spectra (C)
	<code>plww</code>	Plot spectra in whitewash mode (C)
	<code>sc</code>	Start of chart (P)
	<code>sc2</code>	Start of chart in second direction (P)
	<code>trace</code>	Mode for 2D data display (P)
	<code>vo</code>	Vertical offset (P)
	<code>vp</code>	Vertical position of spectrum (P)

`wc` Width of chart (P)
`wc2` Width of chart in second direction (P)

dssan Display stacked spectra automatically without erasing (C)

Syntax: `dssan(<start, finish<, step>><, options>)>`

Description: Functions the same as the command `dssa` except the graphics window is not erased before starting the display. This allows composite displays of many spectra to be created. The arguments are the same as `dssa`.

Examples: `dssan(1, 3)`

See also: *User Guide: Liquids NMR*

Related: `dssa` Display stacked spectra automatically (C)

dssh Display stacked spectra horizontally (C)

Syntax: `dssh(<start, finish<, step>><, options>)>`

Description: Displays one or more spectra horizontally. When a single spectrum is displayed, integral display is controlled by the parameter `intmod.`, which can have the following values:

- `intmod='off'` turns off the integral display.
- `intmod='full'` displays the entire integral.
- `intmod='partial'` displays every other integral region.

For arrayed 1D spectra or for 2D spectra, a particular trace can be viewed by supplying the index number as an argument. For 2D data sets, spectra can be displayed from either the f_1 or f_2 domain by setting the parameter `trace` equal to '`f1`' or '`f2`', respectively. After entering `ft1d`, interferograms can be viewed by setting `trace='f1'` and then entering `dss`. Multiple spectra can be displayed by supplying indexes of the first and last spectra.

The position of the first spectrum is governed by the parameters `wc`, `sc`, and `vp`. For 1D data, subsequent spectra are positioned relative to the preceding spectrum by the parameters `vo` (vertical offset) and `ho` (horizontal offset). For 2D data, `ho` defines the total horizontal offset between the first and last spectrum. Also for 2D data, `vo` is inactive while the parameter `wc2` defines the total vertical offset between the first and last spectrum. To display spectra horizontally, the command `dssh` causes `vo` to be set to zero and for `ho`, `sc`, and `wc` to be adjusted to fill the screen from left to right with the entire array.

The parameter `cutoff`, if it exists and is active, defines the distance above and below the current vertical position `vp` at which peaks are truncated. By arraying `cutoff` to have two different values, the truncation limits above and below the current vertical position may be controlled independently. For example, `cutoff=50` truncates peaks at `vp+50` mm and `vp-50` mm, and `cutoff=50, 10` truncates peaks at `vp+50` mm and `vp-10` mm.

Arguments: `start` is the index of the first spectra when displaying multiple spectra. It is also the index number of a particular trace to be viewed when displaying arrayed 1D spectra or 2D spectra.

`finish` is the index of the last spectra when displaying multiple spectra.

`step` is the increment for the spectral index when displaying multiple spectra. The default is 1.

`options` can be any of the following:

- '`all`' is a keyword to display all of the spectra.

- 'int' is a keyword to only display the integral, independently of the value of the parameter `intmod`
- 'dodc' is a keyword that causes all spectra to be drift corrected independently.

Examples: `dssh(1,3)`

See also: *User Guide: Liquids NMR*

Related:	<code>cutoff</code>	Data truncation limit (P)
	<code>dss</code>	Display stacked spectra (C)
	<code>dssa</code>	Display stacked spectra automatically (C)
	<code>dssan</code>	Display stacked spectra automatically without erasing (C)
	<code>dsshn</code>	Display stacked spectra horizontally without erasing (C)
	<code>dssn</code>	Display stacked spectra without screen erase (C)
	<code>dsww</code>	Display spectra in whitewash mode (C)
	<code>ft1d</code>	Fourier transform along f_2 dimension (C)
	<code>ho</code>	Horizontal offset (P)
	<code>intmod</code>	Integral display mode (P)
	<code>pl</code>	Plot spectra (C)
	<code>plww</code>	Plot spectra in whitewash mode (C)
	<code>sc</code>	Start of chart (P)
	<code>sc2</code>	Start of chart in second direction (P)
	<code>trace</code>	Mode for 2D data display (P)
	<code>vo</code>	Vertical offset (P)
	<code>vp</code>	Vertical position of spectrum (P)
	<code>wc</code>	Width of chart (P)
	<code>wc2</code>	Width of chart in second direction (P)

dsshn **Display stacked spectra horizontally without erasing (C)**

Syntax: `dsshn(<start,finish<,step>><,options>)>`

Description: Functions the same as the command `dssh` except the graphics window is not erased before starting the display. This allows composite displays of many spectra to be created. The arguments are the same as `dssh`.

Examples: `dssh(1,3)`

See also: *User Guide: Liquids NMR*

Related: `dssh` Display stacked spectra horizontally (C)

dssl **Label a display of stacked spectra (M)**

Syntax: `dssl(<options>)`

Description: Displays a label for each element in a set of stacked spectra. The label is an integer value from 1 up to the number of spectra in the display.

Note that if `wysiwyg='n'`, labels can appear at incorrect positions. The positions were empirically determined for a large screen display and are not guaranteed to be correct for all displays.

Arguments: `options` control the display (more than one option can be entered as long as the options do not conflict with each other):

- 'center', 'left', 'right', 'top', 'bottom', 'above', and 'below' are keywords setting the position of the displayed index relative to each spectrum.
- 'value' is a keyword that produces a display of the values of each array element, instead of an integer index.

- 'list=xxx' produces a display of the values contained in the arrayed parameter xxx.
- 'format=yyy' uses the format yyy to control the display of each label. See the `write` command for information about formats.

Examples: `dssl`
`dssl('top','left')`
`dssl('value','format=%3.1f')`

See also: *User Guide: Liquids NMR*

Related: `dss` Display stacked spectra (C)
`write` Write formatted text to a device (C)

dssn Display stacked spectra without screen erase (C)

Syntax: `dssn(<start,finish,<step>><,options>)>`

Description: Functions the same as the command `dss` except the graphics window is not erased before starting the display. This allows composite displays of many spectra to be created. The arguments are the same as `dss`.

Examples: `dssn(1,3)`

See also: *User Guide: Liquids NMR*

Related: `dss` Display stacked spectra (C)

dsvast Display VAST data in a stacked 1D-NMR matrix format (M)

Applicability: Systems with the VAST accessory.

Syntax: `dsvast<(display order,number of columns displayed)>`

Description: `dsvast` will arrange and display the traces from a reconstructed 2D data set (see (see `vastglue`) as an array of 1D spectra in a matrix of 1D spectra. If no arguments are provided, the number of rows and columns will be determined by the periodicity of the display order based on the `doneQ`. For example, if a block of 96 spectra (typical for a microtiter-plate) have been acquired using VAST automation, the spectra will be displayed in a matrix 8 rows and 12 columns with the well label using the format `[A->H][1->12]`.

The spectra can be plotted using the macro `plvast`.

Arguments: `display order` is optional and its default value is the glue order as listed in `glueorderarray`. A `display order` can be defined using the `plate_glue` program.

`number of columns displayed`. The default value of is deduced by examining the periodicity of the requested display order. The `number of columns displayed` can entered as the second argument or as the first argument if the default `display order` is used.

Examples: `dsvast`
`dsvast(12)`
`dsvast('glue_file',4)` *User Guide: Liquids NMR*

Related: `dsast2d` Display VAST data in a pseudo-2D format (M)
`plvast` Plot VAST data in a stacked 1D-NMR matrix (M)
`plvast2d` Plot VAST data in a pseudo-2D format (M)
`plate_glue` Define a display order (U)

dsvast2d **Display VAST data in a pseudo-2D format (M)**

Applicability: Systems with the VAST accessory.

Syntax: `dsvast2d (number)`

Description: If an array of 1D spectra have been acquired (in particular if a block of 96 spectra has been acquired using VAST automation, especially in a microtiter-plate format), and if these spectra have been glued into a reconstructed 2D dataset (see `vastglue`), this macro will arrange and display them (on the screen) in a convenient pseudo-2D format (almost like an LC-NMR chromatogram). Well labels are not attached to the spectra and spectra are plotted with 8 spectra per row.

Arguments: The default is to display all the spectra (from 1 through `arraydim`) with 8 columns (spectra) and 12 rows. An optional argument `dsvast2d (number)` allows one to specify that only spectra from *I* through *number* should be plotted. The number of spectra displayed is rounded up to the nearest multiple of 8.

See also: *User Guide: Liquids NMR*

Related: `dsast` Display VAST data in a 1D-NMR matrix format (M)
`plvast` Plot VAST data in a stacked 1D-NMR matrix (M)
`plvast2d` Plot VAST data in a pseudo-2D format (M)

dsww **Display spectra in whitewash mode (C)**

Syntax: `dsww (<start , finish < , step >> < , 'int ' >)`

Description: Displays one or more spectra in whitewash mode (after the first spectra, each spectra is blanked out in regions in which it is behind a prior spectra).

Arguments: `start` is the index of the first spectra when displaying multiple spectra. It is also the index number of a particular trace to be viewed when displaying arrayed 1D spectra or 2D spectra; default is to display all spectra.

`finish` is the index of the last spectra when displaying multiple spectra.

`step` is the increment for the spectral index when displaying multiple spectra. The default is 1.

`'int '` is a keyword to display only the integral, independently of the value of the parameter `intmod`

Examples: `dsww (1 , 3)`

See also: *User Guide: Liquids NMR*

Related: `dss` Display stacked spectra (C)
`dssa` Display stacked spectra automatically (C)
`dssan` Display stacked spectra automatically without erasing (C)
`dssh` Display stacked spectra horizontally (C)
`dsshn` Display stacked spectra horizontally without erasing (C)
`dssn` Display stacked spectra without screen erase (C)
`pl` Plot spectra (C)
`plww` Plot spectra in whitewash mode (C)

dtext **Display a text file in graphics window (M)**

Syntax: `dtext (<file , x , y > < : $x_next , $y_next , $increment >`

Description: Displays a text file in the graphics window.

Arguments: `file` is the name of a text file. The default is the current experiment text file.

x and y are coordinates of the first line of text. This positions the location of the output. The default is the upper left-hand corner of the screen.

$\$x_next$ and $\$y_next$ are the coordinates where the start of the next line would have been displayed. This is useful for subsequent character display.

$\$increment$ is the increment between lines.

Examples: `dttext`
`dttext(userdir+' /exp3/text ')`
`dttext(100,100)`
`dttext:$x,$y,$dy`

See also: *Getting Started*

Related: `pltext` Plot a text file (M)
`ptext` Print out a text file (M)
`text` Display text or set new text for current experiment (C)
`write` Write formatted text to a device (C)

dttrig Delay to wait for another trigger or acquire a spectrum (P)

Applicability: Systems with LC-NMR accessory.

Description: If `ntrig` is greater than 0 after a trigger is detected, a pulse sequence waits for `dttrig` seconds before either waiting for another trigger or acquiring a spectrum. Typically, after the LC has positioned the sample in the NMR probe and stopped the pump, there is a small time (30 seconds) during which conditions (pressure, etc.) in the NMR probe are still settling; better NMR performance is obtained if an appropriate delay is inserted using `dttrig`. If `dttrig` does not exist, a value of 0 is assumed. If `dttrig` does not exist, the `parlc` macro can create it.

See also: *User Guide: Liquids NMR*

Related: `ntrig` Number of trigger signals to wait before acquisition (P)
`parlc` Create LC-NMR parameters (M)

dtune Tune lock channel on GEMINI 2000 (M)

Applicability: *GEMINI 2000* systems

Syntax: `dtune`

Description: Turns on the lock (^2H) transmitter, directing about 0.5 watts of rf to the probe coil. Before entering `dtune`, be sure to move the proper cable on the back of the left-hand magnet leg to the BNC connector labeled TUNE, and also to move the proper cable leading to the probe to the BNC connector labeled TUNE. Enter `tuneoff` to turn off the transmitter. `dtune` cannot be executed while the console is acquiring or interactive acquisition (`acqi`) is connected.

CAUTION: An incorrectly tuned lock channel can damage equipment and cause erratic results. Only qualified service personnel should tune the lock channel.

See also: *Getting Started*.

Related: `acqi` Interactive acquisition display process (C)
`btune` Tune broadband channel on broadband *GEMINI 2000* (M)
`ctune` Tune carbon channel on $^1\text{H}/^{13}\text{C}$ *GEMINI 2000* (M)
`htune` Tune proton channel on *GEMINI 2000* (M)
`tuneoff` Turn off probe tuning mode, *MERCURY-Series*, *GEMINI 2000* (M)

E

e Eject sample (M)

Applicability: Systems (including *MERCURY* and *GEMINI 2000*) with spin control hardware.

Syntax: `e`

Description: Ejects the sample from the probe by turning on the eject air and the slow drop air. The `e` macro functions the same as the `eject` macro.

See also: *Getting Started*

Related: `eject` Eject sample (M)
`i` Insert sample (M)
`insert` Insert sample (M)

eaddr Display Ethernet address (M,U)

Syntax: `eaddr`

Description: Displays the name of the local host and its hardware Ethernet address. The 48-bit address is presented in octal, decimal, and hexadecimal formats.

See also: *Getting Started*

Related: `dnode` Display list of valid limNET nodes (M,U)

ecc Set up parameters to get eddy current compensation data (M)

Applicability: Systems with the imaging module.

Syntax: `ecc`

Description: Loads parameter sets during imaging installation for a pulse sequence to obtain eddy current compensation data using balance gradients.

See also: *Imaging Module Installation Manual*

Related: `eddyout` Data analysis of eddy current compensation (M)

ecctabl Put gcal value and ecc file into table (M)

Applicability: Systems with the imaging module.

Syntax: `ecctabl<(ecc_file)<,gcal>>`

Description: Moves the `gcal` value and `ecc` file into the reference table `ecctabl` in `$vnmrsystem/imaging/eddylib`. If the `gcal` value or file name would overwrite data already in the table, the monitor displays a prompt to confirm the overwrite.

Arguments: `ecc_file` specifies the name of the `ecc` file to be placed in the `ecctabl` reference table. The default value is the file name `'curecc'`.

`gcal` specifies the `gcal` value to be placed in the `ecctabl` reference table. The default is the current `gcal` value.

Examples: `ecctabl`
`ecctabl('test1',0.001)`

See also: *User Guide: Imaging*

Related: `ecc` Set up parameters to obtain compensation data (M)
`gcal` Gradient calibration constant (P)
`getgcal` Get gcal value from table (M)

ecctool **Open eccTool window (M)**

Applicability: Systems with imaging capabilities.

Syntax: `ecctool`

Description: Opens the `eccTool` window to adjust eddy current compensation parameters.

See also: *User Guide: Imaging*

echo **Display strings and parameter values in text window (C)**

Syntax: `echo(<'-n', >string1, string2, ...)>`

Description: Displays strings and parameter values in the text window similar to the UNIX `echo` command.

Arguments: `'-n'` is a keyword that suppresses advancing to the next line. The default is to advance to the next line.

`string1, string2, ...` are one or more strings (surrounded with single quote marks) or parameters. The format used for numbers is identical to the `%g` format described for the `write` command.

Examples: `echo`
`echo('This is a string')`
`echo('Pulse Width is: ', pwr)`
`echo('-n', 'No new line')`

See also: *VNMR User Programming*

Related: `write` Write formatted text to a device (C)

echo **Current echo index for transformed image (P)**

Applicability: Systems with imaging capabilities.

Description: Stores the current echo index for the transformed image.

See also: *User Guide: Imaging*

Related: `element` Current array index for transformed image (P)

eddyout **Data analysis of eddy current compensation (M)**

Applicability: Systems with the imaging module.

Syntax: `eddyout(start, stop)`

Description: Analyzes the data obtained with the pulse sequence set up by `ecc` for a series of acquisitions obtained after varying delays following shut off of a gradient. `eddyout` calculates the time constants and amplitudes of the eddy currents and recommends new time constants and amplitudes to be set into the compensation networks.

Arguments: `start` specifies the number of starting array of spectra acquired by `ecc`.
`stop` specifies the number of the ending array of spectra acquired by `ecc`.

Examples: `eddyout(1, 16)`

See also: *User Guide: Imaging*

Related: `ecc` Set up parameters to obtain compensation data (M)

eddySend Update acquisition eddy current settings (M)

Applicability: Systems with the imaging module.

Syntax: `eddySend<(file)>`

Description: Assigns the compensation data from `eccTool` to the current eddy current compensation file specified by `curecc`, then sets the compensation data into the acquisition system. `eccTool` uses `eddySend` to automatically track the file(s) in use by `eccTool`.

Arguments: `file` is the file name of data from `eccTool`. If that file exists, that data is assigned to the current compensation file and becomes `curecc`. The default is the data in the current compensation file is loaded from `curecc`.

Examples: `eddySend`
`eddySend('data04')`

See also: *User Guide: Imaging*

Related: `curecc` Name of eddy current compensation file (P)
`eccTool` Pop up `eccTool` window (M)

edit Edit a file with user-selectable editor (M)

Syntax: `edit(file)`

Description: Opens a file for editing using a text editor. The default editor is `vi`. To select another editor, set the UNIX environmental variable `vnmreditor` to the name of the editor (change the line `setenv vnmreditor old_editor` in `.login` to become `setenv vnmreditor new_editor`, e.g., `setenv vnmreditor emacs`) and make sure a script with the prefix `vnmr_` followed by the name of the editor (e.g., `vnmr_emacs`) is placed in the `bin` subdirectory of the VNMR system directory. The script file makes adjustments for the type of graphic interface in use.

Scripts provided with VNMR include `vnmr_vi` and `vnmr_textedit`. To create other scripts, see the `vnmr_vi` script for non-window editor interfaces and the `vnmr_textedit` script for window-based editor interfaces.

Arguments: `file` is the name of the file you wish to edit.

Examples: `edit('myfile')`

See also: *VNMR User Programming*

Related: `paramedit` Edit a parameter and its attributes with user-selected editor (C)
`paramvi` Edit a parameter and its attributes with `vi` editor (M)
`macroedit` Edit a user macro with user-selectable editor (C)
`macrovi` Edit a user macro with `vi` editor (C)
`menuvi` Edit a menu with the `vi` editor (M)
`textvi` Edit text file of current experiment with `vi` editor (M)

eff_echo Effective echo position in EPI experiments (P)

Applicability: Systems with echo planar imaging (EPI) capabilities.

Description: Refers to the echo showing the highest signal in an EPI echo-train. The readout gradient dephaser is adjusted so that the maximum signal occurs at `eff_echo`.

Values: Usually set to `nv/2`.

See also: *User Guide: Imaging*

Related: **nv** Number of phase encode steps for 1st indirectly detected dim. (P)

eject **Eject sample (M)**

Applicability: Systems (including *MERCURY* and *GEMINI 2000*) with spin control hardware.

Syntax: `eject`

Description: Ejects the sample from the probe by turning on the eject air and the slow drop air. The `e` macro functions the same as the `e` macro.

See also: *Getting Started*

Related: **e** Eject sample (M)
i Insert sample (M)
insert Insert sample (M)

elist **Display directory on remote VXR-style system (M,U)**

Syntax: (From VNMR) `elist(remote_node,remote_directory)`
 (From UNIX) `elist remote_node remote_directory`

Description: Lists directory contents on a remote VXR-style (Gemini, VXR-4000, or XL) system.

Arguments: `remote_node` is the name of the remote VXR-style system.
`remote_directory` is the name of the directory on the remote system.

Examples: (From VNMR) `elist('gemini','fidlib')`
 (From UNIX) `elist gemini fidlib`

See also: *Getting Started*

Related: **dnode** Display list of valid limNET nodes (M,U)

element **Current array index for transformed image (P)**

Applicability: Systems with imaging capabilities.

Description: Stores the current array index for the transformed image.

See also: *User Guide: Imaging*

Related: **echo** Current echo index for transformed image (P)

enter **Enter sample information for automation run (M,U)**

Applicability: Systems with an automatic sample changer.

Syntax: (From VNMR) `enter<(file<,configuration_file)>>`
 (From UNIX) `enter <file> <configuration_file>`

Description: Enables entry of sample information for automation runs, including the sample location, user information, solvent used, experiment or experiments to run, and arbitrary text information. `enter` can also access *GLIDE* experiments. `enter('abc')` creates a directory named `abc`. In this directory is a file named `abc`, which contains experiment information. Also in the directory is a directory named `abc.macdir`, which contains *GLIDE*-related information for an automation run.

Arguments: `file` is the name of the file to be edited. The default is that `enter` prompts for this information. If the file already exists, new entries are appended to it.

`configuration_file` is the name of a user-supplied file that customizes `enter` for local use. Several configuration files are provided:

- `enter.conf` is used when defining an experiment when an automation run is not currently active.
- `auto.conf` is used when defining an experiment for a current automation run. The `walkup` macro is provided for this style of entering samples.
- `gilson.conf` is used with the VAST accessory.

Examples: (From VNMR or UNIX) `enter`
 (From VNMR) `enter('mysamples')`
 (From UNIX) `enter MySamples`
 (From VNMR) `enter('mysamples', 'auto.conf')`

See also: *User Guide: Liquids NMR; VNMR User Programming, Walkup NMR Using GLIDE*

Related: `auto` Set up an automation directory (C)
`autogo` Start an automation run (C)
`autoname` Prefix for automation data file (P)
`autora` Resume a suspended automation run (C)
`autosa` Suspend current automation run (C)
`printer` Printer device (P)
`status` Display status of all experiments (C)
`walkup` Walkup automation (M)

enterdialog Start a dialog window using enterexp file (M)

Applicability: Systems with automation.

Syntax: `enterdialog`

Description: Internal macro used by `enter` to start a dialog window using the `enterexp` file in the `dialoglib` directory.

Related: `enter` Enter sample information for automation run (M,U)

epift Process and display image in EPI experiments (M)

Applicability: Systems with echo planar imaging (EPI) capabilities.

Syntax: `epift(index)`

Description: Processes and displays an image in array number `index`. The first data array must contain the reference scan. The phase correction information saved in the file `phasemap` is used to correct phase errors in EPI data. `phasemap` must be present in the current experiment directory. Use `dconi` to view the data.

Arguments: `index` is the array number of the image.

See also: *User Guide: Imaging*

Related: `dconi` Interactive 2D data display (C)
`epiph` Generate phase correction map in EPI experiments (M)
`pcmapapply` Apply phase correction map to data in EPI experiments (C)

epiph Generate phasemap file in EPI experiments (M)

Applicability: Systems with echo planar imaging (EPI) capabilities.

Syntax: `epiph`

Description: Generates the `phasemap` file from the EPI reference scan. The file is generated in the current experiment directory for EPI processing. The first data array must

correspond to the reference scan, which is collected with the phase-encode gradient turned off (`image=0`).

See also: *User Guide: Imaging*

Related: `episet` Set up parameters for EPI experiments (M)
`image` Control phase encoding gradient in EPI experiments (P)
`pcmapgen` Generate phase correction map in EPI experiments (M)

epirs Reverse spectral data in EPI experiments (C)

Applicability: Systems with echo planar imaging (EPI) capabilities.

Syntax: `epirs`

Description: Reverses spectral data. It is used by `epift`.

See also: *User Guide: Imaging*

Related: `epift` Process and display images in EPI experiments (M)

epirun Collect, process, and display EPI data (M)

Applicability: Systems with echo planar imaging (EPI) capabilities.

Syntax: `epirun`

Description: Collects, process, and displays EPI data. It is used to obtain a single EPI image. The `phasemap` file must be present in the current experiment directory.

See also: *User Guide: Imaging*

Related: `epiph` Generate phasemap file in EPI experiments (M)
`episet` Set up parameters for EPI experiments (M)

episet Set up parameters for EPI experiments (M)

Applicability: Systems with echo planar imaging (EPI) capabilities.

Syntax: `episet`

Description: Collects an EPI dataset with the phase-encode gradient turned off (`image=0`). It optimizes parameters for EPI, collects a reference scan, and allows you to adjust the gradient parameters `groa` and `grora` and the timing parameter `tep`. The `phasemap` file is generated in the current experiment directory.

See also: *User Guide: Imaging*

Related: `epiph` Generate phasemap file in EPI experiments (M)
`groa` Readout gradient adjuster in EPI experiments (P)
`grora` Readout dephasing gradient adjuster in EPI experiments (P)
`image` Control phase encoding gradient in EPI experiments (P)
`tep` Post-acquisition delay in EPI experiment (P)

episvib Save EPI images in FDF for ImageBrowser (M)

Applicability: Systems with echo planar imaging (EPI) capabilities.

Syntax: `episvib`

Description: Saves images in Flexible Data Format (FDF) for viewing with ImageBrowser. The first image in an arrayed dataset must contain the reference scan. This scan must be acquired with the phase encode gradient turned off.

See also: *User Guide: Imaging*

Related: `browser` Start ImageBrowser application (U)

eread **Transfer file from remote source (M,U)**

Applicability: Systems with limNET protocol software installed.

Syntax: (From VNMR) `eread(local_file,remote_node,remote_file)`
 (From UNIX) `eread local_file remote_node remote_file`

Description: Copies a remote file to the local host. It will not overwrite a preexisting file.

Arguments: `local_file` is the file name of the local host. If `local_file` is not a dot file (i.e., starts with “.”), `eread` uses the “I1” and “I2” values of the remote file to create an extension and then append it to the local file name.

`remote_node` is a symbolic node name for a specified node file. Use the command `dnode` to list nodes defined on your system. The names of the remote computers or “nodes” available to the limNET protocol are contained in the file `/vnmr/nodes`. *Note that this is not the same file as the name of the remote computers available to the Internet protocol (IP), which are contained in the file /etc/hosts.* Each user only needs to know the “names” of relevant nodes.

`remote_file` is the name of file to be transferred from the remote host.

Examples: (From VNMR) `eread('osv700','VXR4000','dsk1.osv700')`
 (From UNIX) `eread osv700 VXR4000 dsk1.osv700`

See also: *Getting Started*

Related: `dnode` Display list of valid limNET nodes (M,U)
`ewrite` Transfer file to remote destination (M,U)

ernst **Calculate the Ernst angle pulse (C)**

Syntax: `ernst(t1_estimate<,90_pulse_width>)`

Description: Calculates the optimum (“Ernst”) pulse width according to the formula

$$pw = \cos^{-1}(\exp^{-(at+d1)/t1_estimate}) \cdot (pw90/360)$$

The new `pw` value is entered in the parameter table.

Arguments: `t1_estimate` is an estimate of the T_1 for a peak of interest.

`90_pulse_width` is a 90° pulse width determined by the parameter `pw90`. The default is the current value of parameter `pw90` if `pw90` exists.

Examples: `ernst(5)`
`ernst(3,12.6)`

See also: *Getting Started*

Related: `pw` Pulse width (P)
`pw90` 90° pulse width (P)

errlog **Display recent VNMR error messages (C)**

Syntax: `errlog`

Description: Displays in the text window the most recent VNMR error messages. The global parameter `errloglen` controls the number of lines displayed. If `errloglen` is not defined, `errlog` displays 10 lines by default.

See also: *Getting Started*

Related: `acqstatus` Acquisition status (P)
`errloglen` Number of lines in VNMR error message display (P)

errloglen **Number of lines in VNMR error message display (P)**

Description: Sets the number of lines in the display of VNMR error messages by the `errlog` command.

Values: Integer, default is 10.

See also: *Getting Started*

Related: `errlog` Display recent VNMR error messages (P)

ewrite **Transfer file to remote destination (M,U)**

Applicability: Systems with limNET protocol software installed.

Syntax: (From VNMR) `ewrite(local_file,remote_node,remote_file)`
(From UNIX) `ewrite local_file remote_node remote_file`

Description: Takes a preexisting local file and copies it to a remote host. The file cannot preexist on the remote host.

Arguments: `local_file` is the file name of the local host.

`remote_node` is a symbolic node name for a specified node file. Use the command `dnode` to list nodes defined on your system. The names of the remote computers or “nodes” available to the limNET protocol are contained in the file `/vnmr/nodes`. *Note that this is not the same file as the name of the remote computers available to the Internet Protocol (IP), which are contained in the file `/etc/hosts`.* Each user only needs to know the “names” of relevant nodes.

`remote_file` is the name of file to be transferred from the remote host.

Examples: (From VNMR) `ewrite('osv700','VXR4000','dsk1.osv700')`
(From UNIX) `ewrite osv700 VXR4000 dsk1.osv700`

See also: *Getting Started*

Related: `dnode` Display list of valid limNET nodes (M,U)
`eread` Transfer file from remote source (M,U)

exec **Execute a VNMR command (C)**

Syntax: `exec(command_string)`

Description: Executes the VNMR command given by the string argument.

Arguments: `command_string` is a character string constructed from a macro.

Examples: `exec($cmdstr)`
`exec(parstyle)`

See also: *VNMR User Programming*

exists **Checks if parameter, file, or macro exists and file type (C)**

Syntax: (1) `exists(name,'parameter'<,tree>):$exists`
(2) `exists(name,'file'<,permission>):$exists`
(3) `exists(name,'maclib'): $exists`
(4) `exists(name,'command'): $exists`
(5) `exists(name,'ascii'): $exists`
(6) `exists(name,'directory'): $exists`

Description: Checks for the existence of a parameter, file, command, or a macro from within a macro. It also checks if a file is an ASCII text file or a directory.

Arguments: `name` is the name of a parameter, file, command, or macro.

'parameter' checks if the parameter specified by name exists.

tree is 'global', 'current', 'processed', or 'systemglobal'. The default is 'current'. Refer to the `create` command for a more information on the types of parameter trees.

'file' checks if the file specified by name exists.

permission is a string to be used with an access permission test on the file specified by name. The default is to check only the simple existence of the file. Access permission can be identified by passing the character `r` for read permission, `w` for write permission, and `x` for execute permission. One, two, or three characters can be passed in a single argument. For example, the command `exists('/vnmr/conpar', 'file', 'rw')` checks not only that the file `/vnmr/conpar` exists, but also whether the current user has read and write access to that file.

'maclib' checks if the macro specified by name exists.

'command' checks if the command or macro specified by name exists.

'ascii' checks if the file specified by name is an ASCII text file.

'directory' checks if the file specified by name is a directory.

`$exists` is the return variable that changes according to the second argument:

- For 'parameter', `exists` returns 1 if the parameter specified by name exists in the tree specified by `tree`; otherwise, it returns 0.
- For 'file', `exists` returns 1 if the file specified by name exists with the file permission specified by `permission`; otherwise, it returns 0.
- For 'maclib', `exists` searches the macro libraries in the following order for the macro specified in name and returns 1 if the macro is in the user's `maclib` directory, returns 2 if in a directory defined by `maclibpath`, returns 3 if in a directory defined by `sysmaclibpath`, returns 4 if in the system `maclib` directory, or returns 0 if not found in any of these libraries. Only the value of the first location found is returned.
- For 'command', `exists` searches the command list and macro libraries in the following order and returns 1 if name is a VNMR command, returns 2 if it is in the user's `maclib` directory, returns 3 if in a directory defined by `maclibpath`, returns 4 if in a directory defined by `sysmaclibpath`, returns 5 if in the system `maclib` directory, or returns 0 if not found in any of these libraries. Only the value of the first location found is returned.
- For 'ascii', `exists` returns 1 if the file specified in name is an ASCII text file; otherwise it returns 0.
- For 'directory', `exists` returns 1 if the file specified in name is a directory; otherwise it returns 0.

Examples: `exists('ni', 'parameter'):$twod`
`exists('/vnmr/conpar', 'file', 'rw')`
`exists('wft', 'command'):$num`

See also: *VNMR User Programming*

Related:	<code>create</code>	Create new parameter in a parameter tree (C)
	<code>hidecommand</code>	Execute macro instead of command with same name (C)
	<code>maclibpath</code>	Path to user's macro directory (P)
	<code>which</code>	Display which macro or command is used (M)

exit **Call the vnmr_{exit} command (M)**Syntax: `exit`Description: Calls the `vnmrexit` command to exit from VNMR. As a macro, `exit` provides a user some flexibility in defining other things to do when exiting.**CAUTION:** When you exit from the VNMR user interface on your X display system, whether you are using an X terminal or a Sun computer, and whether you are using OpenWindows, CDE, or Motif, you must first exit from any copy of VNMR running on your system. Failure to do this can cause current parameter values and even current data to be lost.

Alternate: Exit VNMR button in the Secondary Main Menu.

See also: *Getting Started*Related: `vnmrexit` Exit from the VNMR system (C)**exp** **Find exponential value of a number (C)**Syntax: `exp(value)<:n>`Description: Finds the exponential value (base *e*) of a number.Arguments: `value` is a number.`n` is the return value giving the exponential value of `value`. The default is to display the exponential value in the status window.Examples: `exp(.5)`
`exp(val):exp_val`See also: *VNMR User Programming*

Related:	<code>arccos</code>	Calculate arc cosine of real number (M)
	<code>arcsin</code>	Calculate arc sine of real number (M)
	<code>arctan</code>	Calculate arc tangent of real number (M)
	<code>atan</code>	Find arc tangent of a number (C)
	<code>cos</code>	Find cosine value of an angle (C)
	<code>ln</code>	Find natural logarithm of a number (C)
	<code>sin</code>	Find sine value of an angle (C)
	<code>tan</code>	Find tangent value of an angle (C)

expactive **Determine if experiment has active acquisition (C)**Syntax: (1) `expactive<(exp_number)><:$answer>`
(2) `expactive('auto')<:$mode>`
(3) `expactive('current')<:$exp><,$user>`

Description: Determines whether an acquisition is active or pending in an experiment.

Arguments: `exp_number` is the number, from 1 to 9999, of the experiment to be checked. The default is the current experiment.`$answer` is a return value: -1 if an acquisition is not possible (e.g., the system is a data station), 0 if no acquisition active in the requested experiment, 1 if an acquisition active in that experiment, and 2 or larger if an acquisition is queued in the requested experiment (subtract 1 from the value to determine its position in the acquisition queue). With no return argument, the result is displayed on line 3.

'auto' is a keyword to check if the system is in automation mode.

`$mode` is a return value: 1 if the system is in automation mode, or 0 if otherwise. With no return argument, the result is displayed on line 3.

'current' is a keyword that determines whether an active experiment has an active acquisition command running. An experiment is still considered active if it holds up additional acquisitions during its wexp processing by the 'wait' flag. If `expactive('current')` does not have a return argument, results are displayed on line 3.

`$exp` is a return value indicating the current active experiment number: -1 if no acquisition is possible, or 0 if no acquisition is active.

`$user` is a return value indicating the user who started the acquisition. If the system is running in automation mode, `$user` is set to "auto." If no acquisition is running, `$user` is set to "nobody."

Examples: `expactive`
`expactive(3)`
`expactive(2):$active`
`expactive('auto'):$automode`

See also: *Getting Started*

expfit Make least-squares fit to polynomial or exponential curve (U)

Syntax: (From UNIX) `expfit options <analyze.inp >analyze.list`

Description: Makes a least-squares curve fitting to the data supplied in the file `analyze.inp`. For the specialized uses of **analyze**, VNMR macros (e.g., **t1**, **t2**, **kind**) are available that provide the correct file format and avoid the need for the user to select options.

In the regression mode, the type of curve fitting, ('poly1', ...) must be selected. For regression (generalized curve fitting), the regression section in the manual *User Guide: Liquids NMR* shows the input file format and describes the menus that permit option choices indirectly through menu buttons.

The following text file is an example of the file `analyze.inp` (for options T1, T2, kinetics, contact_time, and regression). (1), (2), etc. do not actually appear in the file but are used to identify lines in the description presented below the file.

```
(1) time
(2)   <amp;
(3)   2  4  linear linear

(4)   NEXT  4
(5)   1
(6)   1  1
      2  4
      3  9
      4  16
(4)   NEXT  3
(5)   2
(6)   2  5
      3  10
      4  17
```

This file contains the following information:

- (1) Optional *x*-axis title.
- (2) Optional *y*-axis title, for regression only.

(3) Line containing an integer for the number of peaks, followed by another integer for the number of pairs per peak. If regression, the x -scale type and y -scale type are also listed.

(4) In the regression mode, a line beginning with the keyword NEXT is inserted at the start of each data set when the number of pairs per peak is variable, followed by an integer for the number of pairs for the peak.

(5) An integer that indexes the peaks.

(6) Data pairs, one to a line, listed by peak.

For options `T1`, `T2`, `kinetics`, and `contact_time`, information from the file `fp.out` and from the array `xarray` are used to construct this file; therefore, it is necessary to run `fp` prior to `analyze`. For regression, this file is made by running `expl('regression')`.

For `diffusion`, `contact_time`, and, if not in regression mode, `poly1` and `poly2`, the `analyze.inp` file is slightly different:

```
(1) List of n x-y data pairs
(2) <text line>
(3) <x-values> <y-values>
(4) x y
    . . .
```

(1) Title line.

(2) Descriptive text line.

(3) Number of x values and y values.

(4) Data pairs, one to a line, are listed by peak in the following order:

```
x y (first peak, first pair)
x y (first peak, second pair)
...
x y (second peak, first pair)
...
```

`expfit` also makes a file `analyze.out` that is used by `expl` to display the results of the analysis in addition to output to the standard output, which is usually directed to `analyze.list`.

Arguments: `options` can be any of the following:

`T1` sets T_1 analysis. This value is the default.

`T2` sets T_2 analysis.

`kinetics` sets kinetics analysis with decreasing peak height.

`increment` sets kinetics analysis with increasing peak height.

`list` sets an extended listing for each peak.

`diffusion` sets a special analysis for diffusion experiments.

`contact_time` sets a special analysis for solids cross-polarization spin-lock experiments.

`regression` sets regression mode, providing generalized curve fitting with choices `poly1`, `poly2`, `poly3`, and `exp`:

- `poly0` calculates the mean.
- `poly1` sets a linear fitting.
- `poly2` sets a quadratic fitting.
- `poly3` sets a cubic curve fitting.
- `exp` sets an exponential curve fitting.

Examples: (From UNIX) `expfit d2 T1 list <analyze.inp >analyze.out`
 (From UNIX) `expfit regression exp list <analyze.inp >analyze.out`

See also: *User Guide: Liquids NMR*

Related: **analyze** Generalized curve fitting (C)
expl Display exponential or polynomial curves (C)
fp Find peak heights (C)
kind Kinetics analysis, decreasing intensity (M)
t1 T_1 exponential analysis (M)
t2 T_2 exponential analysis (M)

expl Display exponential or polynomial curves (C)

Syntax: `expl(<options,>line1,line2,...)>`

Description: Displays exponential curves resulting from T_1 , T_2 , or kinetic analyses. Also displays polynomial curves from diffusion or other types of analysis. The parameters **sc**, **wc**, **sc2**, and **wc2** control the size of the display.

In general, the first time `expl` is displayed, it calculates appropriate limits for the two axes. A subsequent call to `expl`, while a previous `expl` is displayed on the graphics screen, uses the axis scaling that displayed `expl`. To have the new `expl` recalculate its own axis limits and not use those currently displayed, call the **autoscale** macro before executing `expl`. Alternately, the axis limit for the `expl` display can be specified using the **scalelimits** macro.

Arguments: `options` can be any of the following:

- 'regression' is a keyword signifying the beginning of generalized curve fitting. `expl` displays the data in the file `regression.inp` as unconnected points and also uses `regression.inp` to create the file `analyze.inp`, which serves as input to **analyze** for curve fitting.
- 'linear', 'square', and 'log' are keywords for display of the data points against a square or logarithmic axis scale, with the exception of the results from regression. The first keyword controls the x -axis scale, the second the y -axis. The default is 'linear'.
- 'link' is a keyword to link the data points rather than a display of the theoretical curve.
- 'nocurve' is a keyword to produce a plot of data points only.
- 'tinysymbol' is a keyword to display small-scale data point symbols.
- 'nosymbol' is a keyword to produce a plot of the curve only.
- 'noclear' is a keyword to not erase the graphics screen before drawing the plot. This prevents the graphics screen from being cleared of data.
- 'oldbox' is a keyword to plot an additional curve on an existing plot. Only the first data set in the file `analyze.out` is plotted. The box and scale description is derived from the file `expl.out` in the current experiment. When the 'oldbox' option is used, a second argument is necessary to identify the curve number and data point symbol to represent the data. This second argument is a number from 1 to 6.
- 'file' is a keyword that, when followed by a file name, makes that file replace the file `analyze.out` as the input to `expl`.

`line1`, `line2`, ... specify the curves to be displayed. The default is to display the first eight curves (if that many exist) along with data points.

Examples: `expl`
`expl(1,3,6)`
`expl('oldbox',5)`
`expl('regression')`
`expl('regression',4,5)`

See also: *User Guide: Liquids NMR*

Related: `analyze` Generalized curve fitting (C)
`autoscale` Resume autoscaling after limits set by `scalelimits` (M)
`expfit` Make least squares fit to polynomial or exponential curve (C)
`pexpl` Plot exponential or polynomial curves (C)
`sc` Start of chart (P)
`sc2` Start of chart in second direction (P)
`scalelimits` Set limits for scales in regression (M)
`wc` Width of chart (P)
`wc2` Width of chart in second direction (P)

expladd Add another diffusion analysis to current display (M)

Applicability: Systems with the diffusion option.

Syntax: `expladd(integral_region)`

Description: Adds results of another diffusion analysis to the currently displayed results.

Arguments: `integral_region` specifies the number of the region whose results are to be added to the existing graph.

Examples: `expladd(1)`

See also: *User Guide: Liquids NMR*

Related: `expl` Display exponential or polynomial curves (C)
`pexpl` Plot exponential or polynomial curves (C)
`pexpladd` Add another diffusion analysis to current plot (M)

explib Display experiment library (M)

Syntax: `explib`

Description: Displays the currently available experiment files. For each experiment, `explib` displays the name of the experiment and its subexperiments, whether an acquisition is active or its position in the acquisition queue, the current size of the experiments, the pulse sequence currently active in the experiments, and the first 50 characters of the text file in the experiment. `explib` also displays a message if the system is in automation mode.

Alternate: **Library** button in the Workspace Menu.

See also: *Getting Started; User Guide: Liquids NMR*

explist Display current experiment chain and approx. time for each (M)

Syntax: `explist`

See also: Displays approximate time for each experiment in a chained experiment.

Related: `autotime` Display approximate time for automation (M)

explog Display log file for experiment (M)

Syntax: `explog`

Description: Displays the log file for an experiment. This file includes when the experiment started, any acquisition errors that may have occurred, and when the experiment finished. Each acquisition generates this information, which is stored in the experiment's `acqfil` directory in a text file named `log`.

See also: *Getting Started*

exptime **Display experiment time (C)**

Syntax: `exptime<(sequence)><:$seconds>`

Description: Estimates the acquisition time for an experiment, based on the parameters used in the current experiment, and displays the time in the format `hh:mm:ss`. The `time` macro uses `exptime` to determine the time of an experiment.

Arguments: `sequence` is a pulse sequence that exists in the `seqlib` directory. If this argument is used, `exptime` estimates the acquisition time for the specified sequence. The default is the current value of `seqlib`.

`$seconds` is a return argument with the number of seconds estimated for the experiment. If this argument is used, the time display is suppressed.

Examples: `exptime`
`exptime('apt')`
`exptime:$etime`
`exptime('noesy'):$est_time`

See also: *Getting Started*

Related: `time` Display experiment time or recalculate number of transients (M)

F

f **Set display parameters to full spectrum (C)**

Syntax: `f`

Description: Sets up the `sp` and `wp` display parameters for a full display of a 1D spectrum. If an FID is displayed, the parameters `sf` and `wf` are set for a full display. In multidimensional data sets, the parameters for both displayed dimensions are set up. For 2D data sets, the parameters `sp`, `wp`, `sp1`, and `wp1` would be set. For planes of higher dimensional data sets, the appropriate two groups of `sp-wp`, `sp1-wp1`, and `sp2-wp2`, parameter pairs are set.

See also: *Getting Started*

Related:	<code>sf</code>	Start of FID (P)
	<code>sp</code>	Start of plot in directly detected dimension (P)
	<code>sp1</code>	Start of plot in 1st indirectly detected dimension (P)
	<code>sp2</code>	Start of plot in 2nd indirectly detected dimension (P)
	<code>wf</code>	Width of FID (P)
	<code>wp</code>	Width of plot in directly detected dimension (P)
	<code>wp1</code>	Width of plot in 1st indirectly detected dimension (P)
	<code>wp2</code>	Width of plot in 2nd indirectly detected dimension (P)

f19 **Automated fluorine acquisition (M)**

Syntax: `f19<(solvent)>`

Description: Prepares parameters for automatically acquiring a standard ^{19}F spectrum. The parameter `wexp` is set to 'procplot' for standard processing. If `f19` is used as the command for automation via the `enter` program, then the macro `au` is supplied automatically and should not be entered on the MACRO line of the `enter` program. However, it is possible to customize the standard `f19` macro on the MACRO line by following it with additional commands and parameters. For example, `f19 nt=1` uses the standard `f19` setup but with only one transient.

Arguments: `solvent` is the name of the solvent. In automation mode, the solvent is supplied by the `enter` program. The default is 'CDCl3'

Examples: `f19`
`f19('DMSO')`

See also: *Getting Started; User Guide: Liquids NMR*

Related:	<code>au</code>	Submit experiment to acquisition and process data (M)
	<code>enter</code>	Enter sample information for automation run (C)
	<code>f19p</code>	Process 1D fluorine spectra (M)
	<code>procl1d</code>	Processing macro for simple (non-arrayed) 1D spectra (M)
	<code>procplot</code>	Automatically process FIDs (M)
	<code>wexp</code>	When experiment completes (P)

f19p **Process 1D fluorine spectra (M)**

Syntax: `f19p`

Description: Processes non-arrayed 1D fluorine spectra using a set of standard macros. `f19p` is called by `procl1d`, but can also be used directly. Fully automatic processing

(up to a point where a spectrum could be plotted) is provided: Fourier transformation (using preset weighting functions), automatic phasing (**aphx** macro), select integral regions (**hregions** macro), adjust integral size (**integrate** macro), vertical scale adjustment (**vsadjc** macro), avoiding excessive noise (**noislm** macro), threshold adjustment (if required, **thadj** macro), and referencing to the TMS signal, if present (**tmsref** macro).

See also: *Getting Started; User Guide: Liquids NMR*

Related:	aphx	Perform optimized automatic phasing (M)
	f19	Automated fluorine acquisition (M)
	hregions	Select integral regions for proton spectra (M)
	integrate	Automatically integrate 1D spectrum (M)
	noislm	Avoids excessive noise (M)
	procl1d	Processing macro for simple (non-arrayed) 1D spectra (M)
	thadj	Adjust threshold (M)
	tmsref	Reference spectrum to TMS line (M)
	vsadjh	Adjust vertical scale for proton spectra (M)

f1coef **Coefficient to construct F1 interferogram (P)**

Description: Holds the coefficient to construct an F1 interferogram for 2D and 3D transformation. Coefficients are used by the **ft2da** and **ft3d** macros. If **f1coef** has a null value, **ft2da** uses the “standard” coefficients. **f1coef** is created by the **par2d** macro.

Values: Series of coefficients, separated by spaces (not a comma), and stored as a string variable. For example, the coefficient for standard States-Hypercomplex data set is **f1coef='1 0 0 0 0 0 -1 0'**.

See also: *User Guide: Liquids NMR*

Related:	f2coef	Coefficient to construct F2 interferogram (P)
	ft2da	Fourier transform phase-sensitive data (M)
	ft3d	Perform a 3D Fourier transform on a 3D FID data set (M,U)
	make3dcoef	Make 3D coefficients file from 2D coefficients (M)
	par2d	Create 2D acquisition, processing, display parameters (M)

f2coef **Coefficient to construct F2 interferogram (P)**

Description: Holds the coefficient to construct an F2 interferogram for 2D and 3D transformation. Coefficients are used by the **ft2da('ni2')** and **ft3d** macros. If **f2coef** has a null value, **ft2da('ni2')** uses the “standard” coefficients. **f2coef** is created by the **par3d** macro.

Values: Series of coefficients, separated by spaces (not a comma), and stored as a string variable. For example, the coefficient for standard States-Hypercomplex data set is **f2coef='1 0 0 0 0 0 -1 0'**.

See also: *User Guide: Liquids NMR*

Related:	f1coef	Coefficient to construct F1 interferogram (P)
	ft2da	Fourier transform phase-sensitive data (M)
	ft3d	Perform a 3D Fourier transform on a 3D FID data set (M,U)
	make3dcoef	Make 3D coefficients file from 2D coefficients (M)
	par3d	Create 3D acquisition, processing, display parameters (M)

fattn **Fine attenuator (P)**

Applicability: All systems except *MERCURY* series and *GEMINI 2000*.

Description: Configuration parameter for whether the current rf channel has a fine attenuator. The value is set using the label Fine Attenuator in the CONFIG window (opened from `config`).

For *MERCURY-VX* only, `fatten` indicates which high band transmitter is present.

Values: 0 specifies the fine attenuator is not present on the channel (Not Present choice in CONFIG window).

4095 specifies the fine attenuator is present on the channel (Present choice in CONFIG window).

See also: *VNMR and Solaris Software Installation*

Related: `config` Display current configuration and possibly change it (M)
`dpwrf` First decoupler fine power (P)
`tpwrf` Observe transmitter fine power (P)

fb Filter bandwidth (P)

Description: Sets the bandwidth of the audio filters, which prevents noise of higher frequency than the spectral limits from “folding in” to the spectrum. Because the transmitter is in the center of the spectrum, the range of audio frequencies that must be filtered out is half the spectral width `sw` (e.g., for a spectral width of 4000 Hz, frequencies higher than ± 2000 Hz should be filtered out). The audio filters have some attenuation at frequencies lower than their nominal cutoff frequency, which is the frequency at which signals have been attenuated by 3 dB (50%). This impacts on quantitative accuracy near the edges of the spectrum so that, except on *GEMINI 2000* $^1\text{H}/^{13}\text{C}$ systems, the standard value of `fb` is 10% more than half of `sw`.

`fb` is automatically changed whenever the spectral width `sw` is changed and thus is normally not a user-entered parameter. For example, typing `sw=4000` automatically sets `fb=2200`, which is 10% more than 2000 Hz. After changing the value of `sw`, `fb` can be changed.

Values: On *UNITYINOVA*, if `sw` is 500,000 or less: 1000 to 256000 Hz, 1000-Hz steps.

On *UNITYINOVA*, if `sw` is greater than 500,000: 256 kHz, 1 MHz.

On *MERCURY-VX* and *MERCURY*: 1 to 25 kHz and 55 kHz. Actual values are a non-linear set, entered in steps of 200, and rounded to the larger available value.

On *UNITYplus*, *UNITY*, and *VXR-S*, if `sw` is 100,000 or less:

- For *UNITYplus*, *UNITY*, most *VXR-S*: 200 to 51200 Hz, 200-Hz steps.
- For some *VXR-S* systems: 100 to 49500 Hz, 100-Hz steps.

On *UNITYplus*, *UNITY*, and *VXR-S*, if `sw` is greater than 100,000:

- For *UNITYplus*: 256 kHz, 1 MHz.
- For *UNITY* and *VXR-S*: 170 kHz, 300 kHz, 700 kHz, 1 MHz, 2 MHz.

On the $^1\text{H}/^{13}\text{C}$ *GEMINI 2000* (`pfiltr='n'`), programmable filters on the observe receiver are not present. `fb` is set to 1500 or 7500 on a 200-MHz *GEMINI 2000*, or set to 2250 or 9000 on a 300-MHz *GEMINI 2000*.

On the broadband *GEMINI 2000* (`pfiltr='y'`), programmable filters on the observe receiver are present. `fb` ranges from 200 Hz to 51.2 kHz, in 200-Hz steps.

See also: *Getting Started*

Related: `pfiltr` Programmable filters (P)
`sw` Spectral width in directly detected dimension (P)
`mrfb` Set the filter bandwidths for multiple receivers (P)

fbc **Apply baseline correction for each spectrum in an array (M)**

Syntax: `fbc`

Description: Applies **bc** -type baseline correction to all the spectra in an array. The partial integral mode should be used to set integral regions to include all significant signals, while leaving blank as large an area of baseline as is possible.

See also: *User Guide: Liquids NMR*

Related: **dosy** Process DOSY experiments (M)

fdfgluer **Make FDF file from header and data parts (U)**

Syntax: (1) `fdfgluer <-align> header_file <data_file
<output_file>>`
 (2) `fdfgluer -infile template <-offset n>
<-align> header_file`
 (3) `fdfgluer -vnmrfile fname -outfile template
<-traces n> <-align> header_file`

Description: Takes an FDF (flexible data format) header file defining a set of data and data from a file, files, or standard input, and combines them to form an FDF data file. Using syntax 1 attaches a header to a raw data file. If the `data_file` argument is given (rather than being taken from standard input), a checksum is calculated and appended to the header.

Using syntax 2 takes the data from a group of raw data files whose names are `template1`, `template2`, etc. These data files can have fixed length headers, which will be ignored.

Using syntax 3 takes data from a VNMR format data file, such as a FID file.

Arguments: `header_file` is the name of the header file created or edited by the user.

`data_file` is the name of file containing data for a FDF file. If this argument is not present, `fdfgluer` takes the data from the standard input.

`output_file` is the name of the FDF file created. If this argument is not present, `fdfgluer` puts the FDF file to the standard output.

`-align` is a numerical argument giving the size of words that the data should be aligned on. For example, `-8` ensures that the length of the header is a multiple of 8 bytes.

`-infile template` gives the base name of the group of files from which to take data. `template` can be a path. `fdfgluer` will read data from files named `template1`, `template2`, `template3`, etc. in numerical order until the next sequential file name is not found.

`-offset n` gives the number of bytes of header in the data files. The first `n` bytes of each data file are ignored.

`-vnmrfile fname` specifies the name of a VNMR format data file to use for the input data.

`-outfile template` specifies the base name of output files to be written using syntax 3. The template should have a “#” somewhere in it. The output files will substitute a serial number (0001, 0002,...) for the #. For example, `-outfile myrat#.fdf` writes data to output files `myrat0001.fdf`, etc.

`-traces n` gives the number of traces to put in each output file in syntax 3.

See also: *User Guide: Imaging, VNMR User Programming*

Related: **fdfsplit** Divide FDF file into header and data parts (U)

fdfsplit **Divide FDF file into header and data parts (U)**

Applicability: Systems with imaging capabilities.

Syntax: `fdfsplit output_file data_file header_file`

Description: Takes an FDF (Flexible Data Format) file and splits it into its data and header parts. Note that the header may still have a checksum value—that value should be removed after the split has completed.

Arguments: `output_file` is the name of the FDF file to be split.

`data_file` is the file name to be given to the data part.

`header_file` is the file name to be given to the header part.

See also: *VNMR User Programming, User Guide: Liquids*

Related: [fdfgluer](#) Make FDF file from header and data parts (C)

fdm1 **Set, write 1D FDM parameters, run FDM (M)**

Syntax: `fdm1<(filename<,n1, v1<, n2, v2<...>>>>`
or
`fdm1 (i)` for the *i*-th trace

Description: Sets 1D Filter Diagonalization Method (FDM) parameters to the default values, writes the parameters to the `curexp/datadir/fdm1.inparm` file, and runs a stand-alone C++ program (`/vnmr/bin/fdm1d`).

Arguments: `filename` is the FID file; the default is `curexp+'acqfil/fid'`.

`n1, n2...` is one or more following variable names (the order is arbitrary):

<code>axis</code>	<code>-1</code> (default) to reverse the spec.
<code>cheat</code>	No cheat if <code>cheat=1</code> , lines are narrower if <code>cheat<1</code> .
<code>cheatmore</code>	No cheatmore if <code>cheatmore=0</code> .
<code>error</code>	Error threshold for throwing away poles.
<code>fidfmt</code>	FID format: VNMR or ASCII.
<code>fdm</code>	1 for FDM; <code>-1</code> for Digital or Discrete Fourier Transform.
<code>fn_SplD</code>	Spectrum file; default is <code>curexp/datadir/fdm1.parm</code> .
<code>Gamm</code>	Smoothing width (line broadening).
<code>Gcut</code>	Maximum width for a pole.
<code>idat</code>	Data type of ASCII FID file <code>-4</code> for complex data, ignored if data is in VNMR format.
<code>i_fid</code>	The <i>i</i> -th trace of the FID.
<code>kcoef</code>	If <code>kcoef > 0</code> , use 'complicated' <code>dk(k)</code> . <code>-1</code> is always preferred.
<code>Nb</code>	Number of basis functions in a single window.
<code>Nbc</code>	Number of coarse basis vectors.
<code>Npower</code>	Number of spectrum data points.
<code>Nsig</code>	Number of points to use.
<code>Nskip</code>	Number of points to skip.
<code>par</code>	Line list file; default is <code>curexp/datadir/fdm1.parm</code>
<code>rho</code>	<code>rho=1</code> is optimal.
<code>specfmt</code>	Spec format: VNMR or ASCII.

`spectyp` Spectrum type: complex (default), real imag, or abs.
`ssw` A test parameter.
`t0` Delay of the first point.
`theta` Overall phase of FID (rp in radians).
`wmax` Maximum spectrum frequency in hertz.
`wmin` Minimum spectrum frequency in hertz.
`v1`, `v2`... is the value for the variable(s).

Examples: `fdml('cheat',0.8)`
`fdml('Nsig',3000,'Nb',20,1'Gamm',0.5)`

See also: *User Guide: Liquids*

fiddc3d **3D time-domain dc correction (P)**

Applicability: All systems; however, although `fiddc3d` is available on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems, such systems can only process 3D data and cannot acquire 3D data.

Description: Sets whether a 3D time-domain dc correction occurs. If `fiddc3d` does not exist, it is created by the macro `par3d`. The time-domain dc correction occurs immediately after any linear prediction operations and before all other operations on time-domain data.

Values: A three-character string. The default value is 'nnn'.

- The first character refers to the f_3 dimension (`sw`, `np`, `fn`), the second character refers to the f_1 dimension (`sw1`, `ni`, `fn1`), and the third character refers to the f_2 dimension (`sw2`, `ni2`, `fn2`).
- Each character may take one of two values: 'n' for no time-domain dc correction along the relevant dimension, and 'y' for time-domain dc correction along the relevant dimension.

See also: *User Guide: Liquids NMR*

Related: `fn` Fourier number in directly detected dimension (P)
`fn1` Fourier number in 1st indirectly detected dimension (P)
`fn2` Fourier number in 2nd indirectly detected dimension (P)
`ft3d` Perform a 3D Fourier transform (M)
`ni` Number of increments in 1st indirectly detected dimension (P)
`ni2` Number of increments in 2nd indirectly detected dimension (P)
`np` Number of data points (P)
`par3d` Create 3D acquisition, processing, display parameters (C)
`ptspec3d` Region-selective 3D processing (P)
`specdc3d` 3D spectral dc correction (P)
`sw` Spectral width in directly detected dimension (P)
`sw1` Spectral width in 1st indirectly detected dimension (P)
`sw2` Spectral width in 2nd indirectly detected dimension (P)

fiddle **Perform reference deconvolution (M)**

Syntax: `fiddle(option<,file><,option<,file>><,start>
<,finish><,increment>)`

Description: Performs reference deconvolution using a reference signal with known characteristics to correct instrumental errors in experimental 1D or 2D spectra.

Arguments: `option` can be any of the following:

- 'alternate' is a keyword specifying the alternate reference phase +- (+ - means? – Editor) (for phase sensitive gradient 2D data).
- 'autophase' is a keyword specifying to automatically adjust the phase of the reference signal.
- 'displaycf' is a keyword specifying to stop at the display of the correction function.
- 'fittedbaseline' is a keyword specifying to use cubic spline baseline correction defined by the choice of integral regions.
- 'invert' is a keyword specifying to invert the corrected difference spectrum/spectra.
- 'noaph' is a keyword specifying not to automatically adjust zero order phase of the reference region.
- 'nodc' is a keyword specifying not to use dc correction of reference region.
- 'noextrap' is a keyword specifying not to use extrapolated dispersion mode.
- 'nohilbert' is a keyword specifying not to use Hilbert transform algorithm and to use extrapolated dispersion mode reference signal unless 'noextrap' is also used as an option.
- 'normalise' is a keyword specifying to keep corrected spectrum integrals equal to that of the first spectrum.
- 'satellites' is a keyword specifying to use satellites defined in file in ideal reference region; file should be in /vnmr/satellites, and should immediately follow 'satellites' in the argument list.
- 'stop1' is a keyword specifying to stop at display of experimental reference FID.
- 'stop2' is a keyword specifying to stop at display of correction function.
- 'stop3' is a keyword specifying to stop at display of corrected FID.
- 'stop4' is a keyword specifying to stop at display of first corrected FID.
- 'verbose' is a specifying keyword to display information about processing in the main window.
- 'writecf' is a keyword specifying to write the correction function to file; the argument file must immediately follow 'writecf'.
- 'writefid' is a keyword specifying to write out corrected FID to file; if file does not begin with /, it is assumed to be in the current working directory. In the argument list, file should immediately follow 'writefid'.

file is the name of the file used with the 'satellites' and 'writefid' options.

start and finish are the indices of the first and last array elements to be processed. increment specifies the steps in which the index is to be incremented. The default is to process all the transformed spectra in an array.

See also: *User Guide: Liquids NMR*

Related:	fiddled	Perform reference deconvolution subtracting alternate FIDs
	fiddleu	Perform reference deconvolution subtracting successive FIDs
	fiddle2d	Perform 2D reference deconvolution
	fiddle2D	Perform 2D reference deconvolution

`fiddle2dd` Perform 2D reference deconvolution subtracting alternate FIDs
`fiddle2Dd` Perform 2D reference deconvolution subtracting alternate FIDs

fiddled Perform reference deconvolution subtracting alternate FIDs (C)

Description: Produces the corrected difference between successive spectra. Refer to the description of `fiddle` for details.

See also: *User Guide: Liquids NMR*

Related: `fiddle` Perform reference deconvolution

fiddleu Perform reference deconvolution subtracting successive FIDs (C)

Description: Produces corrected differences between successive FIDs and the first FID. Refer to the description of `fiddle` for details.

See also: *User Guide: Liquids NMR*

Related: `fiddle` Perform reference deconvolution

fiddle2d Perform 2D reference deconvolution (C)

Description: Functions the same as the `fiddle` program except `fiddle2d` performs 2D reference deconvolution. Refer to the description of `fiddle` for details.

See also: *User Guide: Liquids NMR*

Related: `fiddle` Perform reference deconvolution

fiddle2D Perform 2D reference deconvolution (C)

Description: Functions the same as the `fiddle` program except `fiddle2D` performs 2D reference deconvolution. Refer to the description of `fiddle` for details.

See also: *User Guide: Liquids NMR*

Related: `fiddle` Perform reference deconvolution

fiddle2dd Perform 2D reference deconvolution subtracting alternate FIDs (C)

Description: Functions the same as the `fiddle` program except `fiddle2dd` performs 2D reference deconvolution. Refer to the description of `fiddle` for details.

See also: *User Guide: Liquids NMR*

Related: `fiddle` Perform reference deconvolution

fiddle2Dd Perform 2D reference deconvolution subtracting alternate FIDs (C)

Description: Functions the same as the `fiddle` program except `fiddle2Dd` performs 2D reference deconvolution. Refer to the description of `fiddle` for details.

See also: *User Guide: Liquids NMR*

Related: `fiddle` Perform reference deconvolution

fidpar Add parameters for FID display in current experiment (M)

Syntax: `fidpar`

Description: Creates the FID display parameters `axisf`, `crf`, `deltaf`, `dotflag`, `vpf`, and `vpfi` in the current experiment. Use `fidpar` to define these parameters in old parameter sets (they are already defined in new parameter sets).

See also: *Getting Started*

Related:	<code>addpar</code>	Add selected parameters to current experiment (M)
	<code>axisf</code>	Axis label for FID displays and plots (P)
	<code>crf</code>	Current time domain cursor position (P)
	<code>deltaf</code>	Difference of two time cursors (P)
	<code>dotflag</code>	Display FID as connected dots (P)
	<code>vpf</code>	Current vertical position of FID (P)
	<code>vpfi</code>	Current vertical position of imaginary FID (P)

fifolpsize **FIFO loop size (P)**

Applicability: All systems except *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*. `fifolpsize` is not used on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*. The correct value is for 512 the *GEMINI 2000* and 2048 for the *MERCURY-VX* and *MERCURY*. The `config` program sets this value.

Description: Configuration parameter for the size of the FIFO loop. The size depends on which controller board is present on the system—the Output board, the Acquisition Controller board, or the Pulse Sequence Controller board (refer to the description of the `acquire` statement in the manual *VNMR User Programming* for information on identifying the boards). The value is set using the label Fifo Loop Size in the CONFIG window (opened by `config`).

Values: 63 is used with the Output board (Part No. 953520).
1024 is used with the Acquisition Controller board (Part No. 969204).
2048 is used with the Pulse Sequence Controller board (Part No. 992560).

See also: *VNMR and Solaris Software Installation*

Related:	<code>config</code>	Display current configuration and possibly change it (M)
----------	---------------------	--

fixgrd **Convert gauss/cm value to DAC (M)**

Syntax: `fixgrd(gradient_value):parameter`

Description: Uses the `gcal` value in the probe table to return the DAC value for a specified gradient strength.

Arguments: `gradient_value` is the required gradient strength in gauss/cm.
`parameter` is any local variable or VNMR variable.

Examples: `fixgrd(20):gzlv1`

Related:	<code>gcal</code>	Gradient calibration constant (P)
----------	-------------------	-----------------------------------

file **File name of parameter set (P)**

Description: Contains the file name of the parameter set returned by a `rt` or `rtp` command. This parameter is reset when the `go` command is issued. If the system is not in automation mode (`auto='n'`), `file` is reset to the 'exp' value. If the system is in automation mode (`auto='y'`), `file` is set to the path of the directory where the data is stored.

See also: *Getting Started*

Related:	<code>auto</code>	Automation mode active (P)
	<code>go</code>	Submit experiment to acquisition (C)
	<code>rt</code>	Retrieve FID (C)
	<code>rtp</code>	Retrieve parameters (C)

files **Interactively handle files (C)**

Syntax: `files<(files_menu)>`

Description: Brings up the interactive file handling program. With this program, the mouse and keyboard are used to copy, delete, rename, change directories, and load and save experiment data. The `files` command uses the graphics window to display file names. A mouse clicked on a file name selects it and the file name is displayed in reverse video. Various operations can be conducted on one or more selected files. The menus used for the `files` program are placed in the standard `menulib` directories. Refer to the manual *Getting Started* for more information on using menus, and refer to the manual *VNMR User Programming* for information on programming menus.

Arguments: `files_menu` is the `files` menu to control the menu buttons; the default menu is 'files_main' or the last active `files` menu.

Alternate: File button in the Main Menu.

Examples: `files`
`files('files_dir')`

See also: *VNMR User Programming*

Related: `filesinfo` Return files display information (C)
`tape` Control tape options of `files` program (P)

filesinfo **Return file information for files display (C)**

Syntax: (1) `filesinfo('number'):$number_files`
(2) `filesinfo('name'<,file_number>):$file`
(3) `filesinfo('redisplay')`

Description: Allows access to the list of files selected from the `files` interactive display. `filesinfo` is normally used only by the macros that implement the menu functions of the file system and not entered from the keyboard. The command will not execute unless the `files` program is active.

Arguments: 'number' is a keyword to return the number of files selected in the `files` display, or 0 if no files have been selected.

`$number_files` is the return variable when 'number' is used.

'name' is a keyword to return a list of file names selected in the `files` display.

`file_number` is a number following the 'name' keyword to return only the file name in the list given by `file_number`.

`$file` is a string variable that returns the file name when 'name' is used.

'redisplay' is a keyword that causes the current contents of the directory to be displayed. This display is useful after making changes in the directory, such as deleting or creating a file.

See also: *VNMR User Programming*

Related: `files` Interactively handle files (C)

filter **Gaussian low-pass filter for image processing (M)**

Applicability: Systems with imaging capabilities.

Syntax: `filter(strength)`

Description: Sets the processing parameters `gf`, `gfs`, `gf1`, and `gfs1` to create a low-pass filter for improving the signal-to-noise (S/N) ratio in images. S/N improvement

is achieved at the expense of resolution. The results of the parameter setting performed by `filter` can be applied to the image using the `wft2d` command. The parameters `gf`, `gfs`, `gf1`, and `gfs1` are calculated to center the filter in both the t_1 and t_2 dimensions. The filter setting can be bypassed with the `ft2d` command. A side effect of `filter` is to reset the maximum, minimum and step values for all of the Gaussian processing parameters. This is to allow precise setting of the filter.

Arguments: `strength` is a number from 0 to 100 that represents the attenuation, in dB, applied to the signal at the edges of the sampling windows in the t_1 and t_2 dimensions. For example, `strength` set to 6 produces a Gaussian filter for t_1 and t_2 that reduces the signal at the edge of the sampling window by half (i.e., a 6-dB attenuation). If `strength` is set to 0, `gf`, `gfs`, `gf1`, and `gfs1` are set to 'n', effectively turning the parameters off.

Examples: `filter(10)`

See also: *User Guide: Imaging*

Related:	<code>ft2d</code>	Fourier transform 2D data (C)
	<code>gf</code>	Gaussian function on directly detected dimension (P)
	<code>gf1</code>	Gaussian function on 1st indirectly detected dimension (P)
	<code>gfs</code>	Gaussian shift constant on directly detected dimension (P)
	<code>gfs1</code>	Gaussian shift constant on 1st indirectly detected dimension (P)
	<code>wft2d</code>	Weight and Fourier transform 2D data (C)

filtfile **File of FIR digital filter coefficients (P)**

Description: Specifies name of a file of FIR (finite impulse response) digital filter coefficients. This file is a text file with one real filter coefficient per line (complex filters are not supported). If the parameter `filtfile` does not exist in the current experiment, enter `addpar('downsamp')` or `addpar('oversamp')` to add it. Entering `addpar('downsamp')` creates the digital filtering and downsampling parameters `downsamp`, `dscoef`, `dsfb`, `ds1sfrq`, and `filtfile`. Similarly, entering `addpar('oversamp')` creates digital filtering and oversampling parameters `def_osfilt`, `filtfile`, `oscoef`, `osfb`, `osfilt`, `os1sfrq`, and `oversamp`.

Values: File name. The file must be in the user's `vnmr/sys/filtlib` directory.

See also: *Getting Started*

Related:	<code>addpar</code>	Add selected parameters to current experiment (M)
	<code>def_osfilt</code>	Default value of <code>osfilt</code> (P)
	<code>downsamp</code>	Downsampling factor applied after digital filtering (P)
	<code>dscoef</code>	Digital filter coefficients for downsampling (P)
	<code>dsfb</code>	Digital filter bandwidth for downsampling (P)
	<code>ds1sfrq</code>	Bandpass filter offset for downsampling (P)
	<code>oscoef</code>	Digital filter coefficients for oversampling (P)
	<code>osfb</code>	Digital filter bandwidth for oversampling (P)
	<code>osfilt</code>	Oversampling filter for real-time DSP (P)
	<code>os1sfrq</code>	Bandpass filter offset for oversampling (P)
	<code>oversamp</code>	Oversampling factor for acquisition (P)
	<code>pards</code>	Create additional parameters used for downsampling (M)
	<code>paros</code>	Create additional parameters used for oversampling (M)

fitplot **Adjust plot parameters (M)**

Applicability: Systems with imaging capabilities.

Syntax: `fitplot`

Description: If the parameter `axis` is set to 'cc', `fitplot` uses an algorithm that adjusts the display and subsequent plot to present the image in the largest possible format for the current conditions specified by the `wcmax`, `wc2max`, and `trace` parameters. For example, `fitplot` could be entered as `fitplot imageprint page` for plotting. This algorithm leaves a column of 50 mm for plotting parameters down the left-hand edge of the paper. `fitplot` also has other algorithms for different settings of the `axis` and `ni` parameters.

See also: *User Guide: Imaging*

Related:	<code>axis</code>	Axis labels for displays and plots (P)
	<code>imageprint</code>	Plot noninteractive gray scale image (M)
	<code>ni</code>	Number of increments in 1st indirectly detected dimension (P)
	<code>page</code>	Submit plot and change plotter page (C)
	<code>trace</code>	Mode for <i>n</i> -dimensional data display (P)
	<code>wcmax</code>	Maximum width of chart (P)
	<code>wc2max</code>	Maximum width of chart in second direction (P)

`fitspec` Perform spectrum deconvolution (C, U)

Syntax: (From VNMR) `fitspec(<'usell'><,><'setsfreq'>)>`
 (From UNIX) `fitspec`

Description: Fits experimental data to Lorentzian and/or Gaussian lineshapes. `fitspec` uses as a starting point data in a file `fitspec.inpar`, which must be prepared prior to performing the calculation. This file contains the frequency, intensity, linewidth, and (optionally) the Gaussian fraction of the lineshape. Any number followed by an asterisk (*) is held fixed during the calculation; all other parameters are varied to obtain the best fit. `fitspec` creates a file `fitspec.data`, which is a text representation of the spectral data (that part of the spectrum between `sp` and `sp+wp`). After the calculation is finished, the results of the fit are contained in a file `fitspec.outpar`, with a format identical to `fitspec.inpar`.

It is often useful to use the output from a deconvolution as the input to a spin simulation to ensure the most accurate possible frequencies for the spin simulation calculation. For this reason, the frequencies and amplitudes of the calculated lines in a deconvolution are automatically stored in the parameters `slfreq` and `slamp`, respectively, from where they can serve as input to an iterative spin simulation. If the spin system is defined *after* a deconvolution is performed, this information is lost (`slfreq` and `slamp` are reset). In this case, `fitspec('setslfreq')` can be used to copy the information from `fitspec.outpar` back into `slfreq` and `slamp`. This is not necessary if you define the spin system before performing the deconvolution (you need not perform the entire spin simulation, only define the spin system).

Arguments: 'usell' is a keyword to prepare the file `fitspec.inpar` from the last line listing (stored in `llfrq` and `llamp`). All lines are set to have a linewidth of `slw` and a fixed Gaussian fraction of 0. If another starting point is desired, this file can be edited with a text editor. Alternatively, the macro `usemark` may be used.

'setslfreq' is a keyword to copy the information from the file `fitspec.outpar` back into the parameters `slfreq` and `slamp`.

Examples: `fitspec`
`fitspec('usell')`
`fitspec('setslfreq')`

See also: *User Guide: Liquids NMR*

Related:	<code>llamp</code>	List of line amplitudes (P)
	<code>llfrq</code>	List of line frequencies (P)
	<code>setgauss</code>	Set a Gaussian fraction for lineshape (M)
	<code>slamp</code>	Measured line amplitudes (P)
	<code>slfreq</code>	Measured line frequencies (P)
	<code>sp</code>	Start of plot (P)
	<code>usemark</code>	Use “mark” output as deconvolution starting point (M)
	<code>wp</code>	Width of plot (P)

fixpar **Correct parameter characteristics in experiment (M)**

Syntax: `fixpar`

Description: After bringing parameters into the current experiment with `convert`, `rt`, `rtp`, or `rtv`, `fixpar` is automatically executed. `fixpar` updates old parameter characteristics and reconciles parameter differences due to the hardware on the spectrometer. If a macro `userfixpar` exists, `fixpar` runs it also. This allows an easy mechanism to customize parameter sets.

See also: *Getting Started*

Related:	<code>convert</code>	Convert data set from a VXR-style system (C)
	<code>fixpar3rf</code>	Create parameters for third rf channel (M)
	<code>fixpar4rf</code>	Create parameters for fourth rf channel (M)
	<code>parfix</code>	Update parameter set (M)
	<code>parversion</code>	Version of parameter set (P)
	<code>rt</code>	Retrieve FIDs (C)
	<code>rtp</code>	Retrieve parameters (C)
	<code>rtv</code>	Retrieve individual parameters (C)
	<code>updatepars</code>	Update all parameter sets saved in a directory (M)
	<code>userfixpar</code>	Macro called by <code>fixpar</code> (M)

fixparflow

Syntax: `fixparflow`

Description:

Arguments:

Examples: `fixparflow`

Related:

fixpar3rf **Create parameters for third rf channel (M)**

Applicability: Systems with a second decoupler.

Syntax: `fixpar3rf`

Description: Checks for the existence of all acquisition parameters related to the second decoupler. Any parameters found to be absent are created, characterized, and initialized by the macro. `fixpar3rf` is run as a part of the standard `fixpar` macro if the system configuration parameter `numrfch` is greater than 2 (i.e., the number of rf channels on the system is set at 3 or more).

See also: *Getting Started*

Related:	<code>fixpar</code>	Correct parameter characteristics in experiment (M)
	<code>fixpar4rf</code>	Create parameters for fourth rf channel (M)
	<code>numrfch</code>	Number of rf channels (P)

fixpar4rf Create parameters for fourth rf channel (M)

Applicability: Systems with a third decoupler.

Syntax: `fixpar4rf`

Description: Checks for the existence of all acquisition parameters related to the third decoupler. Any parameters found to be absent are created, characterized, and initialized. `fixpar4rf` is run as a part of the standard `fixpar` macro if the system configuration parameter `numrfch` is greater than 3 (i.e., the number of rf channels on the system is set at 4).

See also: *Getting Started*

Related: `fixpar` Correct parameter characteristics in experiment (M)
`fixpar3rf` Create parameters for third rf channel (M)
`numrfch` Number of rf channels (P)

fixpar5rf Create parameters for fifth rf channel (M)

Applicability: Systems with a deuterium decoupler channel as the fourth decoupler.

Syntax: `fixpar5rf`

Description: Checks for the existence of all acquisition parameters related to the fourth decoupler. Any parameters found to be absent are created, characterized, and initialized. `fixpar5rf` is run as a part of the standard `fixpar` macro if the system configuration parameter `numrfch` is greater than 4 (i.e., the number of rf channels on the system is set at 5).

See also: *Getting Started*

Related: `fixpar` Correct parameter characteristics in experiment (M)
`fixpar4rf` Create parameters for fourth rf channel (M)
`numrfch` Number of rf channels (P)

fixup Adjust parameter values selected by setup macros (M)

Syntax: `fixup`

Description: Called by the experiment setup macros `h1`, `c13`, `hc`, `hcapt`, `capt`, and `hcosy`. As provided, the text of `fixup` is all in quotes so that it does nothing. It is intended to provide each user with a mechanism to make adjustments to values selected by the setup macros.

See also: *User Guide: Liquids NMR*

flashc Convert compressed 2D data to standard 2D format (C)

Syntax: `flashc(<'nf'>, 'ms' | 'mi' | 'rare', ns, traces, echoes)`

Description: Converts 2D FID data files from compressed formats (`seqcon='nncsn'`, `seqcon='nccnn'`, `seqcon='nnccn'`) to standard format (`seqcon='ncsnn'`) or from standard format to compressed format. Compressed data is taken by using the `nf` parameter; that is, compressed data is acquired as one large uninterrupted “multiFID” acquisition.

`flashc` reads the file `fid` in the `acqfil` subdirectory of the current experiment.

`flashc` can convert a compressed-compressed multislice, multiecho, or multi-image sequence. It can also convert a “rare” type sequence with a compressed phase-encode echo train.

`flashc` changes the values of the following VNMR parameters:

Compressed-compressed or standard format to compressed format

- **ni** is set to 1 if no argument is provided.
- **nf** is set to the value of **nf** divided by the multislice, **ms**, or multi-image, **mi**, value.
- **arraydim** is set to the product of its original value and the value of the **traces** argument.
- **arrayelems** is set to 1 if no parameters were arrayed during data acquisition or to 2 if any parameter was arrayed during data acquisition.

Compressed format to standard format

- **nf** is set to the value of the **traces** argument, or to 1 if no argument is provided.
- **ni** is set to the value of **nf** divided by the multislice, **ms**, or multi-image, **mi**, value.
- **arraydim** is set to the product of its original value and the original value of **nf**.
- **arrayelems** is set to 1 if no parameters were arrayed during data acquisition or to 2 if any parameter was arrayed during data acquisition.

Arguments: **nf** is the number of FIDs in the second dimension of a 2D experiment. When converting data in the standard format to a compressed format, **nf** must always be the first argument.

When converting compressed-compressed or “rare” type sequences, the first argument must be a string defining the type of compression:

- 'mi' is a keyword for the multi-image type of compression.
- 'ms' is a keyword for the multislice type of compression.
- 'rare' is a keyword for the “rare” multiecho, rare type, fast-imaging data sets.

(*Standard to compressed*) **ns** is the number of images slices or array elements to be retained.

(*Compressed-compressed or rare to standard*) **traces** is the number of compressed traces to retain for each **ni**. The parameter **nf** is set to this number after **flashc** has run.

(*Compressed-compressed or rare to standard*) **echoes** is the number of compressed echoes, used with “rare” type formatting.

Examples: *Compressed-compressed or standard format to compressed format*

```
flashc('nf') (standard to compressed)
flashc('nf','ms',ns) (compressed phase-encode and multislice)
flashc('nf','mi',ns) (compressed multi-image and phase-encode)
```

Compressed-compressed format or rare format to standard format

```
flashc (simple compressed phase-encode)
flashc('ms',ns) (compressed phase-encode and multislice)
flashc('mi',ns) (compressed multi-image and phase-encode)
flashc('rare',ns,etl)
```

See also: *User Guide: Imaging*

Related:	arraydim	Dimension of experiment (P)
	ft2d	Fourier transform 2D data (C)
	ft3d	Fourier transform 3D data (C)
	nf	Number of FIDs (P)

ni Number of increments in 1st indirectly detected dimension (P)
seqcon Acquisition loop control (P)

flip Flip between graphics and text windows (C)

Syntax: `flip<('graphics'|'text' \`
`<,'off'|'on'|'autooff'|'autoon'>>`

Description: Windows on the display screen often overlap, with some windows on top or in front of other windows. `flip` brings the graphics window or text window to the top of the screen. It can also control under what circumstances the graphics or text windows will come to the front due to a parameter change or by the actions of certain interactive programs (`dg`, `dg1`, `ds`, etc.) that send commands to write or draw in a window.

Arguments: `'graphics'` is a keyword to bring the graphics window to the front.
`'text'` is a keyword to bring the text window to the front.
`'off'` is a keyword that the window specified by the first argument will not come to the front due to parameter changes or from commands that write to the window. After `'off'` is set, entering `flip`, `flip('text')`, or `flip('graphics')` will bring the window to the front.
`'on'` is a keyword to reset the action of the `'off'` keyword.
`'autooff'` is a keyword that the window specified by the first argument will not come up to the front due to parameter changes, but specific commands that write or draw to the window (`dg`, `dg1`, `ds`, etc. and `flip`, `flip('text')`, `flip('graphics')`) will still bring the window to the front.
`'autoon'` is a keyword to reset the action of the `'autooff'` keyword.

Alternate: Flip button in the Permanent menu.

Examples: `flip`
`flip('graphics')`
`flip('text','autooff')`

See also: *Getting Started, VNMR User Programming*

Related: **large** Use large graphics window (C)
small Use small graphics window (C)

flipflop Set up parameters for FLIPFLOP pulse sequence (M)

Applicability: Systems with solids module. Sequence is not supplied on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Syntax: `flipflop`

Description: Sets up a multipulse parameter set for tuning out “phase glitch” in the probe and pulse amplifier.

See also: *User Guide: Solid-State NMR*

fliplist Standard flip angle list (P)

Applicability: Systems with imaging capabilities.

Description: Contains an array of real values defining values of the standard flip angles used for the pulses in the `plist` array (e.g., `fliplist=180,90,180`). The `nd`, `seqcon`, `plist`, `patlist`, `pwrlist`, `fliplist`, and `sslist` parameters configure a particular parameter set for an application sequence defined by the value of the `seqfil` parameter. The `plist`, `patlist`,

`pwrlist`, `fliplist`, and `sslist` parameters provide information concerning the rf pulse and conjugate gradients used by the sequence.

See also: *User Guide: Imaging*

Related:	<code>nD</code>	Application dimension (P)
	<code>patlist</code>	Active pulse template parameter list (P)
	<code>plist</code>	Active pulse length parameter list (P)
	<code>pwrlist</code>	Active pulse power level parameter list (P)
	<code>seqcon</code>	Acquisition loop control (P)
	<code>seqfil</code>	Application object code name (P)
	<code>sslist</code>	Conjugate gradient list (P)

FLUORINE **Set up parameters for fluorine spectrum (M)**

Applicability: *GLIDE* only

Syntax: `FLUORINE`

Description: Internal macro that sets up a fluorine spectrum in *GLIDE*. This macro is not used if fluorine is the first experiment in the chain.

flush **Write out data in VNMR memory (C)**

Syntax: `flush`

Description: Writes out the current data and parameters in memory buffers. Normally, this information is not written to disk until exiting VNMR or joining another experiment. One reason to use `flush` is to be able to access experimental data from a program separate from the VNMR program.

See also: *VNMR User Programming*

fn **Fourier number in directly detected dimension (P)**

Description: Selects the Fourier number for the Fourier transformation along the directly detected dimension. This dimension is often referred to as the f_2 dimension in 2D data sets, the f_3 dimension in 3D data sets, etc.

Values: 'n' or a number equal to a power of 2 (minimum is 32). If `fn` is not *entered* exactly as a power of 2, it is automatically rounded to the nearest higher power of 2 (e.g., setting `fn=32000` gives `fn=32768`). `fn` can be less than, equal to, or greater than `np`, the number of directly detected data points:

- If `fn` is less than `np`, only `fn` points are transformed.
- If `fn` is greater than `np`, `fn` minus `np` zeros are added to the data table (“zero-filling”).
- If `fn`= 'n', `fn` is automatically set to the power of 2 greater than or equal to `np`.

See also: *Getting Started*

Related:	<code>fn1</code>	Fourier number in 1st indirectly detected dimension (P)
	<code>fn2</code>	Fourier number in 2nd indirectly detected dimension (P)
	<code>np</code>	Number of data points (P)

fn1 **Fourier number in 1st indirectly detected dimension (P)**

Description: Selects the Fourier number for the Fourier transformation along the first indirectly detected dimension. This dimension is often referred to as the f_1

dimension of a multi-dimensional data set. The number of increments along this dimension is controlled by the parameter **ni**.

Values: **fn1** is set in a manner analogous to the parameter **fn**, with **np** being substituted by $2 * ni$.

See also: *User Guide: Liquids NMR*

Related:	fn	Fourier number in directly detected dimension (P)
	fn2	Fourier number in 2nd indirectly detected dimension (P)
	ni	Number of increments in 1st indirectly detected dimension (P)
	np	Number of data points (P)

fn2 **Fourier number in 2nd indirectly detected dimension (P)**

Description: Selects the Fourier number for the Fourier transformation along the second indirectly detected dimension. This dimension is often referred to as the f_2 dimension of a multidimensional data set. The number of increments along this dimension is controlled by the parameter **ni2**. **fn2** is set in a manner analogous to the parameter **fn**, with **np** being substituted by $2 * ni2$.

See also: *User Guide: Liquids NMR*

Related:	fn	Fourier number in directly detected dimension (P)
	fn1	Fourier number in 1st indirectly detected dimension (P)
	ni2	Number of increments in 2nd indirectly detected dimension (P)
	np	Number of data points (P)

fn2D **Fourier number to build up 2D DOSY display in frequency domain (P)**

Description: In 2D DOSY sequences (Dbppste, DgcsteSL, Doneshot, Dbppsteinept), replaces **fn** when setting up the 2D display.

See also: *User Guide: Liquids NMR*

Related:	ddif	Synthesize and display DOSY plot (C)
	dosy	Process DOSY experiments (M)

focus **Send keyboard focus to VNMR input window (C)**

Syntax: `focus`

Description: Sends keyboard focus to the VNMR input window. This is only useful for macro programming.

See also: *VNMR User Programming*

foldcc **Fold INADEQUATE data about two-quantum axis (C)**

Syntax: `foldcc`

Description: Symmetrizes 2D INADEQUATE data along the P-type double-quantum axis and applies an automatic **dc** baseline correction. `foldcc` functions for both hypercomplex and complex 2D data.

See also: *User Guide: Liquids NMR*

Related:	dc	Calculate spectral drift correction (C)
	foldj	Fold J-resolved 2D spectrum about $f_1=0$ axis (C)
	foldt	Fold COSY-like spectrum along diagonal axis (C)
	rotate	Rotate 2D data (C)

F

foldj **Fold J-resolved 2D spectrum about $f_1=0$ axis (C)**

Syntax: `foldj`

Description: Symmetrizes heteronuclear 2D-J or rotated homonuclear 2D-J experiments about the $f_1=0$ axis. `foldj` functions with both complex and hypercomplex 2D data.

See also: *User Guide: Liquids NMR*

Related: `foldcc` Fold INADEQUATE data about 2-quantum axis (C)
`foldt` Fold COSY-like spectrum along diagonal axis (C)
`rotate` Rotate 2D data (C)

foldt **Fold COSY-like spectrum along diagonal axis (C)**

Syntax: `foldt<('symm' | 'triang')>`

Description: Folds COSY-like correlation spectra about the diagonal. The 2D spectrum must exhibit a *P-type diagonal* for `foldt` to work properly (a P-type diagonal goes from the bottom left-hand side to the top right-hand side of the contour display.) `foldt` functions for both hypercomplex and complex 2D data but requires that `fn=fn1` and `sw=sw1`.

Arguments: 'symm' is a keyword for the folding process to perform a symmetrization of the data by replacing every two symmetry-related points with the one point therein that has the least magnitude. This value is the default.

'triang' is a keyword for the folding process to perform a triangularization of the data by replacing every two symmetry-related points with their geometric mean.

See also: *User Guide: Liquids NMR*

Related: `fn` Fourier number in directly detected dimension (P)
`fn1` Fourier number in 1st indirectly detected dimension (P)
`foldcc` Fold INADEQUATE data about 2-quantum axis (C)
`foldj` Fold J-resolved 2D spectrum about $f_1=0$ axis (C)
`rotate` Rotate 2D data (C)
`sw` Spectral width in directly detected dimension (P)
`sw1` Spectral width in 1st indirectly detected dimension (P)

fontselect **Open FontSelect window (C)**

Syntax: `fontselect`

Description: Opens the FontSelect window for defining fonts in window panes created by `setgrid`. A different font can be selected for every window pane combination of rows and columns. Separate fonts can also be selected for a large or small overall graphic window.

Alternate: FontSelect button in the Windows menu.

See also: *Getting Started*

Related: `curwin` Current window (P)
`jwin` Activate current window (M)
`mapwin` List of experiment numbers (P)
`setgrid` Activate selected window (M)
`setwin` Activate selected window (C)

format **Format a real number or convert a string for output (C)**

Syntax: (1) `format(real_number, length, precision):return`
 (2) `format(string, 'upper' | 'lower' | 'isreal'):return`

Description: Using syntax 1, `format` takes a real number or real type variable and formats it into a string with given length and precision and rounds it off if necessary (see examples 1 to 4 below). `format` can also be used to format a real type variable as a real number (see example 5).

Using syntax 2, `format` converts a string variable into a new string of characters either all upper case or all lowercase (see examples 6 and 7) or tests the string to determine if it represents a real number (see example 8).

Arguments: `real_number` is the real type variable containing the value to be formatted.
`length` is the length of for formatted real number. If `length` is set to 0, just enough places are used to hold the number.

`precision` is the precision (i.e., the number of places to the right of the decimal point) of the formatted real number. If `precision` is set to 0, output is an integer.

`string` is the string variable to be converted into upper or lower case.

'`upper`' is a keyword to convert the string variable given by `string` into all upper case characters.

'`lower`' is a keyword to convert `string` into all lower case characters.

'`isreal`' is a keyword that tests the first argument to verify that the argument satisfies the rules for a real number. When given, `format` returns a 1 in the first argument and can represent a real number and a zero otherwise.

`return` is the return string variable, real number, or integer.

Examples: 1. `format(a, 5, 2):n1` If `a=24.1264` then `n1='24.13'`
 2. `format(a, 9, 4):n2` If `a=24.1264` then `n2='24.1264'`
 3. `format(a, 0, 3):n3` If `a=24.1264` then `n3='24.126'`
 4. `format(a, 2, 0):n1` If `a=24.1264` then `n1='24'`
 5. `format(a, 2, 0):r1` If `a=24.1264` then `r1=24`
 6. `format(solvent, 'upper'):n2` If `solvent='CDCl3'`
 then `n2='cdcl3'`
 7. `format(solvent, 'lower'):n3` If `solvent='CDCl3'`
 then `n3='CDCDL3'`
 8. `format($1, 'isreal'):$a` If `$1=1` then `$a=1`

See also: *VNMR User Programming*

Related: `n1, n2, n3` Name storage for macros (P)
`r1-r7` Real-value storage for macros (P)

fp **Find peak heights or phases (C)**

Syntax: `fp(<'phase', ><index1, index2, ...>)>`

Description: Following a line listing (either `d11` or `n11`), `fp` measures the peak height of each peak in an array of spectra. The results of the analysis are written to a text file `fp.out` in the current experiment directory. If the `npoint` parameter is defined in the current parameter set and this parameter is "on," it determines the range of data points over which a maximum is searched when determining peak heights. The possible values of `npoint` are 1 to `fn/4`. The default is 2.

Arguments: '`phase`' is a keyword to measure the phase of each peak instead of height.
`index1, index2, ...` restricts measuring peak heights or phases to the lines listed.

Examples: `fp`
`fp(1,3)`
`fp('phase')`

See also: *User Guide: Liquids NMR*

Related: `dll` Display listed line frequencies and intensities (C)
`fn` Fourier number in directly detected dimension (P)
`getll` Get line frequency and intensity from line list (C)
`nl` Position cursor at the nearest line (C)
`nll` Find line frequencies and intensities (C)
`npoint` Number of points for `fp` peak search (P)

fpmult First point multiplier for np FID data (P)

Description: Allows error correction if the first point of an FID is misadjusted. In a 1D experiment, this adjustment influences the overall integral of the spectrum. For n -dimensional experiments, if the correction is not made, “ridges” can appear. In 2D experiments, the ridges appear as “ f_2 ridges.” In 3D experiments, the ridges appear as “ f_3 ridges.” These ridges can clearly be seen in the noise region on the top and bottom of a 2D spectrum (when `trace='f1'`) as a low-intensity profile of the diagonal. The sign and intensity of the ridges is controlled by the magnitude of `fpmult`.

It has been recognized that the first point of a FID that is sampled at exactly time equal to zero must be multiplied by 0.5 for the Fourier transform to function properly. The `fpmult` parameter gives you a method to fine-tune the actual correction factor.

Values: Default is 1.0, except that if the processing involves backward extension of the time-domain data with linear prediction, the default changes to 0.5. If `fpmult` is set to 'n', `fpmult` takes on its default value.

See also: *User Guide: Liquids NMR*

Related: `fpmult1` First point multiplier for `ni` interferogram data (P)
`fpmult2` First point multiplier for `ni2` interferogram data (P)
`np` Number of data points (P)
`trace` Mode for n -dimensional data display (P)
`wft2da` Weight and Fourier transform phase-sensitive data (M)

fpmult1 First point multiplier for ni interferogram data (P)

Description: Operates on `ni` hypercomplex or complex interferogram data in a manner analogous to `fpmult`. In many 2D experiments, the t_1 values are adjusted so there is no first-order phasing in the f_1 and f_2 dimensions. In this case, `fpmult1` should be 0.5. If the t_1 value is adjusted so that there is a 180° first-order phase correction, `fpmult1` should be 1.0.

Values: Default value is 0.5. If `fpmult1` is set to 'n', it takes on its default value.

See also: *User Guide: Liquids NMR*

Related: `fpmult` First point multiplier for `np` FID data (P)
`fpmult2` First point multiplier for `ni2` interferogram data (P)
`ni` Number of increments in 1st indirectly detected dimension (P)

fpmult2 First point multiplier for ni2 interferogram data (P)

Description: Operates on `ni2` hypercomplex or complex interferogram data in a manner analogous to `fpmult`. In many 3D experiments, the t_2 value is adjusted so that

there is no first-order phasing in the f_1 and f_2 dimensions. In this case, `fpmult2` should be 0.5. If the t_2 value is adjusted so that there is a 180° first-order phase correction, `fpmult2` should be 1.0.

Values: Default value is 0.5. If `fpmult2` is set to 'n', it takes on its default value.

See also: *User Guide: Liquids NMR*

Related: `fpmult` First point multiplier for np FID data (P)
`fpmult1` First point multiplier for ni interferogram data (P)
`ni2` Number of increments in 2nd indirectly detected dimension (P)

fr Full recall of a display parameter set (M)

Syntax: (1) `frset_number`
 (2) `fr(set_number)`

Description: Performs a full recall of a display parameter set, setting all parameters to exactly as they were when the corresponding `s` command was entered.

Arguments: `set_number` is the number of the display parameter set.

Examples: `fr2`
`fr(3)`

See also: *Getting Started*

Related: `r` Recall display parameter set (M)
`s` Save display parameters as a set (M)

fread Read parameters from file and load them into a tree (C)

Syntax: `fread(file<, tree<, 'reset | value' >>)`

Description: Reads VNMR parameters from a file and loads the parameters into a tree. The tree can be global, current, processed, or systemglobal. `fread` can read from any file that has parameters stored in the correct VNMR format.

Note that if parameters are read into the global tree, certain important system parameters are not loaded because these parameters should not be changed. The parameters that are not loaded are `userdir`, `systemdir`, `curexp`, `autodir`, `auto`, `vnmraddr`, and `acqaddr`.

Arguments: `file` is the name of the file containing parameters stored in VNMR format.
`tree` is one of the keywords 'global', 'current', 'processed', or 'systemglobal'. The default is 'current'. This argument specifies the type of tree into which the parameters are loaded. Refer to the `create` command for more information on types of trees.

'reset' is a keyword that causes the parameter tree to be cleared before the new parameter file is read. Without this option, parameters read from a file are added to the existing preloaded parameters. To use this option, `tree` must also be specified.

'value' is a keyword that causes only the values of the parameters in the file to be loaded. If a preloaded variable does not already exist, a new one is not created. Parameter attributes are not changed, and enumerated values are not changed. To use this option, `tree` must also be specified.

Examples: `fread('/vnmr/stdpar/H1.par/procpar')`
`fread('sampvar', 'global')`
`fread('setvar', 'current', 'reset')`
`fread('var1', 'processed', 'value')`

See also: *VNMR User Programming*

Related:	<code>auto</code>	Automation mode active (P)
	<code>autodir</code>	Automation directory absolute path (P)
	<code>create</code>	Create new parameter in a parameter tree (C)
	<code>curexp</code>	Current experiment directory (P)
	<code>destroy</code>	Destroy a parameter (C)
	<code>display</code>	Display parameters and their attributes (C)
	<code>fsave</code>	Save parameters from a tree to a file (C)
	<code>rtp</code>	Retrieve parameters (C)
	<code>systemdir</code>	VNMR system directory (P)
	<code>userdir</code>	VNMR user directory (P)

fsave Save parameters from a tree to a file (C)

Syntax: `fsave(file<, tree>)`

Description: Writes parameters from a parameter tree to a file.

Arguments: `file` is the name of the file, which can be any valid file for which the user has write permission. If the file already exists, it will be overwritten.

`tree` is one of the keywords 'global', 'current', 'processed', or 'systemglobal'. The default is 'current'. Refer to the `create` command for more information on types of trees.

Examples: `fsave('var1')`
`fsave('sampvar', 'global')`

See also: *VNMR User Programming*

Related:	<code>create</code>	Create new parameter in a parameter tree (C)
	<code>destroy</code>	Destroy a parameter (C)
	<code>display</code>	Display parameters and their attributes (C)
	<code>fread</code>	Read parameters from file and load them into a tree (C)
	<code>svp</code>	Save parameters from current experiment (C)

fsq Frequency-shifted quadrature detection (P)

Description: On ^{UNITY}*INOVA*, *MERCURY-VX*, and *MERCURY* systems, selects whether to use frequency-shifted quadrature detection. When `fsq` is turned on, if `dsp` is on, the observe frequency is offset by `oslsfrq`, and the digital filter is also offset by `oslsfrq`. The default value of `oslsfrq` is $1.25 * sw$.

On systems other than ^{UNITY}*INOVA*, *MERCURY-VX*, and *MERCURY*, frequency-shifted quadrature detection can be done using inline DSP. The effect of `fsq` is to offset only the digital filter by `oslsfrq`. The observe frequency must be offset by `oslsfrq` by modifying the pulse sequence as described in the manual *Getting Started*.

Values: 'n' turns frequency-shifted quadrature detection off. 'y' turns it on.

See also: *Getting Started*

Related:	<code>dsp</code>	Type of DSP for data acquisition (P)
	<code>oslsfrq</code>	Bandpass filter offset for oversampling (P)
	<code>oversamp</code>	Oversampling factor for acquisition (P)
	<code>sw</code>	Spectral width in directly detected dimension (P)

ft Fourier transform 1D data (C)

Syntax: (1) `ft(<options>, <'nf'>, <start>, <finish>, <step>)`
 (2) `ft('inverse', exp_number, expansion_factor)`

Description: In syntax 1, performs a Fourier transform on one or more 1D FIDs without weighting applied to the FID. `ft` executes a left-shift, zero-order phase rotation, and a frequency shift (first-order phase rotation) according to the parameters `lsfid`, `phfid`, and `lsfrq`, respectively, on the time-domain data, prior to Fourier transformation. The type of Fourier transform to be performed is determined by the parameter `proc`. Solvent suppression is turned on or off with the parameters `ssfilter` and `ssorder`. For arrayed data sets, `ft` Fourier transforms all of the array elements. To Fourier transform selected array elements, `ft` can be passed numeric arguments.

In syntax 2, `ft` performs an inverse Fourier transform of the entire spectrum. (VNMR does not currently support inverse Fourier transformation of arrayed 1D or 2D data sets.)

Arguments: `options` can be any of the following (all string arguments must precede the numeric arguments):

- `'acq'` is a keyword to check if any elements of a multi-FID experiment have already been transformed. If so, these previously transformed elements will not be retransformed.
- `'nodc'` is a keyword to not perform the usual FID drift correction.
- `'nods'` is a keyword to prevent an automatic spectral display (`ds`) from occurring. This outcome is useful for various plotting macros.
- `'noft'` is a keyword to skip the Fourier transform, thereby allowing use of all spectral manipulation and plotting commands on FIDs.
- `'zero'` is a keyword to zero the imaginary channel of the FID prior to the Fourier transform. This zeroing occurs after any FID phasing. Its use is generally limited to wideline solids applications.

`'nf'` is a keyword that makes a single FID element containing `nf` traces to be transformed as if it were `nf` separate FID elements. If `'nf'` precedes the list of numeric arguments, the rules for interpreting the numeric arguments change slightly. Passing no numeric arguments results in the transformation of all `nf` traces in the first FID element. Passing a single numeric argument results in the transformation of all `nf` traces in the requested FID element (e.g., `ft('nf', 3)` transforms all `nf` traces for element 3). Regardless of the requested FID element, the resulting spectra are labeled as 1 to `nf` because multiple elements cannot be transformed using `ft('nf')`. Subsequent numeric arguments are interpreted as previously described.

`start` is the index of a particular element to be transformed. For an array, `start` is the index of the first element to be transformed.

`finish` is the index of the last element to be transformed for an array.

`step` specifies the increment between successive elements that are to be transformed for an array. The default is 1.

`'inverse'` is a keyword specifying an inverse Fourier transform.

`exp_number` is the number of the experiment, from 1 to 9, for storing the resulting FID from the inverse Fourier transform.

`expansion_factor` defines the expansion of the spectrum before the inverse Fourier transform is performed. This argument is equivalent to a multiplier for the `fn` parameter. The multiplier is restricted to between 1 and 32 and is rounded up internally to the nearest power of 2.

Examples: `ft`
`ft(1)`
`ft(3,7)`

```
ft(2,10,2)
ft('nf',3)
```

Alternate: Transform button in the 1D Data Processing Menu.

See also: *Getting Started*

Related:

dcrmv	Remove dc offsets from FIDs in special cases (P)
fn	Fourier number in directly detected dimension (P)
lsfid	Number of points to left-shift the np FID (P)
lsfrq	Frequency shift of the fn spectrum in Hz (P)
nf	Number of FIDs (P)
phfid	Zero-order phasing constant for np FID (P)
proc	Type of processing on the np FID (P)
ssfilter	Full bandwidth of digital filter to yield a filtered FID (P)
ssorder	Order of polynomial to fit digitally filtered FID (P)
wft	Weight and Fourier transform 1D data (C)

ft1d **Fourier transform along f_2 dimension (C)**

Syntax: (1) `ft1d(element_number)`
 (2) `ft1d(<'nf', element_number)`
 (3) `ft1d(<options>, <coefficients>)`

Description: Performs the first Fourier transformation along the f_2 dimension, without weighting, and matrix transposition. `ft1d` allows the display of t_1 interferograms with the `dcon` and `dconi` commands. For arrayed 2D FID data, a single array element can be weighted and transformed using syntax 1 or 2. The keyword 'nf' is used in syntax 2 to specify that the 2D data is collected in the compressed form using 'nf'. Complex and hypercomplex interferograms can be constructed explicitly by supplying a series of options and coefficients using syntax 3.

For information on real as opposed to complex Fourier transforms, see the descriptions of the `proc`, `procl`, and `proc2` parameters. For information on left-shifting, zero-order phase rotation, and frequency shifting of the FID and interferogram time-domain data during the 2D Fourier transformation, see the descriptions of the parameters `lsfid`, `lsfid1`, `lsfid2`, `phfid`, `phfid1`, `phfid2`, `lsfrq`, `lsfrq1`, and `lsfrq2`, as appropriate. For information on the lfs (low-frequency suppression) and zfs (zero-frequency suppression) solvent suppression options, see the description of the parameters `ssfilter` and `ssorder`, and the macro `parfidss`.

Arguments: `element_number` is a single array element to be weighted and transformed. `options` can be the keywords 'ptype' or 'ntype' but neither serve a useful function because the differential effect of these arguments is applied only during the course of the second Fourier transformation. The default is 'ntype'.

`coefficients` are a series of coefficients according to the following scheme: RR1 is the coefficient used to multiply the real part (first R) of spectra set 1 before it is added to the real part (second R) of the interferogram. IR2 would thus represent the contribution from the imaginary part of spectra set 2 to the real part of the interferogram, and so on. The scheme is depicted below.

```
ft1d(RR1, IR1, RR2, IR2, . . . , RI1, II1, RI2, II2, . . . )
```

where:

```
RR1*REAL(w2, element=1) -> REAL(t1)
IR1*IMAG(w2, element=1) -> + REAL(t1)
RR2*REAL(w2, element=2) -> + REAL(t1)
```

```

IR2*IMAG(w2,element=2) -> + REAL(t1)
. . .
RI1*REAL(w2,element=1) ->  IMAG(t1)
II1*IMAG(w2,element=1) -> + IMAG(t1)
RI2*REAL(w2,element=2) -> + IMAG(t1)
II2*IMAG(w2,element=2) -> + IMAG(t1)
. . .

```

See also: *User Guide: Liquids NMR*

Related:	dconi	Interactive 2D data display (C)
	ft2d	Fourier transform 2D data (C)
	lsfid	Number of complex points to left-shift np FID (P)
	lsfid1	Number of complex points to left-shift ni interferogram (P)
	lsfid2	Number of complex points to left-shift ni2 interferogram (P)
	lsfrq	Frequency shift of the fn spectrum (P)
	lsfrq1	Frequency shift of the fn1 spectrum (P)
	lsfrq2	Frequency shift of the fn2 spectrum (P)
	parfidss	Create parameters for time-domain solvent subtraction (M)
	phfid	Zero-order phasing constant for np FID (P)
	phfid1	Zero-order phasing constant for ni interferogram (P)
	phfid2	Zero-order phasing constant for ni interferogram (P)
	proc	Type of processing on np FID (P)
	proc1	Type of processing on ni interferogram (P)
	proc2	Type of processing on ni2 interferogram (P)
	pmode	Processing mode for 2D data (P)
	ssorder	Order of polynomial to fit digitally filtered FID (P)
	ssfilter	Full bandwidth of digital filter to yield a filtered FID (P)
	wft2d	Weight and Fourier transform 2D data (C)

ft1da **Fourier transform phase-sensitive data (M)**

Syntax: `ft1da<(options)>`

Description: Performs the first (f_2) transform of a 2D transform or the first part of a 3D transform. Otherwise, `ft1da` has the same functionality as the `ft2da` command. See the description of `ft2da` for further information.

Arguments: `options` are the same as used with `ft2da`. See `ft2da` for details.

See also: *User Guide: Liquids NMR*

Related:	ft2d	Fourier transform 2D data (C)
	ft2da	Fourier transform phase-sensitive data (M)
	wft1da	Weight and Fourier transform phase-sensitive data (M)
	wft2da	Weight and Fourier transform phase-sensitive data (M)

ft1dac **Combine arrayed 2D FID matrices (M)**

Syntax: `ft1dac<(<mult1><,mult2>,...<,multn>)>`

Description: Allows ready combination of 2D FID matrices within the framework of the 2D Fourier transformation program. No weighting is performed. `ft1dac` requires that the data be acquired either without f_1 quadrature or with f_1 quadrature using the TPPI method. This macro is used for TOCSY (with multiple mixing times).

Arguments: `mult1`, `mult2`, ..., `multn` are multiplicative coefficients. The n th argument is a real number and specifies the multiplicative coefficient for the n th 2D FID matrix.

See also: *User Guide: Liquids NMR*

Related: **ft2dac** Combine arrayed 2D FID matrices (M)
tocsy Set up parameters for TOCSY pulse sequence (M)
wft1da Weight and Fourier transform phase-sensitive data (M)
wft1dac Combine arrayed 2D FID matrices (M)

ft2d **Fourier transform 2D data (C)**

Syntax: (1) `ft2d(array_element)`
 (2) `ft2d('nf' <array_element>)`
 (3) `ft2d(<options>, <plane_number>, <coefficients>)`
 (4) `ft2d('ni' | 'ni2', element_number, increment)`
 (5) `ft2d('ni' | 'ni2', increment, <coefficients>)`

Description: Performs the complete 2D Fourier transformation, without weighting, in both dimensions. If the first Fourier transformation has already been done using **ft1d**, **wft1d**, **ft1da**, or **wft1da**, the **ft2d** command performs only the second (t_1) transform.

For arrayed 2D FID data, a single array element can be weighted and transformed using syntax 1. If the data is collected in “compressed” form using 'nf', syntax 2 must be used. Complex and hypercomplex interferograms can be constructed explicitly by supplying a series of coefficients using syntax 3. If an arrayed 3D data set is to be selectively processed, the format of the arguments to **ft2d** changes to syntax 4. For example, `ft2d('ni', 1, 2)` performs a 2D transform along **np** and **ni** of the second **ni2** increment and the first element within the explicit array. This command yields a 2D **np-ni** frequency plane.

Arrayed 3D data sets can also be subjected to 2D processing to yield 2D absorptive spectra. If the States-Haberhorn method is used along both f_1 (**ni** dimension) and f_2 (**ni2** dimension), there are generally 4 spectra per (**ni,ni2**) 3D element. In this case, using syntax 5, entering `ft2d('ni2', 2, <16 coefficients>)` performs a 2D transform along **np** and **ni2** of the second **ni** increment using the ensuing 16 coefficients to construct the 2D t_1 -interferogram from appropriate combinations of the 4 spectra per (**ni,ni2**) 3D element.

If there are n data sets to be transformed, as in typical phase-sensitive experiments, $4 * n$ coefficients must be supplied. The first $2 * n$ coefficients are the contributions to the real part of the interferogram, alternating between absorptive and dispersive parts of the successive data sets. The next $2 * n$ coefficients are the contributions to the imaginary part of the interferogram, in the same order. Thus, using the definition that the first letter refers to the source data set, the second letter refers to the interferogram, and the number identifies the source data set, we have the following cases:

<i>Data sets</i>	<i>Coefficient order</i>
1	RR1, IR1, RI1, II1
2	RR1, IR1, RR2, IR2, RI1, II1, RI2, II2
3	RR1, IR1, RR2, IR2, RR3, IR3, RI1, II1, RI2, II2, RI3, II3
.

The coefficients are often 1, 0, or -1, but this is not always the case. Any non-integral coefficient can be used, and as many coefficients can be nonzero as is

desired. Up to 32 coefficients can be supplied, which at 4 per data set allows the addition, subtraction, etc., of eight 2D data sets (e.g., 8 different phase cycles).

For information on real as opposed to complex Fourier transforms, see the descriptions of the `proc`, `proc1`, and `proc2` parameters. For information on left-shifting, zero-order phase rotation, and frequency shifting of the FID and interferogram time-domain data during the 2D Fourier transformation, see the descriptions of the parameters `lsfid`, `lsfid1`, `lsfid2`, `phfid`, `phfid1`, `phfid2`, `lsfrq`, `lsfrq1`, and `lsfrq2`, as appropriate. For information on the `lfs` (low-frequency suppression) and `zfs` (zero-frequency suppression) solvent suppression options, see the description of parameters `ssfilter` and `ssorder`, and macro `parfidss`.

Arguments: `array_element` is a single array element to be transformed.

`options` can be any of the following (all string arguments must precede the numeric arguments):

- `'ptype'` is a keyword to transform P-type data to yield a P-type contour display.
- `'ntype'` is a keyword to transform N-type data to yield a P-type contour display. This is the default.
- `'t2dc'` is a keyword to apply a dc correction to each t_2 FID prior to the first Fourier transform. The last 1/16-th of the time domain data is used to calculate the dc level.
- `'t1dc'` is a keyword to apply a dc correction to each t_1 interferogram prior to the second Fourier transform. The last 1/16-th of the time domain data is used to calculate the dc level.
- `'f2sel'` is a keyword to allow only preselected f_2 regions to be transformed along t_1 . The t_1 interferograms in the non-selected f_2 regions are zeroed but *not* transformed. The same mechanism used to select baseline regions for baseline correction (`bc`) is used to select the f_2 regions to be transformed along t_1 . Set `intmod='partial'` and partition the integral of the spectrum into several regions. The even numbered f_2 regions (e.g., 2, 4, 6) are transformed along t_1 ; the odd numbered regions are not transformed along t_1 .
- `'nf'` is a keyword to transform arrayed or multi-slice 2D data that has been collected in the compressed form as single 2D FIDs with multiple (`nf`) traces.
- `'ni2'` is a keyword to transform non-arrayed 2D data that have been collected with `ni2` and `sw2` (instead of `ni` and `sw1`). `addpar('3d')` creates the necessary processing parameters for the `'ni2'` operation.
- `'noop'` is a keyword to not perform any operation on the FID data. This option is used mainly to allow macros, such as `wft2da`, to have the same flexibility as commands.

`coefficients` are a series of coefficients according to the following scheme: `RR1` is the coefficient used to multiply the real part (first R) of spectra set 1 before it is added to the real part (second R) of the interferogram. `IR2` would thus represent the contribution from the imaginary part of spectra set 2 to the real part of the interferogram, and so forth. The scheme is depicted below.

```
ft2d(RR1, IR1, RR2, IR2, . . . , RI1, II1, RI2, II2, . . . )
```

where:

```
RR1*REAL(w2,element=1) -> REAL(t1)
IR1*IMAG(w2,element=1) -> + REAL(t1)
```

```

RR2*REAL(w2,element=2) -> + REAL(t1)
IR2*IMAG(w2,element=2) -> + REAL(t1)
. . .
RI1*REAL(w2,element=1) ->   IMAG(t1)
II1*IMAG(w2,element=1) -> + IMAG(t1)
RI2*REAL(w2,element=2) -> + IMAG(t1)
II2*IMAG(w2,element=2) -> + IMAG(t1)

```

'ni' is a keyword to selectively transform a particular **np-ni** 2D plane within a non-arrayed 3D data set. To identify the plane, 'ni' is followed by the `plane_number` argument, an integer from 1 through **ni2**.

'ni2' is a keyword to selectively transform a particular **np-ni2** 2D plane within a non-arrayed 3D data set. To identify the plane, 'ni2' is followed by the `plane_number` argument, an integer from 1 through **ni**.

`element_number` is the number of an element within the explicit array when selectively processing an arrayed 3D data set; it ranges from 1 to **ni2**

`increment` is the increment within the explicit array when selectively processing an arrayed 3D data set; it ranges 1 to `arraydim/(ni*ni2)`.

Examples: `ft2d(1,0,0,0,0,0,1,0)`
`ft2d(1)`
`ft2d('nf',3)`
`ft2d('ptype',...)`

See also: *User Guide: Liquids NMR*

Related:	dconi	Interactive 2D data display (C)
	dcrmv	Remove dc offsets from FIDs in special cases (P)
	fpmult	First point multiplier for np FID data (P)
	fpmult1	First point multiplier for ni interferogram data (P)
	ft1d	Fourier transform along f ₂ dimension (C)
	lsfid	Number of complex points to left-shift np FID (P)
	lsfid1	Number of complex points to left-shift ni interferogram (P)
	lsfid2	Number of complex points to left-shift ni2 interferogram (P)
	lsfrq	Frequency shift of the fn spectrum (P)
	lsfrq1	Frequency shift of the fn1 spectrum (P)
	lsfrq2	Frequency shift of the fn2 spectrum (P)
	parfidss	Create parameters for time-domain solvent subtraction (M)
	phfid	Zero-order phasing constant for np FID (P)
	phfid1	Zero-order phasing constant for ni interferogram (P)
	phfid2	Zero-order phasing constant for ni2 interferogram (P)
	proc	Type of processing on np FID (P)
	proc1	Type of processing on ni interferogram (P)
	proc2	Type of processing on ni2 interferogram (P)
	pmode	Processing mode for 2D data (P)
	ssorder	Order of polynomial to fit digitally filtered FID (P)
	ssfilter	Full bandwidth of digital filter to yield a filtered FID (P)
	wft1d	Weight and Fourier transform f ₂ for 2D data (C)
	wft2d	Weight and Fourier transform 2D data (C)

ft2da **Fourier transform phase-sensitive data (M)**

Syntax: `ft2da<(options)>`

Description: Processes 2D FID data and 2D planes at particular t₁ or t₂ times from a 3D data set for a pure absorptive display. `ft2da` differs from `wft2da` only in that, in the case of `wft1da`, weighting of the time-domain data is performed prior to

the FT. `ft2da` functions analogously to `ft1da` and `wft1da`, except that `ft2da` and `wft2da` perform only the f_2 Fourier transform.

Macros `ft1da`, `wft1da`, `ft2da`, and `wft2da` function for hypercomplex 2D FID data (`phase=1, 2`) and for TPPI 2D FID data (`phase=3` or `phase=1, 4`) acquired either with `ni` or `ni2`. If the data were acquired with `ni`, no additional arguments need be used with the macros. If the data were acquired with `ni2`, the keyword '`ni2`' must be used.

For `phase=1, 2`:

```
wft2da=wft2d('ptype',1,0,0,0,0,0,1,0)
```

For `phase=3`: `wft2da=wft2d(1,0,0,0)`

For `phase=1, 4`:

```
wft2da=wft2d('ptype',1,0,0,0,0,0,1,0)
```

Macros `ft1da`, `wft1da`, `ft2da`, and `wft2da` support selective 2D processing within a 3D FID data set. All permutations of hypercomplex and TPPI modes of data acquisition in t_1 and t_2 can be handled. For selective f_2f_3 processing, the numeric argument immediately following the '`ni2`' keyword is interpreted to be the t_1 increment number, which specifies the particular f_2f_3 plane (`plane_number`, see below) to be processed. For selective f_1f_3 processing, the t_2 increment number either follows the keyword '`ni`', which is optional, or is associated with the first numeric argument that does not immediately follow a '`bc`' keyword.

For information on real as compared to complex Fourier transformation, see the description of `proc` or `procl`. For information on the lfs (low-frequency suppression) and zfs (zero-frequency suppression) solvent suppression options, see the description of parameters `ssfilter` and `ssorder`, and the macro `parfidss`.

Arguments: options can be any of the following (the order is not important):

- '`ntype`', '`t2dc`', '`t1dc`', and '`f2sel`' are keywords that function the same as when supplied to the `ft2d` and `wft2d` commands. Refer to the `ft2d` command for a description of these options.
- '`bc`' is a keyword for a baseline correction of the phase-corrected f_2 spectra prior to the f_1 Fourier transform. The baseline regions must have been previously determined. The default polynomial order is 1, which leads to a spline fit. A different polynomial order can be specified by inserting a numerical argument following '`bc`'.
- '`dc`' is a keyword for a drift correction (`dc`) of the f_2 spectra prior to the f_1 Fourier transformation.
- '`ni`' is a keyword to selectively transform a particular `np-ni` 2D plane within a non-arrayed 3D data set. To identify the plane, '`ni`' is followed by `plane_number`, an integer from 1 through `ni2`.
- '`ni2`' is a keyword to selectively transform a particular `np-ni2` 2D plane within a non-arrayed 3D data set. To identify the plane, '`ni2`' is followed by `plane_number`, an integer from 1 through `ni`.
- '`old`' is a keyword to allow data acquired before the February 25, 1988, VNMR software release to be processed correctly but not to allow a `bc2d` between the f_2 and f_1 Fourier transforms. '`old`' does not function for selective 2D processing within 3D data sets. If no `ni2` or `ni` `plane_number` is given, it is assumed that the data set is only 2D in either `ni2` or `ni`, respectively.

See also: *User Guide: Liquids NMR*

Related:	<code>f1coef</code>	Coefficient to construct F1 interferogram (P)
	<code>f2coef</code>	Coefficient to construct F2 interferogram (P)
	<code>ft1da</code>	Fourier transform phase-sensitive data (M)
	<code>parfidss</code>	Create parameters for time-domain solvent subtraction (M)
	<code>phase</code>	Phase selection (P)
	<code>proc</code>	Type of processing on the <code>np</code> FID (P)
	<code>procl</code>	Type of processing on the <code>ni</code> interferogram (P)
	<code>ssorder</code>	Order of polynomial to fit digitally filtered FID (P)
	<code>ssfilter</code>	Full bandwidth of digital filter to yield a filtered FID (P)
	<code>wft1da</code>	Weight and Fourier transform phase-sensitive data (M)
	<code>wft2da</code>	Weight and Fourier transform phase-sensitive data (M)

`ft2dac` **Combine arrayed 2D FID matrices (M)**

Syntax: `ft2dac(<mult1><,mult2>,...<,multn>)>`

Description: Allows ready combination of 2D FID matrices within the framework of the 2D FT program. No weighting is performed. Data must be acquired either without f_1 quadrature or with f_1 quadrature using the TPPI method. `ft2dac` is used with TOCSY (with multiple mixing times).

Arguments: `mult1`, `mult2`, ..., `multn` are multiplicative coefficients. The `n`th argument is a real number and specifies the coefficient for the `n`th 2D FID matrix.

See also: *User Guide: Liquids NMR*

Related:	<code>ft1dac</code>	Combine arrayed 2D FID matrices (M)
	<code>tocsy</code>	Set up parameters for a TOCSY pulse sequence (M)
	<code>wft1dac</code>	Combine arrayed 2D FID matrices (M)
	<code>wft2dac</code>	Combine arrayed 2D FID matrices (M)

`ft3d` **Perform a 3D Fourier transform on a 3D FID data set (M,U)**

Description: Transforms 3D FID data into 3D spectral data. `ft3d` can be entered from a VNMR macro or directly from UNIX. Each type of entry is described below. A final section explains the `ft3d` coefficient file.

Additional parameter control for the operation of `ft3d` is available. This allows drift corrections and partial Fourier transformation. See the descriptions of `specdc3d`, `fiddc3d`, and `ptspec3d` for information.

ft3d Entered from VNMR

Syntax: (From VNMR) `ft3d(<data_directory><,number_files>
<,'nocoeff'><,'t1t2'|'t2t1'><,'fdf'><,'nofdf'>
<,plane_type>)>`

Description: Executes the program `ft3d` in the VNMR system `bin` directory. The 3D FID data must be loaded into the experiment in which the `ft3d` macro is to be run. `ft3d` is started up in background mode by this macro so that VNMR remains free for interactive processing. You can start a 3D transform from within `exp4` and, at the same time, continue with any 1D or 2D processing of the 3D FID data within the same experiment using VNMR.

Distributed f_1f_2 processing has the following system and network requirements:

- The master host system (the system on which the macro `ft3d` is executed from within VNMR) must define the names of the networked computers

that are to participate in the distributed processing. The file `/etc/hosts.3D` must contain these names in the following format:

```
unity1
unity2
datastation1
datastation2
```

- Each participating computer must recognize the name of the user that started up the master `ft3d` program as a valid user name on its system. For example, if user `steve` issues the `ft3d` command within VNMR running on computer `unity0`, `steve` must be a valid user on all other computer systems that are to be used in the distributed f_1f_2 processing.
- Each computer system must have NFS access to the 3D data directory.

Arguments: The order of the arguments is not important.

`data_directory` (without the `/data` subdirectory appended) specifies the output directory for the 3D spectral data file(s). The default directory for the 3D spectral data is `curexp/datadir3d`.

`number_files` sets the number of 3D data files (`data1, data2, ... data n` , where n is `number_files`) used to store the transformed 3D data. `number_files` must be an integer and be 32 or less. When `number_files` is entered, distributed f_1f_2 processing is performed by `ft3d` if possible.

'`nocoeff`' is a keyword for the `set3dproc` command within the `ft3d` macro to not create a 3D coefficient file prior to invoking the `ft3d` program. This option is useful if you have modified an existing 3D coefficient file and do not want it to be overwritten prior to the 3D transform. See below for information on coefficient files. By default, `ft3d` calls the `make3dcoef` macro to create a coefficient file using the `f1coef` and `f2coef` string parameter values.

'`t1t2`' and '`t2t1`' are keywords to explicitly define the order of the `t1` and `t2` arrays (other than `ni` and `ni2`). By default, `ft3d` looks at the `array` parameter and if any parameter other than `phase` and `phase2` are arrayed, the macro aborts.

'`fdf`' indicates that the output of `ft3d` is to be an FDF (Flexible Data Format) file named `data.fdf`. This is the default if the parameter `appmode` is set to '`imaging`'. Distributed processing can still be performed if `number_files` is set appropriately. 3D FDF files can be viewed with the `disp3d` program, or selected slices can be extracted with ImageBrowser (started by the `browser` command from UNIX).

'`nofdf`' indicates that the final output is the group of `data1, data2, ...` files, and that no FDF format file should be produced. This is the default if the parameter `appmode` is *not* set to '`imaging`'.

`plane_type` sets plane extraction following the complete 3D FT with the following keywords:

- '`xall`' indicates that all three 2D plane types, f_1f_3 , f_2f_3 , and f_1f_2 , are to be automatically extracted at the end of the 3D Fourier transform.
- '`f1f3`', '`f2f3`', and '`f1f2`' can be used to select any combination of plane types to be extracted.

Any of these options can be submitted more than once to the `ft3d` program, but the `getplane` program will display an error and abort if any one plane type is defined for extraction more than once.

Examples: (From VNMR) `ft3d`
 (From VNMR) `ft3d('nocof', 'f1f3', 'f2f3')`

ft3d Entered from UNIX

Syntax: (From UNIX) `ft3d -e exp_number -f -r <options>`

Description: The `ft3d` program can also be run directly from the UNIX environment on the host computer. An information file must be present before `ft3d` can execute successfully but it need contain only valid processing information for the t_3 dimension and valid Fourier numbers for the t_1 and t_2 transforms. Valid weighting and phasing parameters for the f_1 and f_2 dimension do not need to be set while `wftt3` executes. After several FIDs have been collected, you can determine acceptable f_3 weighting and phasing parameters. After setting `fn1` and `fn2` to the desired values, the 3D processing information file can be created by typing `set3dproc` in the VNMR command line. At that point, the next invocation of `ft3d` by the macro `wftt3` causes all (t_1, t_2) increment sets up to and including the current increment in t_3 to be processed.

To start `ft3d` on a remote computer running as a data station for the system, log in as `root` and enter one of the following commands so that the master `ft3d` program can properly communicate with the computer:

- On ^{UNITY}INOVA systems, enter `/vnmr/acqbin/Infoprc &`
- On GEMINI 2000, UNITYplus, UNITY, and VXR-S systems, enter `/vnmr/acqbin/acqinfo_svc &`

With the `Infoprc` or `acqinfo_svc` program running, enter `ft3d` with the `-h` option and the necessary arguments. The `ft3d` program invoked with the `-h` option is considered to be the master program and is responsible for spawning additional remote `ft3d` processes.

Each remote computer must be able to access the 3D data directory as if it were stored on a local disk, must recognize the user name under which the master `ft3d` program is being run, and must also have permission to read from and write to that directory. If the 3D data directory contains four f_3 transformed data files (`data1–data4`), the master `ft3d` program uses the first three remote computer systems listed in file `hosts.3D` that respond.

If the multihost processing option is selected, the number of computers involved will be no more than the number of sets the f_3 spectral data is partitioned into. This number is selected with the `-m` option (see below).

If you are unsure of whether to use `Infoprc` or `acqinfo_svc` on the remote computer, change directories to `/vnmr/acqbin`, enter `lf`, and check which program is present.

Note that if the host computer is rebooted, the background command (`Infoprc` or `acqinfo_svc`) has to be entered again.

Arguments: Note that entering `ft3d` with an ampersand (`&`) after the arguments makes the command execute in the background. As a result, the UNIX prompt reappears after the command is entered and further commands can be entered and executed while the `ft3d` command is processing.

- `-e exp_number` is the experiment number where 3D processing is to occur. This argument is required. It must be written as a minus sign, the letter `e`, a space, and a valid experiment number from 1 to 9 (e.g., `-e 3` sets experiment 3). The experiment must already exist.

The following two options should always be set for reliable operation:

- `-f` specifies that any existing 3D data sets in the experiment should be deleted. This option requires no additional value.

- `-r` calls for explicit data reduction after the 3D Fourier transform. Data reduction consists of retaining only the “real-real-real” part of the completely transformed 3D data set. The `-r` option is mandatory and is enforced within `ft3d` regardless of the user command line input.

options can be any of the following:

- `-F header_file` indicates that an FDF (Flexible Data Format) output file should be produced, using the FDF header found in `header_file`. The output file will be named `data.fdf`, and the `data1`, `data2`, ... files will not be produced.
- `-h` selects the multihost processing option. The `/etc/hosts.3d` file must exist and contain the names of the remote hosts, one host name per line. Each remote host must also have either the program `INFOPROC` or the program `acqinfo_svc` running in the background (one of these programs is already running on any computer being used as a spectrometer host).
- `-l` specifies that a log file be generated in the data subdirectory of the `datadir3d` directory.
- `-m` partitions the f_3 transformed spectral data over more than one data file. This partitioning is necessary if the distributed processing capability of `ft3d` is to be used in performing the remaining f_1 and f_2 transforms. The syntax `-mfiles` is used to specify `nfiles`, the number of data files into which the 3D spectral data is to be divided (e.g., `-m4` specifies 4 data files). Each such data file contains an f_3 subset of the f_1f_2 spectral planes. If `nfiles` is not specified, `ft3d` reports an error and aborts. If `nfiles` is less than an internally calculated value (based on `memsize` and the maximum size for a single 2D transform), the number of data files is set to the internally calculated value; otherwise, `nfiles` determines the number of data files to be used. The maximum number of such files is currently defined to be 32. These 3D data files are labeled `data1`, `data2`, ..., `datan`.
- `-o` specifies an alternative output directory for the processed 3D data. The default directory is `datadir3d` within the current experiment. A full UNIX path must follow the `-o` option.
- `-p` specifies the time-domain dimensions to be processed. If `-p` is used, the processed dimensions can be specified as `f3f2f1`, `f3f2`, `f2f3`, `f2f1`, `f1f2`, `f3`, `f2`, and `f1`. The values `f3f1` and `f1f3` are not allowed because processing must be done sequentially in the order f_3 , then f_2 , and then f_1 . If the `-p` option is not invoked, `ft3d` defaults to `f3f2f1`, resulting in a completely transformed 3D data set.
- `-s` specifies processing of the f_3 dimension of the 3D FID data concurrently with data acquisition. In practice, concurrent f_3 processing is realized by setting `wnt='wftt3'` in the VNMR parameter set and starting the 3D acquisition by entering `au`. The macro `wftt3` handles the call to `ft3d` at the appropriate times during data collection.
- `-x` specifies that plane extractions be performed at the end of 3D processing. The available planes are defined as `f1f2`, `f1f3`, and `f2f3`. If more than one plane extraction is desired, the planes are separated by a colon. For example, `-x f1f2:f1f3:f2f3` would extract all three planes. The planes are placed in the `extr` subdirectory of `datadir3d`.

Examples: (From UNIX) `ft3d -r -f -l -e 2 &`

(From UNIX) `ft3d -r -f -l -e 2 -x f1f2:f1f3:f2f3 &`

See also: *User Guide: Liquids NMR*

Related:	<code>appmode</code>	Application mode (P)
	<code>browser</code>	Start ImageBrowser application (U)
	<code>dconi</code>	Interactive 2D data display (C)
	<code>disp3d</code>	Display 3D data (U)
	<code>fiddc3d</code>	3D time-domain dc correction (P)
	<code>f1coef</code>	Coefficient to construct F1 interferogram (P)
	<code>f2coef</code>	Coefficient to construct F2 interferogram (P)
	<code>getplane</code>	Extract planes from a 3D spectral data set (M)
	<code>killft3d</code>	Terminate any <code>ft3d</code> process started in an experiment (M,U)
	<code>make3dcoef</code>	Make 3D coefficients file from 2D coefficients (M)
	<code>ptspec3d</code>	Region-selective 3D processing (P)
	<code>set3dproc</code>	Set 3D processing (C)
	<code>specdc3d</code>	3D spectral dc correction (P)
	<code>wftt3</code>	Process f_3 dimension during 3D acquisition (M)

full **Set display limits for a full screen (C)**

Syntax: `full`

Description: Sets the horizontal control parameters (`sc` and `wc`) and the vertical control parameters (`sc2` and `wc2`) to produce a display (and subsequent plot) on the entire screen (and page). For 2D data, space is left for the scales. If a 1D interactive spectral display is active, the display is automatically updated; for 2D displays, an appropriate command or the Menu button must be used to cause redisplay to occur.

Alternate: Full Screen button on the 1D Display Size Selection Menu, or Full Screen button on the 2D Display Size Selection Menu.

See also: *Getting Started; User Guide: Liquids NMR*

Related:	<code>center</code>	Set display limits for center of screen (C)
	<code>fullt</code>	Set display limits for full screen with room for traces (C)
	<code>left</code>	Set display limits for left half of screen (C)
	<code>right</code>	Set display limits for right half of screen (C)
	<code>sc</code>	Start of chart (P)
	<code>sc2</code>	Start of chart in second direction (P)
	<code>wc</code>	Width of chart (P)
	<code>wc2</code>	Width of chart in second direction (P)

fullsq **Display largest square 2D display (M)**

Syntax: `fullsq`

Description: Adjusts `sc`, `sc2`, `wc`, and `wc2` parameters to show the largest possible square 2D display.

Related:	<code>full</code>	Set display limits for a full screen (C)
	<code>fullt</code>	Set display limits for a full screen with room for traces (C)
	<code>sc</code>	Start of chart (P)
	<code>sc2</code>	Start of chart in second direction (P)
	<code>wc</code>	Width of chart (P)
	<code>wc2</code>	Width of chart in second direction (P)

fullt **Set display limits for a full screen with room for traces (C)**

Syntax: `fullt`

Description: Sets the horizontal control parameters (**sc** and **wc**) and the vertical control parameters (**sc2** and **wc2**) to produce a display (and subsequent plot) in the entire screen (and page) with room for traces (**dconi**). For 2D data, space is left for the scales.

Alternate: Full with Traces button in the 2D Display Size Selection Menu.

See also: *User Guide: Liquids NMR*

Related:	center	Set display limits for center of screen (C)
	full	Set display limits for a full screen (C)
	left	Set display limits for left half of screen (C)
	right	Set display limits for right half of screen (C)

g2pul **Set up pulse sequence for gradient evaluation (M)**

Applicability: Systems with the pulsed field gradient or imaging module.

Syntax: `g2pul`

Description: Performs gradient recovery measurements. With `gzlvl1` on during `gt1`, the system recovery to homogeneity can be measured after delay `d2`. Typical values are `gt1=0.040` (40 ms) and gradient strength on full (`gzlvl=32767`). `g2pul` sets an experiment environment suitable for these tests. The `gradaxis` parameter is used by `g2pul` to select the x, y, or z gradient axis.

See also: *VNMR User Programming*

Related: `gradaxis` Select gradient axis (P)

ga **Submit experiment to acquisition and FT the result (M)**

Syntax: `ga(<'nocheck'><,<'next'><,<'wait'>>>`

Description: Performs experiment described by the current acquisition parameters, checking parameters `loc`, `spin`, `gain`, `wshim`, `load`, and `method` to determine the necessity to perform various actions in addition to simple data acquisition. This may involve a single FID or multiple FIDs, as in the case of arrays or 2D experiments. `ga` causes the data to be automatically weighted and Fourier transformed (`wft`) at the end of each FID data acquisition.

Before starting the experiment, `ga` executes two user-created macros if they exist. The first is `usergo`, a macro that allows the user to set up general conditions for the experiment. The second is a macro whose name is formed by `go_` followed by the name of the pulse sequence (from `seqfil`) to be used (e.g., `go_s2pul`, `go_dept`). The second macro allows a user to set up experiment conditions suited to a particular sequence.

Arguments: `'nocheck'` is a keyword to override checking if there is insufficient free disk space for the complete 1D or 2D FID data set to be acquired.

`'next'` is a keyword to put the experiment started with `ga('next')` at the head of the queue of experiments to be submitted to acquisition.

`'wait'` is a keyword to stop submission of experiments to acquisition until `wexp` processing of the experiment, started with `ga('wait')`, is finished.

Alternate: Go,Wft button in the Acquire Menu.

See also: *Getting Started*

Related:

<code>au</code>	Submit experiment to acquisition and process data (M)
<code>change</code>	Submit a change sample experiment to acquisition (M)
<code>gain</code>	Receiver gain (P)
<code>go</code>	Submit experiment to acquisition (M)
<code>go_</code>	Pulse sequence setup macro called by <code>go</code> , <code>ga</code> , and <code>au</code> (M)
<code>load</code>	Load status of displayed shims (P)
<code>loc</code>	Location of sample in tray (P)
<code>lock</code>	Submit an Autolock experiment to acquisition (C)
<code>method</code>	Autoshim method (P)
<code>sample</code>	Submit change sample, Autoshim experiment to acquisition (M)
<code>seqfil</code>	Pulse sequence name (P)

<code>shim</code>	Submit an Autoshim experiment to acquisition (C)
<code>spin</code>	Submit a spin setup experiment to acquisition (C)
<code>spin</code>	Sample spin rate (P)
<code>su</code>	Submit a setup experiment to acquisition (M)
<code>usergo</code>	Experiment setup macro called by <code>go</code> , <code>ga</code> , and <code>au</code> (M)
<code>wft</code>	Weight and Fourier transform 1D data (C)
<code>wshim</code>	Conditions when shimming is performed (P)

gadm **Display *GLIDE* administration tool (C)**

Syntax: `gadm`

Description: Displays an administration tool for users to create their own experiment and solvent lists in the *GLIDE* interactive window. The VNMR administrator can use `gadm` to create groups of users, with each group assigned different experiment lists and solvent lists, as well as to allow or disallow users saving data to disk.

See also: *Walkup NMR Using GLIDE*

Related: `glide` Interactive window data acquisition and processing (C)

gain **Receiver gain (P)**

Description: Sets receiver gain or, by setting `gain='n'`, enables Autogain for automatic adjustment of gain. Low gain in multiline, high-dynamic-range samples can cause a number of problems, including intermodulation distortions and extra lines in the spectrum. Too high a gain, on the other hand, can cause receiver overload and consequent baseline distortions. Autogain capability allows the observe channel to be set optimally for detecting and digitizing NMR signals from a wide variety of samples.

Autogain adjusts the observe channel gain such that the NMR signal takes about 50 percent of the maximum range of the ADC. This setting allows a comfortable leeway for variations in signal. The program begins acquisition in the normal manner but the first transient (after any requested steady state transients) is examined for signal level. If the intensity is too low or too high, the gain is changed and the process is repeated until the intensity is within the proper range, and then normal acquisition commences. The final gain value used for the experiment is stored and when the experiment is finished, setting `gain='y'` results in the value being displayed in the `dgs` parameter group.

If the gain is reduced by the Autogain procedure such that the noise does not trigger the least significant 1 or 2 bits in the ADC and the signal still overloads either the receiver or ADC, the system stops and displays a message indicating Autogain failure.

Values: On *MERCURY-VX* and *MERCURY* systems, 0 to 38, in steps of 2 dB (38 represents the highest possible receiver gain and 0 the lowest).

On *GEMINI 2000* systems, 0 to 40, in steps of 2 dB (40 represents the highest possible receiver gain and 0 the lowest).

On systems other than *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*, 0 to 60, in steps of 2 dB (60 represents highest possible receiver gain and 0 lowest). On 500-, 600-, and 750-MHz *UNITY INOVA* and *UNITYplus*, low-band gain is limited from 18 to 60.

'n' enables Autogain, in which the gain is automatically adjusted at the start of acquisition for an optimum value. After the acquisition is finished, setting

`gain='y'` then allows the value of `gain` to be read. `gain='n'` may not be used for arrayed experiments.

See also: *Getting Started*

Related: `dgs` Display group of special/automation parameters (M)
`gf` Prepare parameters for FID/spectrum display in `acqi` (M)

gap Find gap in the current spectrum (M)

Syntax: `gap(gap,height):found,position,width`

Description: Looks for a gap between the lines of the currently displayed spectrum. It can be used to automatically place inserts, parameter printouts, trace labels, etc. The search starts on the left side (low-field end) of the spectrum.

Arguments: `gap` is the width of the desired gap.

`height` is the starting height (same as the lower limit for the insert).

`found` is a return value that is set to 1 if the search is successful, or set to 0 if unsuccessful.

`position` is a return value that is set to the distance from the left edge of the chart (not the plot) to the left end of the gap (3 mm from the nearest peak to the left, positioning with “left gravity”) if the search is successful, or set to the position (no spacing to the nearest line) of the largest gap found if unsuccessful.

`width` is a return value set to the total width of the first gap if the search is successful, or set to the width of largest gap found if unsuccessful.

Examples: `gap(120,80);$1,$2,$3`

See also: *VNMR User Programming*

gaussian Set up unshifted Gaussian window function (M)

Syntax: `gaussian(<t1_inc><,t2_inc>)>`

Description: Sets up an unshifted Gaussian window function in 1, 2, or 3 dimensions. The macro checks whether the data is 1D, 2D, and 3D.

Arguments: `t1_inc` is the number of t1 increments. The default is `ni`.

`t2_inc` is the number of t2 increments. The default is `ni2`.

See also: *Getting Started; User Guide: Liquids NMR*

Related: `ni` Number of increments in 1st indirectly detected dimension (P)
`ni2` Number of increments in 2nd indirectly detected dimension (P)
`pi3ssbsq` Set up pi/3 shifted sinebell-squared window function (M)
`pi4ssbsq` Set up pi/4 shifted sinebell-squared window function (M)
`sqcosine` Set up unshifted cosine-squared window function (M)
`qsinebell` Set up unshifted sinebell-squared window function (M)

gcal Gradient calibration constant (P)

Applicability: Systems with the pulsed field gradient or the imaging module.

Description: Stores the proportionality constant between the parameter values (DAC units) controlling the desired gradient and the intensity of the gradient expressed in gauss/cm. The gradients generated in the magnet require calibration of the gain on the gradient compensation board so that coordinate data, slice positions, and the field of view can be set up accurately. `gcal` should be located in each user's `vnmr/sys/global` file.

Values: Number that is probe dependent, in gauss/cm-DAC unit. On the Performa I PFG module, 0.00028 to 0.00055 gauss/cm-DAC unit is nominal; On the Performa II, 0.0014 to 0.0025 gauss/cm-DAC unit is nominal.

See also: *User Guide: Imaging*

Related: `ecctabl` Put `gcal` value and `ecc` file into table (M)
`getgcal` Get `gcal` value from table (M)
`setgcal` Set gradient calibration constant (M)

`gcoil` **Current gradient coil (P)**

Description: Reserved parameter that specifies which physical gradient set is currently installed. This allows convenient updating of important gradient characteristics when one gradient set is interchanged for another. When set, `gcoil` reads the gradient table file of the same name in `/vnmr/imaging/gradtables` and sets the gradient calibration parameters.

`gcoil` is local to each individual experiment. It is normally set the same as `sysgcoil` for acquiring new data, but can be set to other gradient names when working with saved data or data from another instrument. Each possible gradient name should have an associated file of that name located in the directory `/vnmr/imaging/gradtables`. Look at any file in this directory for an example of the proper `gradtable` format, or use the macro `createtable` to make new `gradtables` entries.

If the parameter `gcoil` does not exist in a parameter set and a user wants to create it, you must set the protection bit that causes the macro `_gcoil` to be executed when the value for `gcoil` is changed. There are two ways to create `gcoil`:

- Use the macro `updtgcoil`, which will create the `gcoil` parameter if it does not exist and set the correct protection bits.
- Enter the following commands:

```
create('gcoil','string')
setprotect('gcoil','set',9)
```

`gcoil` and the associated gradient calibration parameters `boresize`, `gmax`, and `trise` are updated with the values listed in the table on the right each time a parameter set is retrieved, or when an experiment is joined. In the rare case that a `gradtables` file is

Variable Name	Value
<code>boresize</code>	22.50 cm
<code>gmax</code>	5.00 gauss/cm
<code>trise</code>	0.000500 sec

modified, but the value of `gcoil` is not changed, manually force an update of the calibration parameters. Updating may be accomplished either by setting `gcoil` to itself, for example, `gcoil=gcoil`, or by using the macro `_gcoil`.

Be aware that if an old dataset is returned and processed, gradient parameters associated with that dataset will replace any new `gcoil` parameters.

The table above is a gradient table (gradient coil name: asg33) for a horizontal imaging system with all three axes set to the same maximum gradient strength.

On the right is a gradient table (gradient coil name: tc203) for a three-axis gradient set with unequal maximum gradient strength.

<i>Variable Name</i>	<i>Value</i>
boresize	5.10 cm
trise	0.000200 sec
gxmax	29.00 gauss/cm
gymax	27.00 gauss/cm
gzmax	70.00 gauss/cm

Related:	boresize	Magnet bore size (P)
	creategtable	Generate new gradient calibration file (M)
	gmax	Maximum gradient strength (P)
	setgcoil	Assign sysgcoil configuration parameter (M)
	sysgcoil	System gradient coil (P)
	trise	Gradient rise time (P)
	updtgcoil	Update gradient coil (M)

See also: *User Programming*

gCOSY **Change parameters for gCOSY experiment (M)**

Syntax: `gCOSY<('GLIDE')>`

Description: Converts the current parameter set to a gCOSY experiment.

Arguments: 'GLIDE' is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding proton spectrum for the experiment.

Related: **COSY** Change parameters for COSY experiment (M)

gcosy **Set up pulse sequence for gradient COSY (M)**

Applicability: Systems with the pulsed field gradient or the imaging module.

Syntax: `gcosy`

Description: Converts a 1D standard two-pulse sequence parameter set into a set ready to run a PFG (pulsed field gradient) absolute-value COSY experiment.

See also: *User Guide: Liquids NMR*

gcrush **Crusher gradient level (P)**

Description: Predefined parameter available for use in setting a crusher gradient level, often paired with the timing parameter `tcrush`.

See also: *User Guide: Imaging*

Related: **gspoil** Spoiler gradient level (P)
tspoil Gradient spoiling time (P)

gdiff **Diffusion gradient level (P)**

Description: Predefined parameter available for use in setting a diffusion gradient level, often paired with the timing parameters `tdiff` or `tdelta`.

get1d **Select a 1D experiment for processing (M)**

Syntax: `get1d<(experiment)>`

Description: In nonautomation mode, the macros `hcosy`, `hcapt`, `capt`, `hcdept`, and `cdept` all acquire two or more data sets in the experiment in which the macro was executed. These data sets are stored, complete with Fourier transformed data. The data sets are also stored directly in the experiment. The `get1d` macro is used to select which data set should be active for processing in that experiment. After `get1d` is executed, data can be stored in the conventional way with the `svf` command (e.g., when `hcosy` completes, `get1d` can be used to process the 1D data set).

Arguments: `experiment` is the 1D data set to be used for processing. The default is the 'H1' experiment.

Examples: `get1d`
`get1d('apt')`

See also: *Getting Started*

Related:

<code>capt</code>	Automated carbon and APT acquisition (M)
<code>cdept</code>	Automated carbon and DEPT acquisition (M)
<code>get2d</code>	Select a 2D experiment for processing (M)
<code>hcapt</code>	Automated proton, carbon, and APT acquisition (M)
<code>hcdept</code>	Automated proton, carbon, and DEPT acquisition (M)
<code>hcosy</code>	Automated proton and COSY acquisition (M)
<code>svf</code>	Save FIDs in current experiment (C)

get2d **Select a 2D experiment for processing (M)**

Syntax: `get2d<(experiment)>`

Description: In nonautomation mode, the macros `hcosy`, `hcapt`, `capt`, `hcdept`, and `cdept` all acquire two or more data sets in the experiment in which the macro was executed. These data sets are stored complete with Fourier transformed data. The data sets are also stored directly in the experiment. The `get2d` macro is used to select which data set should be active for processing in that experiment. After entering `get2d`, data may be stored in the conventional way with the `svf` command. For example, following completion of `hcosy`, `get2d` can be used to process the 2D data set.

Arguments: `experiment` is the 2D data set that should be used for processing. The default is the 'relayh' experiment.

Examples: `get2d('hetcor')`

See also: *Getting Started*

Related:

<code>get1d</code>	Select a 1D experiment for processing (M)
<code>svf</code>	Save FIDs in current experiment (C)

getdim **Return dimensionality of experiment (M)**

Syntax: `getdim:dimensions`

Description: Used in other macros to determine the number of dimensions of the current data set. Many macros make decisions based on whether a data set is multidimensional or 1D. `getdim` makes it easier to access this information.

Arguments: `dimensions` is a return variable giving the number of dimensions of the data. If `ni3` is 2 or greater, `dimensions` is set to 4; if `ni2` is 2 or greater, `dimensions` is set to 3; if `ni` is 2 or greater, `dimensions` is set to 2; and if `ni` is less than 2 or undefined, `dimensions` is 1.

Examples: `getdim:r1`

See also: *Getting Started*

Related: **ni** Number of increments in 1st indirectly detected dimension (P)
ni2 Number of increments in 2nd indirectly detected dimension (P)
ni3 Number of increments in 3rd indirectly detected dimension (P)

getfile Get information about directories and files (C)

Syntax: (1) `getfile(directory):$number_files`
 (2) `getfile(directory,file_index):$file,$extension`

Description: Returns information about the number of files in a directory or about a particular file in a directory.

Arguments: `directory` is the name of the directory for which information is desired.
`number_files` is the number of files in the directory, with dot files (e.g., `.login`) ignored.
`file_index` is the number of file for which information is desired (the order is UNIX-dependent).
`file` is the name of the file, excluding any extension, identified by the `index` (see examples below).
`extension` is the extension of the file name identified by the `file_index`. For example, if `file_index` points to the file named `s2pul.fid`, `getfile` returns the string `s2pul` to `$file` and the string `fid` to `$extension`. If the file name pointed to has no extension (e.g., `dummy`), no value is returned to `$extension`. If the file name has more than one extension, only the last extension is returned to `$extension` (e.g., the file `fid.tmp.par` returns `fid.tmp` to `$file` and `par` to `$extension`).

Complete paths (full file names) can be reconstructed like this:

```
getfile('dir',i):$filename,$ext
if ($ext='') then $path='dir'+ '/' +$filename
else $path='dir'+ '/' +$filename+'.'+$ext
endif
```

Paths for the `rt` command can be reconstructed like this:

```
$path='dir'+ '/' +$filename.
```

Examples: `getfile('dir'):$entries`
`$temp = 0`
`while ($temp < $entries)`
 `$temp = $temp + 1`
 `getfile('dir',$temp):$filename,$ext`
 `...`
`endwhile`

See also: *VNMR User Programming*

getgcal Get gcal value from table (M)

Applicability: Systems with the imaging module.

Syntax: `getgcal<(ecc_file)>`

Description: Retrieves value of the gradient calibration constant `gcal` from the reference table `ecctabl` in the directory `$vnmrsystem/imaging/eddylib`. If the value would overwrite the current value of `gcal`, the monitor displays a prompt to confirm the overwrite.

Arguments: `ecc_file` specifies the name of the `ecc` file in the reference table `ecctabl`. The default value is `'curecc'`.

Examples: `getgcal`
`getgcal('test1')`

See also: *User Guide: Liquids NMR*

Related: `ecc` Set up parameters to obtain compensation data (M)
`ecctabl` Put `gcal` value and `ecc` file into table (M)
`gcal` Gradient calibration constant (P)

getll Get intensity and line frequency of line (C)

Syntax: `getll(line_number)<:height,frequency>`

Description: Finds the height and frequency of line from a line listing. It assumes a previous line list using `dll`.

Arguments: `line_number` is the number of the line in the line list.
`height` is the intensity of the specified line.
`frequency` is the line frequency with units defined by the parameter `axis`.

See also: *VNMR User Programming*

Related: `axis` Axis label for displays and plots (P)
`dll` Display listed line frequencies and intensities (C)
`fp` Find peak heights (C)
`nll` Find line frequencies and intensities (C)

getparam Retrieve parameter from probe file (M)

Syntax: `getparam(param<,nucleus>):$value`

Description: Retrieves the value of a parameter from the current probe file. The name of the probe file is referenced from the parameter `probe`.

Arguments: `param` is the name of the parameter to be retrieved.
`nucleus` is the nucleus to be retrieved from the probe file. The default is the current value of the parameter `tn`
`value` is a return variable with the value of the retrieved parameter.

Examples: `getparam('tpwr'):tpwr`
`getparam('dmf','H1'): $dmf`

See also: *Getting Started*

Related: `addnucleus` Add new nucleus to existing probe file (M)
`addparams` Add parameter to current probe file (M)
`addprobe` Create new probe directory and probe file (M)
`probe` Probe type (P)
`setparams` Write parameter to current probe file (M)
`tn` Nucleus for the observe transmitter (P)
`updateprobe` Update probe file (M)

getplane Extract planes from a 3D spectral data set (M)

Applicability: All systems; however, although `getplane` is available on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems, such systems can only process 3D data and cannot acquire 3D data.

Syntax: `getplane(<<data_dir><,<plane_dir><,<plane_type>>>`

Description: Executes the program `getplane` in the VNMR system bin directory (`$vnmrsystem/bin`). `getplane` checks whether there is sufficient file space on the disk partition to accommodate the extracted planes. If space is insufficient, `getplane` writes an error to the VNMR text window and aborts. `getplane` does not delete the output plane directory if it is run multiple times to individually extract different plane types.

Arguments: `data_dir` specifies the directory (without the `/data` subdirectory) containing the input 3D spectral data. The first non-keyword argument to `getplane` is always taken to be `data_dir`.

`plane_dir` specifies the directory (without the `/extr` subdirectory) in which the extracted planes are to be stored. The second non-keyword argument to `getplane` is always taken to be `plane_dir`. If `plane_dir` is not specified, `data_dir` also specifies the output plane directory. If both `data_dir` and `plane_dir` are not specified, the input data directory and the output plane directory are set to `curexp/datadir3d`. The parameter `plane` is always set equal to the output plane directory.

`plane_type` can be any of the following keywords:

- `'xall'` is a keyword to extract all three 2D plane types: `f1f3`, `f2f3`, `f1f2`.
- `'f1f3'`, `'f2f3'`, `'f1f2'` are keywords to extract their respective 2D planes.
- Any of these keywords can be submitted more than once to the `getplane` macro, but the `getplane` program displays an error and aborts if any one plane type is defined for extraction more than once.

Examples: `getplane`
`getplane('data3d.inp','data3d.planes','f1f3','f2f3')`

See also: *User Guide: Liquids NMR*

Related:	<code>dplane</code>	Display a 3D plane (M)
	<code>dproj</code>	Display a 3D plane projection (M)
	<code>dsplanes</code>	Display a series of 3D planes (M)
	<code>ft3d</code>	Perform a 3D Fourier transform (M)
	<code>nextpl</code>	Display the next 3D plane (M)
	<code>path3d</code>	Path to currently displayed 2D planes from a 3D data set (P)
	<code>plane</code>	Currently displayed 3D plane type (P)
	<code>plplanes</code>	Plot a series of 3D planes (M)
	<code>prevpl</code>	Display the previous 3D plane (M)

getreg **Get frequency limits of a specified region (C)**

Syntax: `getreg(region_number) <:minimum,maximum>`

Description: Returns the frequency limits of a region. The spectrum should have been previously divided into regions with the `region` command or by manually using a cursor and the `z` command or Resets menu button.

Arguments: `region_number` specifies the number of the region.
`minimum,maximum` are return values set to the frequency limits, in Hz, of the specified region.

Examples: `getreg(1):$a,$b`
`getreg($4):cr,$lo`
`getreg(R1-1):r2,r3`

See also: *VNMR User Programming*

Related:	cz	Clear integral reset points (C)
	ds	Display a spectrum (C)
	numreg	Return the number of regions in a spectrum (C)
	region	Divide spectrum into regions (C)
	z	Add integral reset point at cursor position (C)

getsn **Get signal-to-noise estimate of a spectrum (M)**

Syntax: `getsn:current_sn,predicted_sn`

Description: Estimates spectrum signal-to-noise using the following algorithm:

- Measures four adjacent 5-percent portions at the left edge of the spectrum, finding the root-mean-square noise, and taking the smallest of the four values. By measuring four different values and finding root-mean-square noise instead of peak noise, the result should be reliable even if several signals are present in the selected regions.
- Next, estimates the signal level using the vertical scale adjustment macros: **vsadjh** for proton, **vsadjc** for carbon, and **vsadj** for other nuclei. For carbon spectra, this algorithm ignores solvent lines and TMS. For proton spectra, in addition to ignoring the largest line in the spectrum, if the tallest line is greater than three times the height of the second tallest line, the second highest line is used instead. For other nuclei, `getsn` uses the tallest line in the spectrum.
- Finally, estimates the signal-to-noise at the end of the experiment by a simple extrapolation (multiplying by the square root of **nt/ct**).

Arguments: `current_sn` is a return value set to the current signal-to-noise level.

`predicted_sn` is a return value set to the predicted signal-to-noise level at the end of the experiment.

See also: *User Guide: Liquids NMR*

Related:	ct	Completed transients (P)
	nt	Number of transients (P)
	testsn	Test signal-to-noise ratio (M)
	vsadj	Adjust vertical scale (M)
	vsadjc	Adjust vertical scale for carbon spectra (M)
	vsadjh	Adjust vertical scale for proton spectra (M)

gettxt **Get text file from VNMR data file (C)**

Syntax: `gettxt(file)`

Description: Copies text from a data file to the current experiment.

Arguments: `file` is the name of a VNMR data file saved from an experiment (i.e., a directory with a `.fid` or `.par` suffix). Do not include the file name suffix.

Examples: `gettxt('/vnmr/fidlib/fid1d')`

See also: *Getting Started*

Related: **puttxt** Put text file into another file (C)

getvalue **Get value of parameter in a tree (C)**

Syntax: `getvalue(parameter<,index><,tree>)`

Description: Gets the value of any parameter in a tree. The value of most parameters can be accessed simply by using their name in an expression. For example, `sw?` or `r1=np` accesses the value of `sw` and `np`, respectively. However, parameters in the processed tree cannot be accessed that way; `getvalue` can be used to get the value of a parameter in the processed tree.

Arguments: `parameter` is the name of an existing parameter.
`index` is the number of a single element in an arrayed parameter. Default is 1.
`tree` is one of the keywords 'global', 'current', 'processed', or 'systemglobal'. The default is 'processed'. Refer to the **create** command for more information on the types of parameter trees.

Examples: `getvalue('arraydim')`

See also: *VNMR User Programming*

Related:	create	Create new parameter in a parameter tree (C)
	display	Display parameters and their attributes (C)
	setgroup	Set group of a parameter in a tree (C)
	setlimit	Set limits of a parameter in a tree (C)
	setprotect	Set protection mode of a parameter (C)
	settype	Change type of a parameter (C)
	setvalue	Set value of any parameter in a tree (C)

gf **Prepare parameters for FID/spectrum display in acqi (M)**

Syntax: `gf`

Description: Provided as a model for preparing parameters for the FID and spectrum display in **acqi**. The unmodified version of this macro turns off phase cycling, autoshimming, autolocking, spin control, temperature control, sample changer control, and autogain. It also selects the current pulse sequence and parameter set by issuing the command `go('acqi')` and the command `acqi('par')`. The automation parameters `cp`, `wshim`, `alock`, `spin`, `temp`, `loc`, and `gain` are then reset to their original values. Users can customize **gf** by copying it into their private `maclib` directory and editing that version to suit their needs.

See also: *Getting Started*

Related:	acqi	Interactive acquisition display process (C)
	alock	Automatic lock status (P)
	cp	Cycle phase (P)
	dmgf	Absolute-value display of FID data and spectrum in <code>acqi</code> (P)
	gain	Receiver gain (P)
	go	Submit an experiment to acquisition (C)
	loc	Location of sample in tray (P)
	spin	Sample spin rate (P)
	temp	Sample temperature (P)
	wshim	Conditions when shimming performed (P)

gf **Gaussian function in directly detected dimension (P)**

Description: Defines a Gaussian time constant of the form $\exp(-(\tau/gf)^2)$ along the directly detected dimension. This dimension is referred to as the f_2 dimension in 2D data sets, the f_3 dimension in 3D data sets, etc.

Values: Number, in seconds. Typical value is `gf='n'`.

See also: *Getting Started*

Related: **gf1** Gaussian function in 1st indirectly detected dimension (P)
gf2 Gaussian function in 2nd indirectly detected dimension (P)
gfs Gaussian shift constant in directly detected dimension (P)

gf1 Gaussian function in 1st indirectly detected dimension (P)

Description: Defines a Gaussian time constant of the form $\exp(-(t/gf1)^2)$ along the first indirectly detected dimension. This dimension is referred to as the f_1 dimension of a multidimensional data set. **gf1** works analogously to the parameter **gf**. The “conventional” parameters, such as **lb** and **gf**, operate on the detected FIDs, while this “2D” parameter is used during processing of the interferograms.

Values: Number, in seconds.

See also: *User Guide: Liquids NMR*

Related: **gf** Gaussian function in directly detected dimension (P)

gf2 Gaussian function in 2nd indirectly detected dimension (P)

Description: Defines a Gaussian time constant of the form $\exp(-(t/gf2)^2)$ along the second indirectly detected dimension. This dimension is referred to as the f_2 dimension of a multidimensional data set. **gf2** works analogously to the parameter **gf**. The **wti** program can be used to set **gf2** on the 2D interferogram data.

Values: Number, in seconds.

See also: *User Guide: Liquids NMR*

Related: **gf** Gaussian function in directly detected dimension (P)
wti Interactive weighting (C)

gflow Flow encoding gradient level (P)

Description: Predefined parameter available for use in setting a flow encoding gradient level, often paired with the timing parameter **tflow**.

See also: *User Guide: Imaging*

gfs Gaussian shift const. in directly detected dimension (P)

Description: Working in combination with the **gf** parameter, **gfs** allows shifting the center of the Gaussian function $\exp(-((t-gfs)/gf)^2)$ along the directly detected dimension. This dimension is referred to as the f_2 dimension in 2D data sets, the f_3 dimension in 3D data sets, etc. Typical value is **gfs = 'n'**.

See also: *Getting Started*

Related: **gf** Gaussian function in directly detected dimension (P)
gfs1 Gaussian shift const. in 1st indirectly detected dimension (P)
gfs2 Gaussian shift const. in 2nd indirectly detected dimension (P)

gfs1 Gaussian shift const. in 1st indirectly detected dimension (P)

Description: Working in combination with the **gf1** parameter, **gfs1** allows shifting the center of the Gaussian function $\exp(-((t-gfs1)/gf1)^2)$ along the first indirectly detected dimension. This dimension is referred to as the f_1 dimension in multidimensional data sets. **gfs1** works analogously to the parameter **gfs**.

The “conventional” parameters (i.e., **lb**, **gf**, etc.) operate on the detected FIDs, while this “2D” parameter is used during processing of the interferograms.

See also: *User Guide: Liquids NMR*

Related: **gf** Gaussian function in directly detected dimension (P)
gf1 Gaussian function in 1st indirectly detected dimension (P)
gfs Gaussian shift const. in directly detected dimension (P)

gfs2 Gaussian shift const. in 2nd indirectly detected dimension (P)

Description: Working in combination with the **gf2** parameter, **gfs2** allows shifting the center of the Gaussian function $\exp(-((t-gfs2)/gf2)^2)$ along the second indirectly detected dimension. This dimension is referred to as the f_2 dimension in multidimensional data sets. **gfs2** works analogously to the parameter **gfs**. The **wti** program can be used to set **gfs2** on the 2D interferogram data.

See also: *User Guide: Liquids NMR*

Related: **gf** Gaussian function in directly detected dimension (P)
gf2 Gaussian function in 2nd indirectly detected dimension (P)
gfs Gaussian shift const. in directly detected dimension (P)
wti Interactive weighting (C)

gHMBC Change parameters for gHMBC experiment (M)

Syntax: `gHMBC<('GLIDE')>`

Description: Converts the current parameter set to a gHMBC experiment.

Arguments: 'GLIDE' is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding proton spectrum for the experiment.

Related: **HMBC** Change parameters for HMBC experiment (M)

ghmqc Set up a PFG HMQC pulse sequence (M)

Applicability: Systems with a pulsed field gradient module.

Syntax: `ghmqc`

Description: Prepares an experiment for a PFG (pulsed field gradient) HMQC using the sequence GHMQC. The sequence sets three gradients, all separately.

Arguments: *User Guide: Liquids NMR*

gHMQC Set up parameters for gHMQC experiment (M)

Syntax: `gHMQC<('GLIDE')>`

Description: Converts the current parameter set to a ^{13}C gHMQC experiment.

Arguments: 'GLIDE' is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding proton spectrum for the experiment.

Related: **HMQC** Change parameters for HMQC experiment (M)
gHMQC15 Set up parameters for ^{15}N gHMQC experiment (M)
gHMQC_d2 Set up parameters for ^{15}N gHMQC using decoupler 2 (M)
gHMQC_d213 Set up parameters for ^{13}C gHMQC using decoupler 2 (M)

gHMQC15 Set up parameters for ^{15}N gHMQC experiment (M)

Syntax: `gHMQC15<('GLIDE')>`

Description: Converts the current parameter set to a gHMQC experiment for ^{15}N .
 Arguments: 'GLIDE' is a keyword to first retrieve the PROTON parameter set for the particular sample.
 Related: [gHMQC](#) Set up parameters for gHMQC experiment (M)

gHMQC_d2 Set up parameters for ^{15}N gHMQC experiment using decoupler 2 (M)

Syntax: `gHMQC_d2<('GLIDE')>`
 Description: Converts the current parameter set to a gHMQC experiment for ^{15}N with decoupler 2 as ^{15}N .
 Arguments: 'GLIDE' is a keyword to first retrieve the PROTON parameter set for the particular sample.
 Related: [gHMQC](#) Set up parameters for gHMQC experiment (M)

gHMQC_d213 Set up parameters for ^{13}C gHMQC experiment using decoupler 2 (M)

Syntax: `gHMQC_d213<('GLIDE')>`
 Description: Converts the current parameter set to a gHMQC experiment for ^{13}C with decoupler 2 as ^{13}C .
 Arguments: 'GLIDE' is a keyword to first retrieve the PROTON parameter set for the particular sample.
 Related: [gHMQC](#) Set up parameters for gHMQC experiment (M)

ghmqcps Set up a PFG HMQC phase-sensitive pulse sequence (M)

Applicability: Systems with a pulsed field gradient module. Not available on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.
 Syntax: `ghmqcps`
 Description: Prepares an experiment for a PFG (pulsed field gradient) HMQC, phase-sensitive version.
 See also: *User Guide: Liquids NMR*

gHMQCTOXY Change parameters for gHMQCTOXY experiment (M)

Syntax: `gHMQCTOXY<('GLIDE')>`
 Description: Converts the current parameter set to a gHMQCTOXY experiment.
 Arguments: 'GLIDE' is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding proton spectrum for the experiment.
 Related: [gHMQC](#) Change parameters for gHMQC experiment (M)
[HMQC](#) Change parameters for HMQC experiment (M)
[HMQCTOXY](#) Change parameters for HMQCTOXY experiment (M)

ghsqc Set up a PFG HSQC pulse sequence (M)

Applicability: Systems with a pulsed field gradient module (except *MERCURY* and *GEMINI 2000*).
 Syntax: `ghsqc<(nucleus)>`
 Description: Converts a 1D standard two-pulse sequence parameter set into a parameter set ready to run a PFG (pulsed field gradient) HSQC experiment, either absolute value or phase sensitive.

Arguments: nucleus is 13C or 15N. The default is 13C.

See also: *User Guide: Liquids NMR*

gHSQC **Set up parameters for gHSQC experiment (M)**

Syntax: gHSQC<(' GLIDE ')>

Description: Converts the current parameter set to a ¹³C gHSQC experiment.

Arguments: ' GLIDE ' is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding proton spectrum for the experiment.

Related: **HSQC** Change parameters for HSQC experiment (M)
gHSQC15 Set up parameters for ¹⁵N gHSQC experiment (M)
gHSQC_d2 Set up parameters for ¹⁵N gHSQC using decoupler 2 (M)
gHSQC_d2 Set up parameters for ¹³C gHSQC using decoupler 2 (M)

gHSQC15 **Set up parameters for ¹⁵N gHSQC experiment (M)**

Syntax: gHSQC15<(' GLIDE ')>

Description: Converts the current parameter set to a gHSQC experiment for ¹⁵N.

Arguments: ' GLIDE ' is a keyword to first retrieve the PROTON parameter set for that particular sample.

Related: **gHSQC** Set up parameters for gHSQC experiment (M)

gHSQC_d2 **Set up parameters for ¹⁵N gHSQC experiment using decoupler 2 (M)**

Syntax: gHSQC_d2<(' GLIDE ')>

Description: Converts the current parameter set to a gHSQC experiment for ¹⁵N with decoupler 2 as ¹⁵N.

Arguments: ' GLIDE ' is a keyword to first retrieve the PROTON parameter set for that particular sample.

Related: **gHSQC** Set up parameters for gHSQC experiment (M)

gHSQC_d213 **Set up parameters for ¹³C gHSQC experiment using decoupler 2 (M)**

Syntax: gHSQC_d213<(' GLIDE ')>

Description: Converts the current parameter set to a gHSQC experiment for ¹³C with decoupler 2 as ¹³C.

Arguments: ' GLIDE ' is a keyword to first retrieve the PROTON parameter set for that particular sample.

Related: **gHSQC** Set up parameters for gHSQC experiment (M)

gHSQCTOXY **Set up parameters for gHSQCTOXY experiment (M)**

Syntax: gHSQCTOXY<(' GLIDE ')>

Converts the current parameter set to a gHSQCTOXY experiment.

Arguments: ' GLIDE ' is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding proton spectrum for the experiment.

Related: **gHSQC** Change parameters for gHSQC experiment (M)
HSQC Change parameters for HSQC experiment (M)
HSQCTOXY Change parameters for HSQCTOXY experiment (M)

gilson **Open the Gilson Control window (C)**

Applicability: UNITY *INOVA* and *MERCURY-VX* only.

Syntax: `gilson`

Description: Opens the Gilson Control window, which enables setup, configuration, and operation of the VAST automatic sampler changer accessory.

See also: *User Guide: Liquids NMR; Sample Management Systems Installation*

gin **Return current mouse position and button values (C)**

Syntax: `gin(<'Bn_press'><'Bn_release'>):$x,$y,$b1,$b2,$b3`

Description: Returns the mouse pointer position and button values. `gin` is most often used with the `draw`, `move`, and `box` commands.

Arguments: 'Bn_press' is a keyword for the mouse button pressed: 'B1_press' for the left button, 'B2_press' for the middle button, or 'B3_press' for the right button. `gin` waits until a button is pressed. For example, given 'B1_press', `gin` waits until button 1 or any key is pressed. If `gin` is waiting for a button press and a keyboard key is pressed, all buttons are set to released (0). The default is to immediately report the mouse position.

'Bn_release' is a keyword for the mouse button released:

'B1_release' for the left button, 'B2_release' for the middle button, or 'B3_release' for the right button. `gin` waits until a button is released. For example, given 'B1_release', `gin` waits until button 1 or any key is released. If `gin` is waiting for a release, all buttons are set to pressed (1). The default is to immediately report the mouse position.

`$x` is the value in the *x* direction, in millimeters, of the pointer. The range of *x* is 0 at the left edge of the chart and `wcmax` at the right edge. If the pointer position is outside the graphics window in the *x* direction, *x* returns -1.

`$y` is the value in the *y* direction, in millimeters, of the pointer. The range of *y* is -20 at the bottom of the chart and `wc2max` at the top. If the pointer position is outside the graphics window in the *y* direction, *y* returns -10000.

`$b1`, `$b2`, `$b3` report the state of the left, middle, and right mouse buttons, respectively. The value is 1 if the corresponding mouse button is down; 0 if the corresponding mouse button is up.

Examples: `gin:$x,$y,$b1,$b2`
`gin('B2_press'):$x,$y,$b1,$b2,$b3`
`gin('B1_release'):$x,$y,$b1`

See also: *VNMR User Programming*

Related:	<code>box</code>	Draw a box on a plotter or graphics display (C)
	<code>draw</code>	Draw line from current location to another location (C)
	<code>move</code>	Move to an absolute location to start a line (C)

glide **Interactive windows for data acquisition and processing (C)**

Syntax: `glide(<'exit'>)`

Description: Provides a mouse-driven method for data acquisition, processing, plotting, and storage. For a general description of how to use *GLIDE*, refer to the manual *Getting Started*. For information on setting up experiments as well as how to adjust icons, text, buttons, and colors, see the manual *Walkup NMR Using GLIDE*.

Arguments: 'exit' is a keyword to exit *GLIDE*.

Examples: `glide`
`glide('exit')`

Alternate: GLIDE button in the Permanent Menu.

See also: *Getting Started, Walkup NMR Using GLIDE*

Related: **AC1-AC9** Automated calibration (M)
gadm Display *GLIDE* administrative tool (C)

globalauto Automation directory name (P)

Description: A global parameter that specifies the name of a directory in which the daily automation directories are saved. This parameter is created and used by the `walkup` macro.

See also: *User Guide: Liquids NMR*

Related: **walkup** Walkup automation (M)

glue Create a pseudo-2D dataset (M)

Applicability: Systems with the LC-NMR accessory.

Syntax: `glue<(num_scans)>`

Description: Steps through the series of FIDs, putting them into `exp5` one by one as an array, and then jumps to `exp5` and changes the parameters `arraydim`, `ni`, and `fn1`, so that the data appear to the user to be a 2D experiment, which can then be processed and displayed with standard 2D commands (`wft2d`, `dconi`, etc.). The parameter `savefile` should exist and should contain the base file name to which a series of FIDs have been saved as `savefile.001`, `savefile.002`, etc.

Arguments: `num_scans` is the number of FIDs copied into the `exp5` array. The default is that `glue` looks for a parameter `nscans` and assumes that all experiments are to be used. Typically, `num_scans` is used if the experiment was aborted prematurely, so that the complete `num_scans` worth of FIDs were not actually acquired.

See also: *User Guide: Liquids NMR*

Related: **nscans** Number of scout/real scan repetitions (P)
savefile Base file name for saving FIDs or data sets (P)

gmapshim Start gradient autoshimming (M)

Applicability: Systems with gradient shimming installed.

Syntax: `gmapshim<('files' | 'mapname' | 'quit')>`

Description: Starts gradient autoshimming if no arguments are used. It can also retrieve a shimmap file or quit gradient autoshimming. When the `gmapshim` program is done, it automatically exits, and the previous data set is retrieved.

CAUTION: Do not spin the sample during gradient shimming.

Arguments: `'files'` is a keyword to enter the gradient autoshimming files menu.

`'mapname'` is a keyword to display the current mapname.

`'quit'` is a keyword to exit from gradient autoshimming and retrieve the previous data set.

Alternate: Gradient Autoslim on Z button in the user gradient shimming menu.

See also: *Getting Started*

Related: `gmapsys` Run gradient autoshimming, set parameters, map shims (M)
`gmapz` Get parameters and files for `gmapz` pulse sequence (M)

`gmapshim_au` **Start acquisition with gradient shimming (M)**

Applicability: Systems with gradient shimming installed.

Syntax: `gmapshim_au`

Description: If `wshim` is not set to 'n', `gmapshim_au` checks the probe file for a lock gradient map name. If the name exists, `gmapshim_au` executes `gmapshim('glideau')` to start gradient shimming followed by acquisition. If the map name does not exist, `gmapshim_au` starts acquisition by running `au('wait')`.

See also: *Getting Started*

Related: `au` Submit experiment to acquisition and process data (M)
`gmapshim` Start gradient autoshimming (M)
`wshim` Conditions when shimming is performed (P)

`gmapsys` **Run gradient autoshimming, set parameters, map shims (M)**

Applicability: Systems with gradient shimming installed.

Syntax: (1) `gmapsys<(option)>`
(2) `gmapsys('shimmap'<,shimmap_option>)`

Description: Enters the Gradient Shimming System menu for setting parameters, mapping the shims, and performing autoshimming. This is the only entry point to the gradient shimming system menu.

If the `gmapz` pulse sequence is not loaded, retrieve parameters from the last `shimmap` used (see current `mapname`) or from `gmapz.par` if no `shimmap` exists.

CAUTION: Do not spin the sample during gradient shimming.

Arguments: `option` is one of the following keywords:

- `'addpar'` adds gradient shimming parameters to the current parameter set.
- `'findgzlvl'` runs an experiment to calibrate `gzlvl`, `gzwin`, and `tof` to optimize the spectral window.
- `'findgzwin'` runs an experiment to calibrate `gzwin` and `tof` to optimize the spectral window.
- `'findtof'` runs an experiment to center `tof` to optimize the spectral window.
- `'rec'` displays the record of shim adjustments from the previous gradient shimming run.
- `'shim'` start autoshimming (same as Autoshim on Z button).
- `'vi'` edits the file `gshim.list`, which is used for editing shim offsets, `mapname`, or selecting coarse and fine shims.
- `'writeb0'` displays the `b0` plot calculated from the first two array elements.

`'shimmap'` is a keyword to run a shim mapping experiment and save the results (same as Make Shimmap button).

`shimmap_option` is one of the following values:

- 'auto' is a keyword to calibrate `gzwin` and then make a shimmap (same as Automake Shimmap button).
- 'manual' is a keyword to use shim offset values set manually from the file `gshim.list` and not the default values to make a shimmap.
- 'overwrite' is a keyword to make a shimmap and overwrite the current mapname if it exists.
- mapname is the prefix of the shimmap file name. The default is the user is queried for mapname before running the experiment.

See also: *User Guide: Liquids NMR*

Related:	<code>gmapshim</code>	Start gradient autoshimming (M)
	<code>gmapz</code>	Get parameters and files for <code>gmapz</code> pulse sequence (M)
	<code>gradtype</code>	Gradients for X, Y, Z axes (P)
	<code>gzwin</code>	Spectral width percentage used for gradient shimming (P)
	<code>seqfil</code>	Pulse sequence name (P)
	<code>gmap_findtof</code>	Gradient shimming flag to first find tof (P)
	<code>gmap_z1z4</code>	Gradient shimming flag to first shim z1-z4 (P)

gmapuser Run gradient autoshimming and set parameters (obsolete)

Description: This macro is no longer used. The `gmapshim` macro has replaced it.

Related: `gmapshim` Start gradient autoshimming (M)

gmapz Get parameters and files for gmapz pulse sequence (M)

Applicability: Systems with gradient shimming installed.

Syntax: `gmapz <(mapname)>`

Description: Retrieves gradient shimming parameters to set up a gradient shimming experiment.

Arguments: mapname is the name of a gradient shimmap file that must exist in the `shimmaps` directory. `gmapz` retrieves parameters and loads the shimmap file from mapname. The default is to retrieve standard gradient shimming parameters from the file `gmapz.par`.

See also: *Getting Started; User Guide: Liquids NMR*

Related:	<code>gmapshim</code>	Start gradient autoshimming (M)
	<code>gmapsys</code>	Run gradient autoshimming, set parameters, map shims (M)
	<code>gmap_findtof</code>	Gradient shimming flag to first find tof (P)

gmap_findtof Gradient shimming flag to first find tof (P)

Applicability: Systems with gradient shimming installed.

Description: When the flag is set to 'y', gradient shimming first performs a calibration to find tof before the start of shimming. This action is recommended for only homospoil deuterium gradient shimming with different solvents. The default value is 'n'.

Values: 'y' turns on the flag.
'n' turns off the flag.

See also: *Getting Started*

Related:	<code>gmapshim</code>	Start gradient autoshimming (M)
	<code>gmapsys</code>	Run gradient autoshimming, set parameters, map shims (M)

`gmapz` Get parameters and files for gmapz pulse sequence (M)
`tof` Frequency offset for observe transmitter (P)

gmap_z1z4 Gradient shimming flag to first shim z1-z4 (P)

Applicability: Systems with gradient shimming installed.

Description: When the flag is set to 'y', if `gzsize` is greater than 4, gradient shimming first shims on z1-z4, and then uses all shims specified by `gzsize`. When the flag is set to 'n' (default), all shims specified by `gzsize` are used.

Values: 'y' turns on the flag.

'n' turns off the flag.

See also: *Getting Started*

Related: `gmapshim` Start gradient autoshimming (M)
`gmapsys` Run gradient autoshimming, set parameters, map shims (M)
`gmapz` Get parameters and files for gmapz pulse sequence (M)
`gzsize` Number of z-axis shims used by gradient shimming (P)

gmax Maximum gradient strength (P)

Description: The allowed maximum gradient level (absolute value) in gauss/cm. `gmax` is one of the calibration entries in a `gradtables` file. `gxmax`, `gymax`, and `gzmax` are used when the maximum gradient level is different for each axis in gauss/cm, which is the case for triple-axis PFG coils.

See also: *VNMR and Solaris Software Installation; User Guide: Imaging*

Related: `boresize` Magnet bore size (P)
`creategtable` Generate new gradient calibration file (M)
`gcoil` Current gradient coil (P)
`gxmax, gymax, gzmax` Maximum gradient strength for each axis (P)
`sysgcoil` System gradient coil (P)
`trise` Gradient rise time (P)

gmqcosy Set up PFG absolute-value MQF COSY parameter set (M)

Applicability: Systems with the pulsed field gradient module.

Syntax: `gmqcosy`

Description: Converts a 1D standard two-pulse sequence parameter set into a parameter set ready to run a PFG (pulsed field gradient) absolute-value MQF COSY experiment.

See also: *User Guide: Liquids NMR*

gnoesy Set up a PFG NOESY parameter set (M)

Applicability: Systems with the pulsed field gradient module.

Syntax: `gnoesy`

Description: Converts a 1D standard two-pulse sequence parameter set into a parameter set ready to run a PFG (pulsed field gradient) NOESY experiment, either absolute value or phase sensitive.

See also: *User Guide: Liquids NMR*

go **Submit experiment to acquisition (M)**

Syntax: `go(<('acqi '><, 'nocheck '><, 'nosafe '><, 'next '><, 'sync '><, 'wait '>>`

Description: Performs the experiment described by the current acquisition parameters, checking parameters `loc`, `spin`, `gain`, `wshim`, `load`, and `method` to determine the necessity to perform various actions in addition to data acquisition. This may involve a single FID or multiple FIDs, as in the case of arrays or 2D experiments. `go` acquires the FID and performs no processing. If free disk space is insufficient for the complete 1D or 2D FID data set to be acquired, `go` prompts the user with an appropriate message and aborts the acquisition initiation process.

Before starting the experiment, `go` executes two user-created macros if they exist. The first is `usergo`, a macro that allows the user to set up general conditions for the experiment. The second is a macro whose name is formed by `go_` followed by the name of the pulse sequence (from `seqfil`) to be used (e.g., `go_s2pul`, `go_dept`). The second macro allows a user to set up experiment conditions suited to a particular sequence.

Arguments: `'acqi '` is a keyword to submit an experiment for display by the `acqi` program. All operations explained above are performed, except acquisition of data is not initiated. The instructions to control data acquisition are stored so that `acqi` can acquire the data when the FID button is clicked. The `gf` macro is recommended instead of running `go('acqi')` directly. Using `gf` prevents certain acquisition events from occurring, such as spin control and temperature change. See the description of `gf` for more information.

`'nocheck '` is a keyword to override checking if there is not enough free disk space for the complete 1D or 2D FID data set to be acquired.

`'nosafe '` is a keyword to disable probe protection during the experiment.

`'next '` is a keyword to put the experiment started with `go('next')` at the head of the queue of experiments to be submitted to the acquisition system. If `go('next')` is entered, the `go` macro remains active until the experiment is submitted to the acquisition system, and no other VNMR commands are processed until the `go` macro finishes.

`'sync '` is a keyword in nonautomation mode that accomplishes the same effect as `go('next')` in synchronizing VNMR command execution with the submission of experiments to the acquisition system. The difference is that `'sync '` does not put the experiment at the head of the queue.

`'wait '` is a keyword to stop submission of experiments to acquisition until `wexp` processing of the experiment, started with `go('wait')`, is finished.

Examples: `go`
`go('nosafe')`
`go('next')`

Alternate: Go button in the Acquire Menu.

See also: *Getting Started*

Related:	<code>acqi</code>	Interactive acquisition display process (C)
	<code>au</code>	Submit experiment to acquisition and process data
	<code>change</code>	Submit a change sample experiment to acquisition (M)
	<code>gain</code>	Receiver gain (P)
	<code>ga</code>	Submit experiment to acquisition and FT the result (C)
	<code>gf</code>	Prepare parameters for FID/spectrum display in <code>acqi</code> (M)
	<code>go_</code>	Pulse sequence setup macro called by <code>go</code> , <code>ga</code> , and <code>au</code> (M)
	<code>load</code>	Load status of displayed shims (P)

<code>loc</code>	Location of sample in tray (P)
<code>lock</code>	Submit an Autolock experiment to acquisition (C)
<code>method</code>	Autoshim method (P)
<code>probe_protection</code>	Probe protection control (P)
<code>sample</code>	Submit change sample, Autoshim exp. to acquisition (M)
<code>seqfil</code>	Pulse sequence name (P)
<code>shim</code>	Submit an Autoshim experiment to acquisition (C)
<code>spin</code>	Submit a spin setup experiment to acquisition (C)
<code>spin</code>	Sample spin rate (P)
<code>su</code>	Submit a setup experiment to acquisition (M)
<code>usergo</code>	Experiment setup macro called by <code>go</code> , <code>ga</code> , and <code>au</code> (M)
<code>wshim</code>	Conditions when shimming is performed (P)

`go_` **Pulse sequence setup macro called by `go`, `ga`, and `au` (M)**

Syntax: `go_macro`

Description: Called by the macros `go`, `ga`, or `au` before starting an experiment. The user typically creates this macro to set up general experiment conditions. The name of the macro is formed by combining `go_` with the name of the pulse sequence macro (from `seqfil`) to be used.

Examples: `go_dept`
`go_noesy`
`go_s2pul`

See also: *Getting Started*

Related:	<code>au</code>	Submit experiment to acquisition and process data (M)
	<code>ga</code>	Submit experiment to acquisition and FT the result (M)
	<code>go</code>	Submit experiment to acquisition (M)
	<code>seqfil</code>	Pulse sequence name (P)
	<code>usergo</code>	Experimental setup macro called by <code>go</code> , <code>ga</code> , and <code>au</code> (M)

`gpat-gpat3` **Gradient shape (P)**

Description: Predefined string parameters available to specify gradient shapes.

See also: *User Guide: Imaging*

`gpe` **Phase encoding gradient increment (P)**

Applicability: Systems with imaging capabilities.

Description: Value of the change in phase encode gradient level from one phase encode step to the next. More precisely, the product of the parameters `gpe` and `tppe` is used internally within the pulse sequence to determine the phase encode gradient increment based on the computed refocusing time for readout and slice selection. `gpe` depends on the field of view and the phase encode gradient duration according to the expression $\gamma \cdot gpe \cdot tppe \cdot lpe = 1$ and is set by either the `imprep` or `setgpe` macros.

See also: *User Guide: Imaging*

Related:	<code>imprep</code>	Set up rf pulses, imaging and voxel selection gradients (M)
	<code>gmax</code>	Maximum gradient strength (P)
	<code>gpe2</code>	Second phase encoding gradient increment (P)
	<code>gpe3</code>	Third phase encoding gradient increment (P)
	<code>lpe</code>	Field of view parameter for phase encode in cm (P)
	<code>nv</code>	Number of 2D phase encode steps to be acquired (P)

`setgpe` Set phase encode gradient levels (M)
`tpe` Duration of the phase encoding gradient pulse (P)

gpmed Phase encode dephasing gradient in the EPI sequence (P)

Applicability: Systems with imaging capabilities.

Description: Determines echo position in the phase-encode direction. A blipped gradient phase encodes the signal with respect to the phase-encode direction. `gpmed` determines the center of the k-space along the phase-encode direction. `gpmed` is usually set so that `eff_echo` appears at the center of the phase encode dimension, t1.

Related: `eff_echo` Effective echo position in EPI experiments (P)

gpemult Phase encode gradient increment multiplier (P)

Applicability: Systems with imaging capabilities.

Description: Multiplier used to correct phase encode gradient increment when using a non-rectangular phase encode gradient shape. For example, a rectangular shaped phase encode gradient has a gradient-time integral equal to 1.571 that of a half-sine gradient of equal duration and peak amplitude. In this case, set `gpemult` to 1.571 to yield the expected field of view.

See also: *User Guide: Imaging*

gradaxis Gradient axis (P)

Applicability: Systems with imaging capabilities.

Description: Selects the gradient axis in macros such as `g2pul` and `profile`.

Values: 'x', 'y', 'z'

See also: *User Guide: Imaging*

Related: `g2pul` Set up pulse sequence for gradient evaluation (M)
`profile` Set up pulse sequence for gradient calibration (M)

gradstepsz Gradient step size (P)

Description: The maximum gradient DAC value. `gradstepsz` determines the type of gradient DAC board used in the system: 12-bit or 16-bit. It is used internally to convert gauss/cm gradient levels to the proper hardware DAC level.

Values: Systems with 12-bit DACs (older SISCO spectrometers without gradient waveform capabilities): -2047 to +2047 units, in integer steps.

Systems with 16-bit DACs (all UNITYplus and beyond, and SISCO spectrometers with gradient waveform capabilities): -32767 to +32767 units, in integer steps.

See also: *VNMR and Solaris Software Installation; User Guide: Imaging*

gradtype Gradients for X, Y, and Z axes (P)

Applicability: Systems with pulsed field gradient (PFG) or imaging capability.

Description: Configuration parameter for systems with optional gradients for axes. The value is set using the label X Axis, Y Axis, Z Axis in the CONFIG window (opened from `config`). The values available for each axis are None, WFG + GCU, Performa I, Performa II/III, Performa II/III + WFG, Performa XYZ, Performa

XYZ + WFG, SIS (12 bit), Homospoil, and Shim DAC. WFG stands for the waveform generator; GCU stands for the gradient compensation unit; and Performa I, II, III, and XYZ are types of PFG modules.

Values: String of three characters (e.g., 'nnp'). The first character is the gradient for the X axis, second for the Y axis, and third for the Z axis. Each axis has value 'n' (None choice in CONFIG window), 'w' (WFG+GCU), 'l' (Performa I), 'p' (Performa II/III), 'q' (Performa II/III + WFG), 't' (Performa XYZ), 'u' (Performa XYZ + WFG), 's' (SIS (12 bit), or 'h' (Homospoil). Homospoil is functional only for the Z axis.

See also: *VNMR and Solaris Software Installation; Getting Started*

Related: `config` Display current configuration and possibly change it (M)
`pfgon` PFG amplifiers on/off control (P)

graphis **Return the current graphics display status (C)**

Syntax: (1) `graphis:$display_command`
 (2) `graphis(command):$yes_no`

Description: Determines what command currently controls the graphics window.

Arguments: `$display_command` is a return value set to the name of the currently controlling command.

`command` is the name of a command to be checked.

`$yes_no` is a return value set to 1 if the command name given by the `command` argument is controlling the graphics window, or set to 0 if it is not controlling the window.

Examples: `graphis:$display`
`if ($display='ds') then`
`...`
`endif`
`graphis('ds'):$ds_on`
`if ($ds_on) then`
`...`
`endif`

See also: *VNMR User Programming*

Related: `textis` Return the current text display status (C)

grayctr **Gray level window adjustment (P)**

Description: Controls the grayscale display available in `dcon`. In the `dconi` program, the center mouse button controls the grayscale bar, which changes the mean gray level and hence the value of `grayctr`. The `grayctr` parameter (along with the parameter `graysl`) records the current settings of the gray bar as the interaction changes; the value can also be set directly. The right mouse button controls the data level of the maximum data intensity. To create `grayctr`, enter `create('grayctr','real')`
`setgroup('grayctr','display')`
`setlimit('grayctr',64,0,1).`

To create the set of imaging parameters `grayctr`, `dcrmv` and `graysl`, and in the current experiment, enter `addpar('image')`.

Values: 0 to 64 (typically 32)

See also: *User Guide: Liquids NMR*

Related: **addpar** Add selected parameters to the current experiment (M)
dcon Display noninteractive color intensity map (C)
dconi Interactive 2D contour display (C)
graysl Gray level slope (contrast) adjustment (P)

graysl Gray level slope (contrast) adjustment (P)

Description: Controls the grayscale display available in **dcon**. In the **dconi** program, the center mouse button controls the grayscale slope as applied to the data changes and hence the value of **graysl**. Negative values of **graysl** will invert black and white; however, negative values can be set only from the keyboard. **graysl** (along with the parameter **grayctr**) records the current settings of the gray bar as the interaction changes; the value can also be set directly. The right mouse button controls the data level of the maximum data intensity. To create **graysl**, enter the following command:

```
create('graysl','real') setgroup('graysl','display')
setlimit('graysl',10,-10,0.1)
```

To create the set of imaging parameters **graysl**, **dcrmv**, and **grayctr** in the current experiment, enter **addpar('image')**.

Values: -10 to +10 (-100 to +100, typically 1)

See also: *User Guide: Liquids NMR*

Related: **addpar** Add selected parameters to the current experiment (M)
dcon Display noninteractive color intensity map (C)
dconi Interactive 2D contour display (C)
grayctr Gray level window adjustment (P)

grecovery Eddy current testing (M)

Applicability: Systems with pulsed field gradient.

Syntax: **grecovery**

Description: Conditions an experiment for eddy current testing so that it is compatible with standard installation procedures.

See also: *Pulsed Field Gradient Modules Installation, VNMR User Guide: Liquids NMR*

grid Draw a grid on a 2D display (M)

Syntax: (1) **grid**<(<spacing><, ><color>)>
(2) **grid**<(start_f2,incr_f2,start_f1,incr_f1,<color>)>

Description: Draws grid lines over a 2D display. Grid lines are drawn on the graphics screen in the XOR mode—entering a second **grid** command with identical arguments erases (not redraws) the grid displayed by the first command.

Arguments: **spacing** specifies the approximate spacing of the grid lines, in cm. The default is intervals of approximately 1 cm, rounded so that the intervals fall at a multiple of 1, 2, or 5 (in Hz), or 1p, 2p, or 5p (in ppm).

color specifies the color of the grid lines and is one of the following keywords: 'red', 'green', 'blue', 'cyan', 'magenta', 'yellow', 'black', or 'white'. The default is 'blue'.

start_f2, **incr_f2**, **start_f1**, **incr_f1** define a grid by supplying the starting and increment frequencies for f2 and f1. Add the p suffix to a value to enter it in ppm (see third example below).

Examples: `grid`
`grid(1.5, 'red')`
`grid(1p, 0.5p, 3p, 0.5p)`

See also: *User Guide: Liquids NMR*

Related: `plgrid` Plot a grid on a 2D plot (M)

griserate Gradient rise rate (P)

Applicability: Systems with imaging capabilities.

Description: Sets the gradient rise rate.

See also: *User Guide: Imaging*

Related: `gcoil` Read data from gradient calibration tables (P)
`gxcal, gycal, gzcal` Gradient calibration constants (P)

gro Readout gradient strength (P)

Applicability: Systems with the or imaging capabilities.

Description: Controls the level of the readout gradient, if present. `imprep` sets `gro` based on its internal algorithm; or use `setgro(value)`, which sets `gro` to a specific value and updates `at` and `sw`. `gro`, `sw`, and `at` are related by the expression $sw = g * lro * gro$, but a change in `lro` does not automatically update `gro` and `sw`.

See also: *User Guide: Imaging*

Related: `at` Acquisition time (P)
`gmax` Maximum gradient strength (P)
`grof` Read out fractional compensation (P)
`gror` Read out compensation gradient (P)
`imprep` Set up rf pulses, imaging and voxel selection gradients (M)
`lro` Field of view size for readout axis (P)
`setgro` Set readout gradient (M)
`sw` Spectral width in directly directed dimension (P)

groa Readout gradient adjuster in EPI experiment (P)

Applicability: Systems with echo planar imaging (EPI) capabilities.

Description: Corrects readout gradient imperfections in EPI experiment by adding an offset (G/cm) to the odd readgradient.

See also: *User Guide: Imaging*

Related: `episet` Set up parameters for EPI experiment (M)
`grora` Readout refocusing gradient adjuster in EPI experiment (P)
`tep` Post-acquisition delay in EPI experiment (P)

gropat Readout gradient shape (P)

Applicability: Systems with imaging capabilities.

Description: Predefined string parameter to specify a readout gradient shape.

See also: *User Guide: Imaging*

gror Read out compensation gradient (P)

Applicability: Systems with imaging capabilities.

Description: Controls the level of the readout refocusing gradient when `pilot= 'n'`. When `pilot= 'y'`, `gror` is ignored by the pulse sequence, and computed internally. In this case the internal value is printed in the window used to start VNMR. `gror` is opposite in sign to `gro` for gradient echo experiments (e.g., FLASH), and has the same sign as `gro` for spin-echo experiments (e.g. SEMS).

Values: Sequence dependent, specified in gauss/cm up to $\pm g_{max}$.

See also: *User Guide: Imaging*

Related: `gmax` Maximum gradient strength (P)
`gro` Read out fractional compensation (P)
`gssr` Slice selection refocusing gradient (P)
`pilot` Automatic sequence setup (P)

grora Readout dephasing gradient adjuster in EPI experiment (P)

Applicability: Systems with echo planar imaging (EPI) capabilities.

Description: Correction gradient value added to the readout refocusing gradient (G/cm) in EPI experiments to center the echo position in the acquisition window.

See also: *User Guide: Imaging*

Related: `episet` Set up parameters in EPI experiment (M)
`groa` Readout gradient adjuster in EPI experiment (P)
`tep` Post-acquisition delay in EPI experiment (P)

groupcopy Copy parameters of group from one tree to another (C)

Syntax: `groupcopy(from_tree, to_tree, group)`

Description: Copies a set of parameters of a group from one parameter tree to another.

Arguments: `from_tree`, `to_tree` are two different parameter trees, each given by the one of the keywords 'global', 'current', or 'processed'. Refer to the `create` command for more information on trees.

`group` is the set of parameters to be copied and is one of the keywords 'all', 'sample', 'acquisition', 'processing', and 'display'.

Examples: `groupcopy('processed', 'current', 'acquisition')`

See also: *VNMR User Programming*

Related: `create` Create new parameter in a parameter tree (C)
`destroy` Destroy a parameter (C)
`destroygroup` Destroy parameters of a group in a tree (C)
`display` Display parameters and their attributes (C)
`setgroup` Set group of a parameter in a tree (C)

gsh2pul Set up parameters for shaped gradients tests (M)

Applicability: Systems with the imaging module.

Syntax: `gsh2pul`

Description: During imaging installation, `gsh2pul` is used to load parameters sets for shaped `gsh2Dpul` gradients tests. `gsh2Dpul` steps the amplifier with the value of `ni`.

Description: *User Guide: Imaging*

Related: `ni` Number of increments in 1st indirectly detected dimension (P)

- gspoil Spoiler gradient level (P)**
 Description: Predefined parameter to set a spoiler gradient level. It is often paired with the timing parameter `tspoil`.
 Related: `tspoil` Spoiling gradient control (P)
- gss Slice selection gradient strength (P)**
 Applicability: Systems with imaging capabilities.
 Description: Controls the level of the slice-select gradient, if present. `imprep` will set `gss` based on the slice thickness and rf pulse bandwidths; or use `setgss` to update only `gss`.
 Values: Number less than $\pm g_{max}$, in gauss/cm.
 See also: *User Guide: Imaging*
 Related: `gmax` Maximum gradient strength (P)
`gssf` Slice selection fractional gradient (P)
`gssr` Slice selection refocusing gradient (P)
`imprep` Set up rf pulses, imaging and voxel selection gradients (M)
`setgss` Select slice or voxel selection gradient levels (M)
`thk` 2D imaging plane slice thickness (P)
- gssf Slice selection fractional refocusing (P)**
 Applicability: Systems with imaging capabilities.
 Description: Fractional multiplier used as a fine tuning adjustment for the `gssr` slice refocusing gradient level.
 Values: 1.0, when the theoretical gradient calculations are correct.
 See also: *User Guide: Imaging*
 Related: `grof` Read out fractional compensation (P)
`gss` Slice selection gradient strength (P)
`gssr` Slice selection refocusing gradient (P)
- gsspat Slice-select gradient shape (P)**
 Description: Predefined string parameter to specify a slice-select gradient shape.
 See also: *User Guide: Imaging*
- gssr Slice selection refocusing gradient (P)**
 Applicability: Systems with imaging capabilities.
 Description: Controls the level of the slice-select refocusing gradient when `pilot='n'`. When `pilot='y'`, `gssr` is ignored by the pulse sequence, and internally computed. The internal value is printed in the window used to start VNMR. `gssr` is normally be opposite in sign to `gss`.
 Values: Number in gauss/cm up to $\pm g_{max}$. Nominal value is `gssr=-0.5*gss`.
 See also: *User Guide: Imaging*
 Related: `gmax` Maximum gradient strength (P)
`gss` Slice selection gradient strength (P)
`gssf` Slice selection fractional gradient (P)
`gror` Read out compensation gradient (P)
`pilot` Automatic sequence setup (P)

- gss2, gss3** **Slice selection gradient level (P)**
 Description: Predefined parameters for specifying gradient levels for different slice selection events in an imaging pulse sequence.
 See also: *User Guide: Imaging*
 Related: [gss](#) Slice selection gradient strength (P)
- gttnoesy** **Set up a PFG TNOESY parameter set (M)**
 Applicability: Systems with the pulsed field gradient (PFG) module. Not available on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.
 Syntax: `gttnoesy`
 Description: Converts a 1D standard two-pulse sequence parameter set into a parameter set ready to run a PFG NOESY experiment (either absolute value or phase sensitive) or a `gttnoesy` experiment.
 See also: *User Guide: Liquids NMR*
- gtnroesy** **Set up a PFG absolute-value ROESY parameter set (M)**
 Applicability: Systems with the pulsed field gradient (PFG) module. Not available on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.
 Syntax: `gtnroesy`
 Description: Converts a 1D standard two-pulse sequence parameter set into a parameter set ready to run a PFG absolute-value ROESY experiment or a `gtnroesy` experiment.
 See also: *User Guide: Liquids NMR*
- gtotlimit** **Gradient total limit (P)**
 Applicability: Systems with three-axis gradients
 Description: Sets the gradient limit, in gauss/cm, of the *x*, *y*, and *z* axes, summed together. This parameter is taken from an entry of the same name in a gradient table and should only exist if a gradient amplifier limits the combined output of all three gradient axis.
 See also: *User Guide: Imaging*
 Related: [createtable](#) Generate system gradient table (M)
 [gcoil](#) Read data from gradient calibration tables (P)
- gtrim** **Trim gradient level (P)**
 Description: Predefined parameter to set a trim gradient level.
 See also: *User Guide: Imaging*
- gvx1-gvox3** **Gradient strength for voxel selection (P)**
 Applicability: Systems with imaging capabilities.
 Description: Voxel-select gradient levels for the first, second, and third dimensions of a voxel in a localized spectroscopy experiment. For example, `imprep` sets `gvox1` based on the corresponding voxel dimension `vovx1`, and rf pulse bandwidth. For nonoblique voxels, the orientation of `gvox1` lies along one of the three main gradient axes, X, Y, or Z. Oblique angle voxel orientation is also available, and for this reason the name `gvox1` is used instead of, for example, `gx`.

Values: Number less than $\pm g_{max}$, in gauss/cm.

See also: *User Guide: Imaging*

Related: `gmax` Maximum gradient strength (P)
`gss` Slice selection gradient strength (P)
`gx` Gradient strength for X, Y, and Z gradients (P)
`vox1, vox2, vox3` Voxel dimension (P)

gx, gy, gz Gradient strength for X, Y, and Z gradients (P)

Applicability: Systems with imaging capabilities.

Description: Defines the gradient strength of the X, Y, and Z gradients, respectively, for localized spectroscopy experiments such as ISIS and VOSY. The gradient strength in conjunction with the length of the selective pulse defines the size of the region of interest.

Values: Number less than to $\pm g_{max}$, in gauss/cm (older pulse sequences, such as `isis.c` and `vosy.c`, use DAC units). The sign is often not important.

See also: *User Guide: Imaging*

Related: `gmax` Maximum gradient strength (P)
`gxcal, gyca1, gzcal` Gradient calibration constants (P)

gxcal, gyca1, gzcal Gradient calibration constants (P)

Applicability: Systems with the older SISCO imaging module.

Description: Stores the proportionality constant for each gradient. The gradients generated in the magnet require calibration so that coordinate data, slice positions, and the field of view can be set up correctly.

Values: Number less than to $\pm g_{max}$, in gauss/cm/DAC (on older SISCO systems).

See also: *User Guide: Imaging*

Related: `gcoil` Read data from gradient calibration tables (P)
`gmax` Maximum gradient strength (P)
`setgcoil` Update system gcoil configuration (M)

gxmax, gymax, gzmax Maximum gradient strength for each axis (P)

Applicability: Systems with three-axis gradients.

Description: Defines the maximum gradient strength, in gauss/cm, for each gradient axis. These values are read in from the selected system gradient table whenever the parameter set is retrieved or the gradient coil defined by `gcoil` has changed. When the values are read in, `gmax` is set to the lowest value of the three.

The parameters `gxmax`, `gymax`, and `gzmax` are used instead of `gmax` when the gradients strengths are not equal for each axis. Unequal gradient strengths per axis are generally true for systems with three-axis PFG coils, which have a strong *z* gradient, and can be true for microimaging systems. Horizontal-bore imaging systems usually have gradients set to the same maximum value, and `gmax` can be used.

See also: *Getting Started; VNMR User Programming, User Guide: Imaging*

Related: `createtable` Generate system gradient table (M)
`gcoil` Read data from gradient calibration tables (P)
`gmax` Maximum gradient strength (P)

gzlvl Pulsed field gradient strength (P)

Applicability: All systems with pulsed field gradient modules.

Description: Specifies the pulsed field gradient DAC value.

Values: Range from +2047 to -2048 for 12-bit gradient module, and from +32767 to -32768 for a 16-bit gradient module.

See also: *Getting Started*

Related: **gzsize** Number of z-axis shims used by gradient shimming (P)
gzwin Spectral window percentage used for gradient shimming (P)

gzsize Number of z-axis shims used by gradient shimming (P)

Applicability: Systems with the pulsed field gradient module.

Description: Specifies the number of z-axis shims used by gradient shimming. For example, **gzsize** set to 4 means that gradient shimming uses shims z1 to z4. By default, coarse shims are used if present, as determined by the **shimset** value

Values: Integer from 1 to 8.

See also: *Getting Started*

Related: **gmapshim** Start gradient autoshimming (M)
gmapsys Run gradient autoshimming, set parameters, map shims (M)
gmapz Get parameters and files for **gmapz** pulse sequence (M)
gzlvl Pulsed field gradient strength (P)
gzwin Spectral width percentage used by gradient shimming (P)
shimset Type of shimset (P)
gmap_z1z4 Gradient shimming flag to first shim z1-z4 (P)

gzwin Spectral width percentage used for gradient shimming (P)

Applicability: Systems with the pulsed field gradient module.

Description: Specifies the percentage of the spectral width **sw** used by gradient shimming for shimmap calculations. The value is set automatically with the buttons Find **gzlvl/gzwin** and Find **gzwin** in the gradient shimming system menu opened by **gmapsys**.

Values: A real number between 0 and 100. The typical value is 50.

See also: *Getting Started*

Related: **gmapshim** Start gradient autoshimming (M)
gmapsys Run gradient autoshimming, set parameters, map shims (M)
gmapz Get parameters and files for **gmapz** pulse sequence (M)
gzlvl Pulsed field gradient strength (P)
gzsize Number of z-axis shims used by gradient shimming (P)
sw Spectral width in directly detected dimension (P)
tof Frequency offset for observe transmitter (P)

H

h1 Automated proton acquisition (M)

Syntax: `h1<(solvent)>`

Description: Prepares parameters for automatically acquiring a standard ^1H spectrum. The parameter `wexp` is set to 'procplot' for standard processing. If `h1` is used as the command for automation via the `enter` command, then `au` is supplied automatically and should not be entered on the MACRO line of the `enter` program. However, it is possible to customize `h1` on the MACRO line by following it with additional commands and parameters. (e.g., entering `h1 nt=1` uses the standard `h1` setup but with only one transient).

Arguments: `solvent` is the name of the solvent. In automation mode, the solvent is supplied by the `enter` program. The default is 'CDCl3'.

Examples: `h1`
`h1('DMSO')`

See also: *Getting Started; User Guide: Liquids NMR*

Related: `au` Submit experiment to acquisition and process data (M)
`enter` Enter sample information for automation run (C)
`h1p` Process 1D proton spectra (M)
`procplot` Automatically process FIDs (M)
`wexp` When experiment completes (P)

h1freq Proton frequency of spectrometer (P)

Description: Configuration parameter for the resonance frequency of ^1H as determined by the field strength of the magnet. The value is set using the label Proton Frequency in the CONFIG window (opened from `config`.)

Values: 085, 100, 200, 300, 400, 500, 600, 750, 800, 900 (in MHz); 3T, 4T.

See also: *VNMR and Solaris Software Installation*

Related: `config` Display current configuration and possibly change it (M)

h1p Process 1D proton spectra (M)

Syntax: `h1p`

Description: Processes non-arrayed 1D proton spectra using standard macros. `h1p` is called by `procl1d`, but can also be used directly. Fully automatic processing (up to a point where a spectrum could be plotted) is provided: Fourier transformation (using preset weighting functions), automatic phasing (`aphx` macro), select integral regions (`hregions` macro), adjust integral size (`integrate` macro), vertical scale adjustment (`vsadjc` macro), avoiding excessive noise (`noislm` macro), threshold adjustment (if required, `thadj` macro), and referencing to the TMS signal if present (`setref` macro, then `tmsref` macro).

See also: *Getting Started; User Guide: Liquids NMR*

Related: `aphx` Perform optimized automatic phasing (M)
`h1` Automated proton acquisition (M)
`hregions` Select integral regions for proton spectra (M)

<code>integrate</code>	Automatically integrate 1D spectrum (M)
<code>noislm</code>	Avoids excessive noise (M)
<code>proclD</code>	Processing macro for simple (non-arrayed) spectra (M)
<code>setref</code>	Set frequency referencing for proton spectra (M)
<code>thadj</code>	Adjust threshold (M)
<code>tmsref</code>	Reference spectrum to TMS line (M)
<code>vsadjh</code>	Adjust vertical scale for proton spectra (M)

h2cal Calculate strength of the decoupler field (C)

Syntax: `h2cal<(j1r, j2r<, j0>)><:gammaH2, pw90, frequency>`

Description: Calculates the strength of the decoupler field. It uses the results from two experiments: one with the decoupler off-resonance at a lower frequency and the other with the decoupler off-resonance at a higher frequency than the frequency of the peak being decoupled.

Arguments: `j1r` is the frequency of the decoupler during these two experiments;. The default is that `h2cal` prompts for a value. If the parameter `dof` is arrayed and has two values, `h2cal` assumes these two values represent the decoupler frequencies; if `dof` is arrayed and has more than two values, `h2cal` prompts for the two decoupler frequencies.

`j2r` is the reduced coupling constants from the two experiments. The default is that `h2cal` prompts for a value

`j0` is the full coupling constant that results when no decoupling is done. The default is a value of 142 Hz, the constant for the standard sample dioxane, or 15 Hz for the methyl iodide sample.

`gammaH2` is a return value set to the strength of the decoupler field.

`pw90` is a return value set to the pulse width of a 90° pulse from the decoupler. It is related to the value of parameter `dmf` through the equation $dmf = 1 / pw90$.

`frequency` is a return value set to the coalescence point (i.e., frequency at which single-frequency decoupling would collapse the dioxane to a singlet).

See also: *Getting Started*

Related: `dmf` Decoupler modulation frequency for first decoupler (P)
`dof` Frequency offset for first decoupler (P)

halt Abort acquisition with no error (C)

Syntax: `halt`

Description: Aborts an experiment that has been submitted to acquisition. If the experiment is active, it is aborted immediately, all data is discarded, and the experiment is interpreted as complete. Any data collected from an earlier block size transfer is retained. If any `wexp` processing is defined, that processing then occurs, followed by any queued experiments. The login name, and the FID directory path in `file` are used as keys to find the proper experiment to abort.

Under some circumstances, there is a delay between the time `go` is entered and the acquisition is started. During this time, instructions based on the selected pulse sequence are being generated. This is signified by the letters “PSG” appearing in the upper left corner of the status window. A `halt` command issued under these circumstances reports that no acquisition is active but it instead stops the instruction generation process and the message “PSG aborted” appears.

See also: *Getting Started*

Related: **aa** Abort acquisition with error (C)
file File name of parameter set (P)
go Submit experiment to acquisition (C)
wexp Specify action when experiment completes (C)
wexp When experiment completes (P)

hc Automated proton and carbon acquisition (M)

Syntax: `hc<(solvent)>`

Description: Combines the operation of the **h1** and **c13** macros. In non-automation mode, both spectra are acquired in the experiment in which the **hc** macro was entered. After the completion of the acquisition, **rttmp** can be used for further processing of the two spectra.

Arguments: `solvent` is the solvent name. In automation mode, the **enter** program supplies the value. In non-automation mode, the default is 'cdcl3'.

Examples: `hc`
`hc('dms0')`

See also: *Getting Started; User Guide: Liquids NMR*

Related: **c13** Automatic carbon acquisition (M)
enter Enter sample information for automation run (M,U)
h1 Automated proton acquisition (M)
rttmp Retrieve experiment data from experiment subfile (M)

hcapt Automated proton, carbon, and APT acquisition (M)

Syntax: `hcapt<(solvent)>`

Description: Combines the operation of the **h1** and **c13** macros and the APT experiment. In non-automation mode, all spectra are acquired in the experiment in which the **hcapt** macro was entered. After acquisition completes, **rttmp** can be used for further processing of the three spectra.

Arguments: `solvent` is the solvent name. In automation mode, the **enter** program supplies the value. In non-automation mode, the default is 'cdcl3'.

Examples: `hcapt`
`hcapt('dms0')`

See also: *Getting Started; User Guide: Liquids NMR*

Related: **apt** Set up parameters for APT experiment (M)
c13 Automatic carbon acquisition (M)
enter Enter sample information for automation run (M,U)
h1 Automated proton acquisition (M)
rttmp Retrieve experiment data from experiment subfile (M)

hcchtocsy Set up parameters for HCCHTOCSY pulse sequence (M)

Applicability: Sequence is not supplied with *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Syntax: `hcchtocsy`

Description: Used for sidechain assignments in fully ¹³C-enriched molecules.

See also: *User Guide: Liquids NMR*

hccorr Automated proton, carbon, and HETCOR acquisition (M)

Syntax: `hccorr<(solvent)>`

Description: Combines the operation of the `h1` and `c13` macros and the HETCOR experiment. In non-automation mode, all spectra are acquired in the experiment in which `hccorr` is entered. After acquisition completes, `rttmp` can be used for further processing of the three spectra.

Arguments: `solvent` is the solvent name. In automation mode, the `enter` program supplies the value. In non-automation mode, the default is 'cdcl3'.

Examples: `hccorr`
`hccorr('dms')`

See also: *Getting Started; User Guide: Liquids NMR*

Related: `c13` Automated carbon acquisition (M)
`enter` Enter sample information for automation run (M,U)
`h1` Automated proton acquisition (M)
`hetcor` Set up parameters for HETCOR experiment (M)
`rttmp` Retrieve experiment data from experiment subfile (M)

hcdept Automated proton, carbon, and DEPT acquisition (M)

Syntax: `hcdept<(solvent)>`

Description: Combines the operation of the `h1` and `c13` macros and the DEPT experiment. In non-automation mode, all spectra are acquired in the experiment in which `hcdept` was entered. After the completion of the acquisition, `rttmp` can be used for further processing of the three spectra.

Arguments: `solvent` is the solvent name. In automation mode, the `enter` program supplies the value. In non-automation mode, the default is 'cdcl3'.

Examples: `hcdept`
`hcdept('dms')`

See also: *Getting Started; User Guide: Liquids NMR*

Related: `c13` Automatic carbon acquisition (M)
`dept` Set up parameters for DEPT experiment (M)
`enter` Enter sample information for automation run (M,U)
`h1` Automated proton acquisition (M)
`rttmp` Retrieve experiment data from experiment subfile (M)

hcosy Automated proton and COSY acquisition (M)

Syntax: `hcosy<(solvent)>`

Description: Combines the operation of the `h1` macro and the COSY experiment. In non-automation mode, both spectra are acquired in the experiment in which `hcosy` is entered. After acquisition completes, `rttmp` can be used for further processing of the two spectra.

Arguments: `solvent` is the solvent name. In automation mode, the `enter` program supplies the value. In non-automation mode, the default is 'cdcl3'.

Examples: `hcosy`
`hcosy('dms')`

See also: *Getting Started; User Guide: Liquids NMR*

Related: `enter` Enter sample information for automation run (C)
`h1` Automated proton acquisition (M)
`rttmp` Retrieve experiment data from experiment subfile (M)

hdofst **Proton homonuclear decoupler offset (P)**

Applicability: *GEMINI 2000* $^1\text{H}/^{13}\text{C}$ systems.

Description: Configuration parameter that normalizes homonuclear decoupler power output by shifting the range of the parameter `dlp` to compensate for differences in individual decoupler boards. The value of `hdofst` is set using the label Homo Dec. Offset in the CONFIG window (opened from `config`). The installation engineer sets this parameter during system installation and users should not need to change the value.

Values: 0 to 2048. Recommended range is 750 to 1200. The default is 1000. *Lower numbers cause increased power output.*

See also: *VNMR and Solaris Software Installation; Getting Started*

Related: `config` Display current configuration and possibly change it (M)
`dlp` Decoupler low power control (P)
`dm` Decoupler mode (P)
`dmm` Decoupler modulation mode (P)
`homdec` Proton homonuclear decoupler present (P)

hdwshim **Hardware shimming (P)**

Applicability: *UNITY INOVA* systems, and *UNITY* and *VXR-S* systems with additional Z1 shimming hardware.

Description: Allows `go`, `su`, `au`, etc., to turn on and off *UNITY* and *VXR-S* shimming hardware. Hardware shimming is automatically suspended during software autoshimming. On *UNITY INOVA*, hardware shimming is only active during acquisition (`go`, `ga`, `au`). `hdwshim` is a global parameter, so it affects all experiments.

Values: 'y' turns hardware shimming on (only during a delay on *UNITY INOVA*).
'p' turns hardware shimming on during presaturation pulse (power level change followed by pulse). Available on *UNITY INOVA* only.
'n' turns shimming off.

See also: *Getting Started*

Related: `au` Submit experiment to acquisition and process data (C)
`go` Submit experiment to acquisition (C)
`su` Submit a setup experiment to acquisition (M)
`ga` Submit experiment to acquisition and FT the result (M)

hdwshimlist **List of shims for hardware shimming (P)**

Applicability: *UNITY INOVA* systems

Description: A global parameter that sets the shims to use during hardware shimming. If it does not exist, hardware shimming uses `z1` by default. To create the parameter, use `create('hdwshimlist', 'string', 'global')`.

Values: Any string composed of `z1`, `z1c`, `z2`, `z2c`, `x1`, `y1`. Commas and blank space are ignored. Shimming is done in the order `z1`, `z2`, `x1`, `y1`, regardless of the order in the string.

Examples: `hdwshimlist='z1'`
`hdwshimlist='z1z2x1y1'`

See also: *Getting Started*

Related: `create` Create new parameter in a parameter tree (C)
`hdwshim` Hardware shimming (P)

H

help **Display current help file (C)**

Syntax: `help`

Description: Displays help information that explains the functions of the menu buttons and current utility that is active. This information is presented in the text window. The order of search for the `help` information file is first in the user's `vnmr:sys` directory, next in the directory whose path is given by the `helppath` global parameter, and finally in the system `help` directory.

Alternate: Help button in the Permanent Menu.

See also: *VNMR User Programming*

Related: `helppath` User help directory absolute path (P)

helppath **Path to user's help directory (P)**

Description: Contains absolute path to a user's `help` files directory. If `helppath` exists for a user, it must be defined in the user's global parameter file. To create `helppath`, enter `create('helppath', 'string', 'global')`.

See also: *VNMR User Programming*

Related: `help` Display current help file (C)

het2dj **Set up parameters for HET2DJ pulse sequence (M)**

Syntax: `het2dj`

Description: Sets up a HET2DJ (heteronuclear 2D-J) experiment.

Alternate: HET2DJ button in the 2D Pulse Sequence Setup Secondary Menu.

See also: *User Guide: Liquids NMR*

Related: `foldj` Fold J-resolved 2D spectrum about $f1=0$ axis (C)

HETCOR **Change parameters for HETCOR experiment (M)**

Syntax: `HETCOR<('GLIDE')>`

Description: Converts the current parameter set to a HETCOR experiment. This is a phase-sensitive, multiplicity-selected experiment.

Arguments: 'GLIDE' is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding carbon spectrum for the experiment.

Related: `hetcor` Set up parameters for HETCOR experiment (M)

hetcor **Set up parameters for HETCOR pulse sequence (M)**

Syntax: `hetcor<(exp_number)>`

Description: Sets up a HETCOR (heteronuclear chemical shift correlation) experiment.

Arguments: `exp_number` is the number of the experiment, from 1 to 9, in which a proton spectrum of the sample already exists.

Alternate: HETCOR button in the 2D Pulse Sequence Setup Secondary Menu.

See also: *User Guide: Liquids NMR*

Related: `plhxcor` Plot X,H-correlation 2D spectrum (M)
`ppcal` Proton decoupler pulse calibration (M)

- hetcorcp1** **Set up parameters for solids HETCOR pulse sequence (M)**
- Applicability: Systems with the solids module.
- Syntax: `hetcorcp1`
- Description: Sets up a parameter set, obtained with XPOLAR or XPOLAR1, for HETCORCP1, the solid-state heteronuclear correlation experiment.
- See also: *User Guide: Solid-State NMR*
- Related: `xpolar` Set up parameters for XPOLAR pulse sequence (M)
`xpolar1` Set up parameters for XPOLAR1 pulse sequence (M)
-
- hetcorps** **Set up parameters for HETCORPS pulse sequence (M)**
- Applicability: Not supplied with *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems.
- Syntax: `hetcorps`
- Description: Sets up parameters for a heteronuclear chemical shift correlation experiment (absolute value and phase sensitive).
- See also: *User Guide: Liquids NMR*
-
- hidecommand** **Execute macro instead of command with same name (C)**
- Syntax: (1) `hidecommand(command_name) < : $new_name >`
(2) `hidecommand('?')`
- Description: Renames (or hides) a built-in VNMR command so that a macro with the same name as the built-in command is executed instead of the built-in command.
- Arguments: `command_name` is the name of the command to be renamed. To reset the built-in command back to its original name, enter `hidecommand` with the hidden name as the argument.
- `$new_name` returns the new name of the built-in command. By using this new name, access is still available to the built-in command.
- `' ? '` is a keyword to display a list of all of the renamed built-in commands and their original names.
- Examples: `hidecommand('sys') : $newname`
`hidecommand('Sys')`
`hidecommand('?')`
- See also: *System Administration; VNMR User Programming*
- Related: `exists` Set up parameters for XPOLAR pulse sequence (M)
`which` Display which macro or command is used (M)
-
- HMBC** **Change parameters for HMBC experiment (M)**
- Syntax: `HMBC< ('GLIDE') >`
- Description: Converts the current parameter set to a HMBC experiment.
- Arguments: `'GLIDE'` is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding proton spectrum for the experiment.
- Related: `gHMBC` Change parameters for gHMBC experiment (M)

- hmqc** **Set up parameters for HMQC pulse sequence (M)**
- Applicability: All systems, except that presaturation or homospoil are not available on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems. Homospoil is not available on *GEMINI 2000* systems.
- Syntax: `hmqc<(isotope)>`
- Description: Sets up a HMQC heteronuclear multiple-quantum coherence) experiment.
- Arguments: `isotope` is the isotope number for the heteronucleus of interest (e.g., 13 for ¹³C).
- See also: *User Guide: Liquids NMR*
-
- HMQC** **Set up parameters for HMQC experiment (M)**
- Syntax: `HMQC<(' GLIDE ')>`
- Description: Converts the current parameter set to a ¹³C HMQC experiment.
- Arguments: ' GLIDE ' is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding carbon spectrum for the experiment.
- Related: [gHMQC](#) Set up parameters for gHMQC experiment (M)
[HMQC15](#) Set up parameters for ¹⁵N HMQC experiment (M)
[HMQC_d2](#) Set up parameters for ¹⁵N HMQC using decoupler 2 (M)
[HMQC_d213](#) Set up parameters for ¹³C HMQC using decoupler 2 (M)
-
- HMQC15** **Set up parameters for ¹⁵N HMQC experiment (M)**
- Syntax: `HMQC15<(' GLIDE ')>`
- Description: Converts the current parameter set to a HMQC experiment for ¹⁵N.
- Arguments: ' GLIDE ' is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding carbon spectrum for the experiment.
- Related: [HMQC](#) Set up parameters for HMQC experiment (M)
-
- HMQC_d2** **Set up parameters for ¹⁵N HMQC experiment using decoupler 2 (M)**
- Syntax: `HMQC_d2<(' GLIDE ')>`
- Description: Converts the current parameter set to a HMQC experiment for ¹⁵N with decoupler 2 as ¹⁵N.
- Arguments: ' GLIDE ' is a keyword to first retrieve the PROTON parameter set for the particular sample.
- Related: [HMQC](#) Set up parameters for HMQC experiment (M)
-
- HMQC_d213** **Set up parameters for ¹³C HMQC experiment using decoupler 2 (M)**
- Syntax: `HMQC_d213<(' GLIDE ')>`
- Description: Converts the current parameter set to a HMQC experiment for ¹³C with decoupler 2 as ¹³C.
- Arguments: ' GLIDE ' is a keyword to first retrieve the PROTON parameter set for the particular sample.
- Related: [HMQC](#) Set up parameters for HMQC experiment (M)

- hmqcr** **Set up parameters for HMQCR pulse sequence (M)**
 Applicability: Not needed in current systems. Normally was used in systems with a ^1H only decoupler.
 Syntax: hmqcr
 Description: Sets up a HMQC (heteronuclear multiple-quantum coherence) experiment with “reverse” configuration.
 See also: *User Guide: Liquids NMR*
- hmqctocsy** **Set up parameters for HMQCTOCSY pulse sequence (M)**
 Applicability: Sequence is not supplied with *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.
 Syntax: hmqctocsy
 Description: Sets up a HMQCTOCSY experiment with an option to null or invert the direct responses.
 See also: *User Guide: Liquids NMR*
- HMQCTOXY** **Set up parameters for HMQCTOXY experiment (M)**
 Syntax: HMQCTOXY<(' GLIDE ')>
 Description: Converts the current parameter set to a ^{13}C HMQCTOXY experiment.
 Arguments: ' GLIDE ' is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding proton spectrum for the experiment.
 Related: [gHMQC](#) Set up parameters for gHMQC experiment (M)
 [HMQC](#) Set up parameters for HMQC experiment (M)
 [HMQCTOXY15](#) Set up parameters for ^{15}N HMQCTOXY experiment (M)
 [HMQCTOXY_d2](#) Set up parameters for ^{15}N HMQCTOXY using decoupler 2 (M)
 [HMQCTOXY_d213](#) Set up parameters for ^{13}C HMQCTOXY using decoupler 2 (M)
- HMQCTOXY15** **Set up parameters for ^{15}N HMQCTOXY experiment (M)**
 Syntax: HMQCTOXY15<(' GLIDE ')>
 Description: Converts the current parameter set to a HMQCTOXY experiment for ^{15}N .
 Arguments: ' GLIDE ' is a keyword to first retrieve the PROTON parameter set for that particular sample.
 Related: [HMQCTOXY](#) Set up parameters for HMQCTOXY experiment (M)
- HMQCTOXY_d2** **Set up parameters for ^{15}N HMQCTOXY using decoupler 2 (M)**
 Syntax: HMQCTOXY_d2<(' GLIDE ')>
 Description: Converts the current parameter set to a HMQCTOXY experiment for ^{15}N with decoupler 2 as ^{15}N .
 Arguments: ' GLIDE ' is a keyword to first retrieve the PROTON parameter set for that particular sample.
 Related: [HMQCTOXY](#) Set up parameters for HMQCTOXY experiment (M)
- HMQCTOXY_d213** **Set up parameters for ^{13}C HMQCTOXY using decoupler 2 (M)**
 Syntax: HMQCTOXY+d213<(' GLIDE ')>
 Description: Converts the current parameter set to a HMQCTOXY experiment for ^{13}C with decoupler 2 as ^{13}C .

Arguments: 'GLIDE' is a keyword to first retrieve the PROTON parameter set for that particular sample.

Related: **HMQCTOXY** Set up parameters for HMQCTOXY experiment (M)

hmqcto3d Set up parameters for HMQC-TOCSY 3D pulse sequence (M)

Applicability: Not supplied with *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems.

Syntax: hmqcto3d

Description: Sets up parameters for a HMQC-TOCSY 3D experiment with a presaturation option.

See also: *User Guide: Liquids NMR*

ho Horizontal offset (P)

Description: Horizontal offset of the each spectrum in a “stacked display” with respect to the previous spectrum,. For 1D data sets, the parameter **vo** sets the vertical offset. For 2D data sets, the parameter **wc2** sets the vertical distance (in mm) between the first and last traces.

Values: Number, in mm, for offset size. For a “left-to-right” presentation, **ho** is typically negative; for “bottom-to-top” presentation, **vo** or **wc2** is positive.

See also: *Getting Started*

Related: **vo** Vertical offset (P)
wc2 Width of chart in second direction (P)

hold Post-trigger delay (P)

Applicability: Systems with imaging capabilities.

Description: Specifies a hold time between an external trigger and the start of the actual pulse sequence events. For example, in cardiac triggered imaging, **hold** provides a mechanism for offsetting the start of the sequence by a variable amount to obtain images at different times in the cardiac cycle.

See also: *User Guide: Imaging*

Related: **ticks** Number of trigger pulses (P)

hom2dj Set up parameters for HOM2DJ pulse sequence (M)

Syntax: hom2dj

Description: Sets up a HOM2DJ (homonuclear J-resolved 2D) experiment.

Alternate: HOM2DJ button in the 2D Pulse Sequence Setup Secondary Menu.

See also: *User Guide: Liquids NMR*

HOMODEC Change parameters for HOMODEC experiment (M)

Syntax: HOMODEC('GLIDE')

Description: Converts the current parameter set to a HOMODEC experiment. A 1D proton spectrum is displayed with **ds_selfrq** menu to do peak selection.

Arguments: 'GLIDE' is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding proton spectrum for the experiment.

Related: **NOESY1D** Change parameters for NOESY1D experiment (M)
TOCSY1D Change parameters for TOCSY1D experiment (M)

homdec Proton homonuclear decoupler present (P)

Applicability: All *MERCURY* systems and *GEMINI 2000* $^1\text{H}/^{13}\text{C}$ systems. On *GEMINI 2000* and *MERCURY-VX* broadband systems, homonuclear decoupling is standard and this parameter must be set to 'y'.

Description: Sets whether the optional proton homonuclear decoupler board is present. The value is set using the Homodecoupler label in the CONFIG window (opened from `config`).

Values: 'n' indicates the board is not on the system (Not Present choice from the CONFIG window). If `homdec` = 'n', no communication with the board is possible: if the board is on, it will stay on, and if it is off, it will stay off.

'y' indicates the board is in the system (Present choice from the CONFIG window). This is the default. `homdec` must be set to 'y' if the homonuclear decoupler board is present, even to turn off the homodecoupler.

See also: *VNMR and Solaris Software Installation; Getting Started*

Related: `config` Display current configuration and possibly change it (M)
`dlp` Decoupler low power control (P)
`hdofst` Proton homonuclear decoupler offset (P)

homo Homodecoupling control for first decoupler (P)

Applicability: All systems except *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*; however, *MERCURY* and *GEMINI 2000* automatically use gated decoupling when `tn` and `dn` are both in the high band (i.e., if `tn` and `dn` is ^1H or ^{19}F , and `dm` = 'y').

Description: Enables time-shared decoupling. Unlike the `dm`, `dmm`, and `hs` parameters, `homo` is not under "status" control. On systems with type 2 or 3 interface board (`apinterface`=2 or `apinterface`=3), `homo` does not control any signal routing; the position of the relevant relays is controlled by whether homonuclear decoupling (`tn` equals `dn`) or heteronuclear decoupling (`tn` not equal to `dn`) is in effect.

Values: On *UNITYINOVA* and *UNITYplus*, the values are 'n' or 'y', where:

- 'n' specifies no gating.
- 'y' specifies that the receiver is gated, which is done by controlling the observe L.O. (local oscillator) line. If `dm` = 'y', first decoupler rf, amplifier (blanked/unblanked), and preamplifier are gated. If `dm` = 'n', no gating of these signals takes place. When `homo` is set to 'y', `dmm` should be set to 'c' for continuous wave (CW) modulation.

On *UNITY* and *VXR-S*, the values are 'n' or 'y', where:

- 'n' disables decoupler time-sharing, which is appropriate for heteronuclear decoupling or for cases in which the decoupler is off during acquisition.
- 'y' selects time-shared decoupling, which is appropriate for homonuclear decoupling in which the receiver is gated off when the decoupler is on. On systems with the type 1 interface board (`apinterface`=1), `homo` = 'y' also causes the decoupler signal to be combined with the observe signal before being sent to the probe.

See also: *Getting Started*

Related: `apinterface` AP Interface board type (P)
`dn` Nucleus for first decoupler (P)
`homo2` Homodecoupling control for second decoupler (P)

homo3 Homodecoupling control for third decoupler (P)
tn Nucleus for observe transmitter (P)

homo2 Homodecoupling control for second decoupler (P)

Applicability: Systems with a second decoupler.

Description: Equivalent to the parameter **homo**. It works in conjunction with the parameters **dm2** and **dmm2**.

Values: 'n', 'y'

See also: *Getting Started*

Related: **dm2** Decoupler mode for second decoupler (P)
dmm2 Decoupler modulation mode for second decoupler (P)
dn2 Nucleus for second decoupler (P)
homo Homodecoupling control for first decoupler (P)

homo3 Homodecoupling control for third decoupler (P)

Applicability: Systems with a third decoupler.

Description: Equivalent to the parameter **homo**. It works in conjunction with the parameters **dm3** and **dmm3**.

Values: 'n', 'y'

See also: *Getting Started*

Related: **dm3** Decoupler mode for third decoupler (P)
dmm3 Decoupler modulation mode for third decoupler (P)
dn3 Nucleus for third decoupler (P)
homo Homodecoupling control for first decoupler (P)

homo4 Homodecoupling control for fourth decoupler (P)

Applicability: Systems with a deuterium decoupler channel as the fourth decoupler.

Description: Equivalent to the parameter **homo**. It works in conjunction with the parameters **dm4** and **dmm4**.

Values: 'n', 'y'

See also: *Getting Started*

Related: **dm4** Decoupler mode for fourth decoupler (P)
dmm4 Decoupler modulation mode for fourth decoupler (P)
dn4 Nucleus for fourth decoupler (P)
homo Homodecoupling control for first decoupler (P)

hoult Set parameters alfa and rof2 according to Hoult (M)

Syntax: `hoult`

Description: Sets the values of **alfa** and **rof2** according to a prescription advanced by D. I. Hoult (*J. Magn. Reson.* **51**, 110 (1983)). These parameters set the times that follow the final pulse, which can be important where the flatness of the baseline is of concern.

See also: *Getting Started*

Related: **alfa** Set **alfa** delay before acquisition (P)
calfa Recalculate **alfa** so that first-order phase is zero (M)
rof2 Receiver gating time following pulse (P)

- hpa** **Plot parameters on special preprinted chart paper (C)**
- Syntax: `hpa`
- Description: Plots a predetermined list of parameters by “filling in the blanks” at the bottom of the preprinted chart paper available for Hewlett-Packard 7475- and 7550-series plotters.
- Alternate: HP Params button in the 1D Plotting Menu (the plotter must be installed as a Hewlett-Packard system for the HP Params button to appear).
- See also: *Getting Started*
- Related: `apa` Plot parameters automatically (M)
`x0` X-zero position of HP plotter or Postscript device (P)
`y0` Y-zero position of HP plotter or Postscript device (P)
-
- hregions** **Select integral regions in proton spectrum (M)**
- Syntax: `hregions`
- Description: Selects integral regions, a critical step in automatic processing of proton spectra. It is critical not only because of aesthetic reasons (some people like many small integrals, others prefer a few large regions), but also because other commands, such as `bc`, depend on the correct integration: `bc` can either fail or it can make broad, unintegrated lines disappear from the spectrum. `hregions` was specifically designed for proton spectra and should not be used for other types of spectra. The result of `hregions` also depends on the lineshape and the signal-to-noise ratio of a spectrum
- See also: *Getting Started; User Guide: Liquids NMR*
- Related: `bc` 1D and 2D baseline correction (C)
`integrate` Automatically integrate 1D spectrum (M)
-
- hs** **Homospoil pulses (P)**
- Applicability: All systems except *GEMINI 2000* (homospoil is not available on *GEMINI 2000* and, therefore, `hs` should be set to 'nn').
- Description: Turns on homospoil pulses at various times in different pulse sequences. Homospoil is a process by which the homogeneity is temporarily made very bad (“spoiled”) to cause any transverse magnetizations present at that time to decay rapidly to zero. `hst` controls the length of any homospoil pulse.
- Values: In a standard two-pulse sequence, homospoil pulses can be inserted during periods A and B (delays `d1` and `d2`): `hs= 'yn'` gives a homospoil pulse at the beginning of `d1`, `hs= 'ny'` gives a pulse during `d2`, and `hs= 'yy'` gives homospoil pulses during both `d1` and `d2`. The desired value is generally `hs= 'nn'`.
- See also: *Getting Started*
- Related: `d1` First delay (P)
`d2` Incremented delay in 1st indirectly detected dimension (P)
`hst` Homospoil time (P)
-
- hsqc** **Set up parameters for HSQC pulse sequence (M)**
- Applicability: Not supplied with *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems.
- Syntax: `hsqc`
- Description: Sets up parameters for a heteronuclear Overbodenhausen experiment using reverse INEPT.

See also: *User Guide: Liquids NMR*

- HSQC** **Set up parameters for HSQC experiment (M)**
- Syntax: HSQC<('GLIDE')>
- Description: Converts the current parameter set to a ^{13}C HSQC experiment.
- Arguments: 'GLIDE' is a keyword to first retrieve the PROTON parameter set for the experiment.
- Related: [gHSQC](#) Change parameters for gHSQC experiment (M)
[HSQC15](#) Change parameters for ^{15}N HSQC experiment (M)
[HSQC_d2](#) Change parameters for ^{15}N HSQC with decoupler 2 (M)
[HSQC_d213](#) Change parameters for ^{13}C HSQC with decoupler 2 (M)
- HSQC15** **Set up parameters for ^{15}N HSQC experiment (M)**
- Syntax: HSQC15<('GLIDE')>
- Description: Converts the current parameter set to a HSQC experiment for ^{15}N .
- Arguments: 'GLIDE' is a keyword to first retrieve the PROTON parameter set for the experiment.
- Related: [HSQC](#) Change parameters for HSQC experiment (M)
- HSQC_d2** **Set up parameters for ^{15}N HSQC experiment using decoupler 2 (M)**
- Syntax: HSQC_d2<('GLIDE')>
- Description: Converts the current parameter set to a HSQC experiment for ^{15}N with decoupler 2 as ^{15}N .
- Arguments: 'GLIDE' is a keyword to first retrieve the PROTON parameter set for the experiment.
- Related: [HSQC](#) Change parameters for HSQC experiment (M)
- HSQC_d213** **Set up parameters for ^{13}C HSQC experiment using decoupler 2 (M)**
- Syntax: HSQC_d213<('GLIDE')>
- Description: Converts the current parameter set to a HSQC experiment for ^{13}C with decoupler 2 as ^{13}C .
- Arguments: 'GLIDE' is a keyword to first retrieve the PROTON parameter set for the experiment.
- Related: [HSQC](#) Change parameters for HSQC experiment (M)
- HSQCTOXY** **Set up parameters for HSQCTOXY experiment (M)**
- Syntax: HSQCTOXY<('GLIDE')>
- Description: Converts the current parameter set to a ^{13}C HSQCTOXY experiment.
- Arguments: 'GLIDE' is a keyword to first retrieve the PROTON parameter set for the experiment.
- Related: [gHSQCTOXY](#) Change parameters for gHSQCTOXY experiment (M)
[HSQCTOXY15](#) Change parameters for ^{15}N HSQCTOXY experiment (M)
[HSQCTOXY_d2](#) Change parameters for ^{15}N HSQCTOXY with decoupler 2 (M)
[HSQCTOXY_d213](#) Change parameters for ^{13}C HSQCTOXY with decoupler 2 (M)

HSQCTOXY15 Set up parameters for ¹⁵N HSQCTOXY experiment (M)

Syntax: HSQCTOXY15<('GLIDE')>

Description: Converts the current parameter set to a HSQCTOXY experiment for ¹⁵N.

Arguments: 'GLIDE' is a keyword to first retrieve the PROTON parameter set for the experiment.

Related: [HSQCTOXY](#) Change parameters for HSQCTOXY experiment (M)

HSQCTOXY_d2 Set up parameters for ¹⁵N HSQCTOXY using decoupler 2 (M)

Syntax: HSQCTOXY_d2<('GLIDE')>

Description: Converts the current parameter set to a HSQCTOXY experiment for ¹⁵N with decoupler 2 as ¹⁵N.

Arguments: 'GLIDE' is a keyword to first retrieve the PROTON parameter set for the experiment.

Related: [HSQCTOXY](#) Change parameters for HSQCTOXY experiment (M)

HSQCTOXY_d213 Set up parameters for ¹³C HSQCTOXY using decoupler 2 (M)

Syntax: HSQCTOXY_d213<('GLIDE')>

Description: Converts the current parameter set to a HSQCTOXY experiment for ¹³C with decoupler 2 as ¹³C.

Arguments: 'GLIDE' is a keyword to first retrieve the PROTON parameter set for the experiment.

Related: [HSQCTOXY](#) Change parameters for HSQCTOXY experiment (M)

hsqctoxyzSE Set up parameters for HSQC-TOCSY 3D pulse sequence (M)

Applicability: Not supplied with *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems.

Syntax: hsqctoxyzSE

Description: Sets up parameters for a HSQC -TOCSY 3D experiment.

See also: *User Guide: Liquids NMR*

hsrotor Display rotor speed for solids operation (P)

Applicability: Systems equipped with the rotor synchronization module.

Description: Controls display of rotor speed. Depending on whether the rotor synchronization module is present (set by the Rotor Synchronization label in the CONFIG window opened from [config](#)), parameter [rotorsync](#) is set to 1 or 0. The [xpolar](#) macro in turn uses this to create [hsrotor](#), which is set to 'y' if rotor synchronization is present. If the parameter [srate](#) exists, it is updated to the spin speed of the rotor at the end of the experiment. The interlock function specified by parameter [in](#) also changes. If [hsrotor](#)= 'y' and [in](#)= 'y', the experiment is terminated if the rotor speed deviates more than 100 Hz.

Values: 'n' makes [srate](#) unmodified by acquisition and turns off the rotor speed display in [Acqstat](#).

'y' makes the hardware information from the rotor synchronization board update [srate](#) and displays the rotor speed in the [Acqstat](#) status display.

See also: *User Guide: Solid-State NMR*

Related: **Acqstat** Bring up the acquisition status display (U)
config Display current configuration and possibly change it (M)
in Interlock (P)
rotorsync Rotor synchronization (P)
srate Spinning speed (P)
xpolar Set up parameters for XPOLAR pulse sequence (M)

hst Homospoil time (P)

Applicability: All systems except *GEMINI 2000*.

Description: Controls pulse length if homospoil is activated by the **hs** parameter.

Values: On *UNITYINOVA* and *UNITYplus*, 0 to 20 ms (limited by hardware).

On *MERCURY-VX* and *MERCURY*, 0 to 20 ms (limited by software, 8 ms is standard).

On *UNITY* and *VXR-S*, 0 to 20 ms (limited by software, 8 ms is standard).

See also: *Getting Started*

Related: **hs** Homospoil pulses (P)

htune Tune proton channel on GEMINI 2000 (M)

Applicability: *GEMINI 2000* $^1\text{H}/^{13}\text{C}$ systems.

Syntax: **htune**

Description: Turns on the ^1H transmitter, directing about 0.5 watts of rf to the probe coil. Before entering **htune**, be sure to move the proper cable on the back of the left-hand magnet leg to the BNC connector labeled TUNE, and also to move the proper cable leading to the probe to the BNC connector labeled TUNE. Enter **tuneoff** to turn off the transmitter. **htune** cannot be executed while the console is acquiring or interactive acquisition (**acqi**) is connected. For the full tuning procedure, see the manual *Autoswitchable NMR Probes Installation*.

See also: *Getting Started; Autoswitchable NMR Probes Installation*

Related: **acqi** Interactive acquisition display process (C)
btune Tune broadband channel on *MERCURY-Series*, *GEMINI 2000* (M)
ctune Tune carbon channel on $^1\text{H}/^{13}\text{C}$ *GEMINI 2000* (M)
dtune Tune lock channel on *GEMINI 2000* (M)
sethw Set values for hardware in acquisition system (C)
su Submit a setup experiment to acquisition (M)
tuneoff Turn off probe tuning mode, *MERCURY-Series GEMINI 2000* (M)

hzmm Scaling factor for plots (P)

Description: Contains the quotient of **w_p** divided by **w_c**, a scaling factor useful for plotting. **hzmm** applies to 1D only.

See also: *Getting Started*

Related: **wc** Width of chart (P)
w_p Width of plot (P)

hztomm Convert locations from Hz or ppm to plotter units (C)

Syntax: (1) **hztomm(x_{position})<:xmm>**
(2) **hztomm(x_{position},y_{position})<:xmm,ymm>**

```
(3) hztomm(<'box', ><'plotter' | 'graphics', >x_left,
          x_right, y_bottom, y_top) <:x1mm, x2mm, y1mm, y2mm>
```

Description: Converts locations from Hz, or ppm, to plotter units.

Arguments: `x_position` in syntax 1 is a location along the 1D axis, in Hz or ppm, to be converted to plotter units using the current values of parameters `sp` and `wp`. Plotter units are mm on most plots and are scaled for graphics display. For ppm entries, use the `p` suffix following numerical values (see first example below).

`x_position, y_position` in syntax 2 is a coordinate, in Hz or ppm, on a 2D plot to be converted to plotter units, using the parameters `sp` and `wp` to convert the horizontal position and the parameters `sp1` and `wp1` to convert the vertical position.

`x_left, x_right, y_bottom, y_top` in syntax 3 are box edges, in Hz or ppm, on a 2D plot to be converted to plotter units, using the parameters `sp` and `wp` to convert the left and right edges, and parameters `sp1` and `wp1` to convert the top and bottom edges.

'`box`' is a keyword to draw a box and to make the first two return arguments, if supplied, give the location of the upper left corner of the box, in plotter units.

'`plotter`' is a keyword to select the plotter. The default is '`graphics`'.

'`graphics`' is a keyword to select the graphics screen. This is the default.

`x1mm, x2mm, y1mm, y2mm` are return arguments giving values in plotter units. If return arguments are not supplied, the results are displayed instead.

Examples: `hztomm(20p)`
`hztomm(xpos, ypos) : xmm, ymm`
`hztomm('box', 'plotter', 20, 50, 10, 30)`

See also: *Getting Started*

Related:	<code>box</code>	Draw a box on a plotter or graphics display (C)
	<code>sp</code>	Start of plot in directly detected dimension (P)
	<code>sp1</code>	Start of plot in 1st indirectly detected dimension (P)
	<code>wp</code>	Width of plot in directly detected dimension (P)
	<code>wp1</code>	Width of plot in 1st indirectly detected dimension (P)

i **Insert sample (M)**

Applicability: All systems (including *MERCURY* and *GEMINI 2000*) if spin control hardware is installed.

Syntax: `i`

Description: Turns off the eject air, waits for sample to slowly drop, and then turns off the slow drop air. The macro `insert` functions the same as `i`.

See also: *Getting Started*

Related: `e` Eject sample (M)
`eject` Eject sample (M)
`insert` Insert sample (M)

ihwinfo **Hardware status of ^{UNITY}INOVA console (U)**

Applicability: ^{UNITY}INOVA consoles (not available for any other type of console).

Syntax: (From UNIX) `ihwinfo('startup' | 'abort')`

Description: Displays status of digital hardware in the ^{UNITY}INOVA console. The output is intended for service personnel and probably not meaningful to users.

Arguments: `'startup'` is a keyword to display the status at the conclusion of the last console startup (powerup, reboot, etc.).

`'abort'` is a keyword to display the status the last time an acquisition was aborted or the console rebooted from the host computer (`abortallacqs`). In this context, exiting from either the FID display or lock display of `acqi` counts as an abort. Only the status from the last abort can be displayed.

Examples: `ihwinfo('startup')`
`ihwinfo('abort')`

See also: *Getting Started*

Related: `abortallacqs` Reset acquisition computer in a drastic situation (C)
`showconsole` Show ^{UNITY}INOVA console configuration parameters (U)

i1 **Interleave arrayed and 2D experiments (P)**

Description: Controls experimental interleaving in arrayed experiments. When interleaving is active, `bs` transients are performed for each member of the array, followed by `bs` more transients for each member of the array, and so on, until `nt` transients have been collected for each member of the array. Thus, `i1` is only relevant if `bs` is less than `nt`.

Values: `'y'` turns on interleaving and `'n'` turns off interleaving.

See also: *User Guide: Liquids NMR*

Related: `bs` Block size (P)
`nt` Number of transients (P)

ilfid **Interleave FIDs during data processing (C)**

Syntax: `ilfid`

Description: Converts a multiple FID element into a single FID. It is possible to effectively extend the Nyquist frequency (i.e., increase the effective spectral width **sw**) by acquiring a number of FIDs with different *tau2* values and then reprocessing the data. **ilfid** does the necessary processing of time-domain data to achieve this extension, assuming that a pulse sequence (not supplied) has been written to generate the required data.

When invoked in an experiment of **nf** FIDs, each of **np** points, **ilfid** sorts the data into a single FID of **np*nf** points that can then be transformed. The interleaving takes the first complex point of each of the **nf** FIDs and places them in sequential order in the new FID. It then takes the second complex point from each of the **nf** FIDs and appends them sequentially to the new FID. This operation is repeated for all complex points. Although **ilfid** adjusts **np** and **nf**, it does not alter other parameters such as **sw**.

CAUTION: Because **ilfid** alters the data irrevocably, it is strongly recommended that you save the FID before using **ilfid**.

Examples: Illustrated below is the interleaving of an FID with **nf**=3 and **np**=4. Each point is represented by two digits. The first digit is the **nf** number and the second digit is the sequential point for that **nf** value. Data before the **ilfid** command:

11, 12, 13, 14; 21, 22, 23, 24; 31, 32, 33, 34

Data after the **ilfid** command:

11, 21, 31, 12, 22, 32, 13, 23, 33, 14, 24, 34

See also: *Getting Started*

Related: **nf** Number of FIDs (P)
np Number of data points (P)
sw Spectral width in directly detected dimension (P)

image Display noninteractive gray scale image (M)

Applicability: Systems with imaging capabilities.

Syntax: **image**

Description: Brings up a **dcon** 2D display of an image (using grayscale and linear scaling of the intensity) that can be used for adjusting the display while using **dconi**.

See also: *User Guide: Imaging*

Related: **dcon** Display noninteractive color intensity map (C)
dconi Interactive 2D data display (C)
dconn Display color intensity map without erasing screen (C)

image Control phase encoding gradient in EPI experiments (P)

Applicability: Systems with echo planar imaging (EPI) capabilities.

Description: Turns on and off the phase encoding gradient in EPI experiments. **image** also specifies the number of EPI images to collect in an arrayed experiment.

Values: 0 specifies that the phase encoding gradient is turned off.

1 specifies that the phase encoding gradient is turned on.

Examples: **image=0, 1, 1, 1** collects a set of four EPI images. The first dataset refers to the reference scan.

See also: *User Guide: Imaging*

imageprint **Plot noninteractive gray scale image (M)**

Syntax: `imageprint`

Description: Sends to the plotter a **dcon** color intensity map with linear instead of logarithmic increments and with grayscale instead of colors.

Alternate: Image button on the 2D Plotting Menu.

See also: *User Guide: Liquids NMR*

Related: **dcon** Display noninteractive color intensity map (C)
image Display noninteractive gray scale image (M)

imark **Annotate an image display (M)**

Applicability: Systems with imaging capabilities.

Syntax: `imark(string<,color>)`

Description: Used to label an image display with characters or strings in any color provided by the **write** command. The labeling is only available inside the axis box of the image and is directed by the 2D cursors.

Arguments: `string` is a text string.

`color` is color of the text on a color display: 'red', 'yellow', 'green', 'cyan', 'blue', 'magenta', and 'white'. The default is 'yellow'.

Examples: `imark('Muscle','red')`

See also: *User Guide: Imaging*

Related: **write** Write formatted text to a device (C)

imcalc **Calculate 2D phasefiles (M,U)**

Applicability: Systems with imaging capabilities.

Syntax: (From VNMR) `imcalc(optype,phf1,<phf2,outphf,args>)`
(From UNIX) `imcalc optype phf1 <phf2 outphf args>`

Description: Provides a means, along with the supporting macros, of performing arithmetic operations at a pixel-by-pixel basis on images. As operands, phasefiles are required that have been previously saved with the VNMR command **svphf**. A new phasefile is generated that represents the result of the selected action.

The UNIX program `imcalc` may be called from a UNIX shell using syntax 1, or called from VNMR with the macro `imcalc` using syntax 2. The macro **imcalci** serves as an interactive interface to the `imcalc` macro by prompting for any required inputs, which vary with the operation type. For unary operations, such as `log`, **imcalci** uses the phasefile resident in the current experiment by default

Arguments: `optype` can be any of the following keywords (place single quotes around the keyword when entering `imcalc` from VNMR):

- `abs` takes the absolute value of an image.
- `add` adds two images.
- `addc` adds a constant value to each pixel in an image.
- `clipmax` sets pixel values above a user-supplied maximum to zero.
- `clipmin` sets pixel values below a user-supplied minimum to zero.
- `div` divides the first image by the second.
- `exp` sets the antilog of an image: (10^{image}).

- `f1roll` wraps an image in the f_1 direction a selected number of pixels.
- `f2roll` wraps an image in the f_2 direction a selected number of pixels.
- `flip_diag` flips an image about $x=y$ diagonal (square images only).
- `flip_horiz` flips an image about the central horizontal axis.
- `flip_vert` flips an image about the central vertical axis.
- `gmean` sets the geometric mean of two images: $\sqrt{\text{image1} \times \text{image2}}$.
- `hline` replaces a selected horizontal trace by the average of the two adjacent traces.
- `log` sets a logarithm of an image: $\log|\text{image}|$.
- `mean` sets the arithmetic mean of two images: $\frac{\text{image1} + \text{image2}}{2}$.
- `mult` multiplies two images.
- `multc` multiplies each pixel in an image by a constant value.
- `phase` computes a resultant image from the phase angle determined by the arctangent of two orthogonal component images.
- `pow` sets exponentiation of an image ($\text{image}^{\text{constant}}$). To invert an image (1/pixel), use `pow` with an exponent of -1 . To get a square root image, use `pow` with an exponent of $1/2$.
- `reverse` sets linear inversion of pixel intensities in an image.
- `rotate_90` rotates an image clockwise 90° (square images only).
- `rotate_180` rotates an image 180° .
- `sub` subtracts the second image from the first (use `add` with a negative multiplier in direct call to UNIX `imcalc` program)
- `thresh` compresses all pixel values above a selected threshold to 1, and below to 0.
- `thresh2` compresses all pixel values above a user-supplied minimum and below a user-supplied maximum to 1, all others to 0.
- `vadd` adds two orthogonal “component” images to form the vector sum: $\sqrt{\text{image1}^2 + \text{image2}^2}$.
- `vline` replaces a selected vertical trace by the average of the two adjacent traces.

Examples: (From UNIX) `imcalc add phf1 phf2 outphf 0.5`

(From VNMR) `imcalc('add', 'phf1', 'phf2', 'destphf' 0.5)`

See also: *User Guide: Imaging*

Related:	<code>add</code>	Add current FID to add/subtract experiment (C)
	<code>makephf</code>	Transform and save images as phasefiles (M)
	<code>spadd</code>	Add current spectrum to add/subtract experiment (C)
	<code>svphf</code>	Save phasefiles (C)

imcalci **Format arguments for imcalc macro (M)**

Applicability: Systems with imaging capabilities.

Syntax: `imcalci(optype)`

Description: Interactively formats arguments for the `imcalc` macro from prompted user inputs. The macro `imcalci` can be run from the VNMR command line or be accessed through the menu system by selecting the ImageCalc option in the Analyze menu.

Arguments: `optype` has the same values as `optype` for the `imcalc` macro.

Examples: `imcalci('add')`

See also: *User Guide: Imaging*

Related: `imcalc` Calculate 2D phasefiles (M,U)

imconi Display 2D data in interactive grayscale mode (M)

Syntax: `imconi`

Description: Calls the `dconi` program with the arguments required for grayscale image display: `dconi('dcon','gray','linear')`.

Related: `dconi` Interactive 2D contour display (C)

imfit Fit arrayed imaging data to T_1 or T_2 exponential data (M,U)

Applicability: Systems with imaging capabilities.

Syntax: (From VNMR) `imfit('t1'|'t2',basename,min_threshold)`
(From UNIX) `imfit t1|t2 basename min_threshold`
`time1 time2 ... timeN`

Description: Performs fitting at each pixel to exponential T_1 or T_2 data. The `imfit` macro from VNMR provides a convenient link to the UNIX `imfit` fitting procedure by setting up and passing the correct arguments to the external program. If data cannot be handled by the VNMR macro, the UNIX `imfit` command can be called directly.

Three synthetic images are created by the `imfit` program, and placed in the `planes` directory of the current experiment. The T_1 or T_2 image are named `basenamet1` or `basenamet2`. An error image `basenamesigma` represents the standard deviation of the fit at each pixel, and a $t=0$ image, `basenamem0`, represent the intercept of the original data at zero time.

The `imfit` macro automatically extracts the timing values for each array element in the data set from whichever parameter has been arrayed, providing these times to the fitting routine. For this reason, the `imfit` macro does not function properly if more than one parameter is arrayed.

Two macros, `t1image` and `t2image`, are provided to do all of the preprocessing required for fitting. They query for the base phasefile names and lower-limit noise threshold, transform and save all of the images, and call the `imfit` macro to complete the fitting process.

T_1 fitting type requires phase-sensitive images progressing from negative to positive in the normal inversion-recovery model.

Arguments: `'t1'` and `'t2'` are keywords for the fitting type, either `'t1'` for inversion-recovery or `'t2'` for decaying exponential (`'t2'` can also be used for saturation-recovery data).

`basename` is the name of a phasefile that represents the arrayed set of images. The phasefile should reside in the `planes` directory and must end in consecutive integer extensions, starting with 1.

`min_threshold` is a value for the lower limit for the fitting program. Pixels whose values in the first image are less than this threshold will not be fit and will be assigned values of zero in the synthesized resultant images.

See also: *User Guide: Imaging*

Related: `makephf` Transform and save images as phasefiles (M)

`t1image` Fit arrayed imaging data to T_1 exponential data (M)

`t2image` Fit arrayed imaging data to T_2 exponential data (M)
`vs` Vertical scale (P)

imprep Set up rf pulses, imaging and voxel selection gradients (M)

Applicability: Systems with imaging capabilities.

Syntax: `imprep`

Description: Sets up rf pulses, imaging gradients, and voxel selection gradients as required by the application, thus providing a universal “one pass” set up of rf power and gradient levels after sequence timing, field of view, and voxel selection parameters have been chosen. `imprep` scans the configuration parameter lists `plist` and `sslist` to determine which rf pulse parameters and gradients are active and then proceeds to set up parameter values.

See also: *User Guide: Imaging*

Related: `plist` Active pulse length parameter list (P)
`sslist` Conjugate gradient list (P)

in Lock and spin interlock (P)

Description: Controls error handling based on lock level and spin speed, and specifies action to be taken based on lock level failure or spinner failure. The action can be to generate an error and halt acquisition, or to generate a warning and continue acquisition.

Values: Can be set to one or two characters:

- If set to two characters, the first character specifies the action for lock failure and the second character specifies the action for spinner failure.
- If set to only one character, that character specifies the same action for either lock or spinner failure.

'n' stops any system checking so that acquisition continues regardless of the lock level or spin speed.

'w' makes the system check the lock level and the spin speed. A warning message is added to the log file if the lock level falls below a preset hardware level (about 20 on the lock meter) or if `spin` is set to a particular value and the spin speed goes out of regulation; however, acquisition is not stopped.

'y' makes the system check the lock level and spin speed. Acquisition is halted if the lock level falls below a preset hardware level (about 20 on the lock meter) or if `spin` is set to a particular value and the spin speed goes out of regulation.

See also: *Getting Started*

Related: `spin` Sample spin rate (P)

inadqt Set up parameters for INADEQUATE pulse sequence (M)

Applicability: All systems except *GEMINI 2000*.

Syntax: `inadqt`

Description: Sets up parameters for 2D INADEQUATE (Incredible Natural Abundance Double-Quantum Transfer Experiment).

Alternate: INADQT button in the 2D Pulse Sequence Setup Secondary Menu.

See also: *User Guide: Liquids NMR*

Related: `foldcc` Fold INADEQUATE data about 2-quantum axis (C)

index2 **Projection or 3D plane index selected (P)**

Applicability: All systems; however, although `index2` is available on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*, such systems can only process 3D data and cannot acquire 3D data.

Description: Stores whether a projection or 3D plane index is selected. It shows the current status only and cannot be used to select a plane or projection. This parameter is also displayed in the Status window below “Index.”

Values: 0 indicates a projection is selected.
1 to the half the Fourier number of the normal axis indicates a 3D plane is selected; the number is the index of the 3D plane.

See also: *User Guide: Liquids NMR*

Related: `dplane` Display a 3D plane (M)
`dproj` Display a 3D plane projection (M)
`nextpl` Display the next 3D plane (M)
`prevpl` Display the previous 3D plane (M)
`select` Select a spectrum or 2D plane without displaying it (C)

inept **Set up parameters for INEPT pulse sequence (M)**

Syntax: `inept`

Description: Sets up parameters for the INEPT (Insensitive Nuclei Enhanced by Polarization Transfer) experiment.

Alternate: *INEPT* button in the 1D Pulse Sequence Setup Menu.

See also: *User Guide: Liquids NMR*

Related: `ppcal` Proton decoupler pulse calibration (M)

initialize_iterate **Set iterate string to contain relevant parameters (M)**

Syntax: `initialize_iterate`

Description: Takes the current spin system (contained in `spinsys`) and derives from it relevant parameters. This can be used to control which parameters are iterated during a spin simulation iteration (e.g., for an ABC spin system, `iterate` is set to 'A,JAB,JAC,B,JBC,C').

Alternate: Set Params button in the Spin Simulation Main Menu

See also: *User Guide: Liquids NMR*

Related: `iterate` Parameters to be iterated (P)

input **Receive input from keyboard (C)**

Syntax: `input(<prompt><,delimiter>):var1,var2,...`

Description: Receives fields of characters from the keyboard and stores them into one or more variables.

Arguments: `prompt` is a string displayed on the command line.
`delimiter` is a character separating input fields. The default is a comma.
`var1,var2,...` are return values. `input` stores the values into as many of these arguments as given and ignores the rest of the input line.

Examples: `input:$b`
`input('Enter pulse width:'):pw`

```
input('x and y coordinates'):cr,crl
input('Enter lastname:firstname',':'): $last,$first
```

See also: *VNMR User Programming*

Related: **string** Create a string variable (C)

ins Integral normalization scale (P)

Description: Sets the integral value, independent of **is** and **vs**. Reported integral values are scaled by **fn**; that is, the reported integral of a given region is independent of **fn**. The **insref** parameter is also used to determine a reference integral value. The **setint** macro sets integral value.

See also: *Getting Started*

Related: **dlni** Display list of normalized integrals (M)
fn Fourier number in directly detected dimension (P)
is Integral scale (P)
insref Fourier number scaled value of an integral (P)
mark Determine intensity of spectrum at a point (C)
setint Set value of an integral (M)
vs Vertical scale (P)

ins2 2D volume value (P)

Description: Adjusts the 2D volume value, independent of **is** and **vs**. The volume is scaled by Fourier numbers for the two dimensions.

See also: *User Guide: Liquids NMR*

Related: **is** Integral scale (P)
ins2ref Fourier number scaled volume of a peak (P)
ll2d Automatic and interactive 2D peak peaking (C)
vs Vertical scale (P)

insref Fourier number scaled value of an integral (P)

Description: Set to the Fourier number scaled value of a selected integral. The reported integral values will be $(integral\ value) * ins / insref / fn$. If **insref** is “not used”, the sum of all integrals will be **ins**. The “not used” mode is the equivalent of the normalized integral mode. If **insref** is zero or not defined, the reported integrals will be $(integral\ value) * ins / fn$.

See also: *Getting Started*

Related: **fn** Fourier number in directly detected dimension (P)
ins Integral normalization scale (P)
liamp Amplitudes of integral reset points (P)
setint Set value of an integral (M)

ins2ref Fourier number scaled volume of a peak (P)

Description: Set to the Fourier number scaled volume of the selected peak. The reported volume is $volume * ins2 / ins2ref / fn / fn1$. If **ins2ref** is “not used”, sum of all volumes is **ins2**. The “not used” mode is equivalent to a normalized volume mode. If **ins2ref** is zero or not defined, the reported volume is $volume * ins2 / fn / fn1$.

See also: *User Guide: Liquids NMR*

Related: **fn** Fourier number in directly detected dimension (P)
fn1 Fourier number in first indirectly detected dimension (P)
ins2 2D volume value (P)
ll2d Automation and interactive 2D peak picking (C)

insert **Insert sample (M)**

Applicability: All systems (including *MERCURY* and *GEMINI 2000*) if spin control hardware is installed.

Syntax: `insert`

Description: Turns off the eject air, waits for the sample to slowly drop, and then turns off the slow drop air. The macro **i** is identical in function to `insert`.

See also: *Getting Started*

Related: **e** Eject sample (M)
eject Eject sample (M)
i Insert sample (M)

inset **Display an inset spectrum (C)**

Syntax: `inset`

Description: Displays the part of the spectrum between the two cursors as an inset. Before entering `inset`, run the `ds` command and display two cursors. The vertical position is shifted up about one-quarter of the height of the whole display canvas. The old spectrum remains on the screen, but the parameters shown at the bottom are relevant to the new display. If present, the integral trace is duplicated. The scale is also duplicated if it is present. After running `inset`, you can shift the displayed spectrum, expand it, or even contract it with the left and right mouse buttons.

See also: *Getting Started*

Related: **ds** Display a spectrum FID (C)

integ **Find largest integral in a specified region (C)**

Syntax: `integ<(highfield,lowfield)><:size,value>`

Description: Finds the largest absolute-value integral in the specified region, or the total integral if no reset points are present between the specified limits.

Arguments: `highfield` and `lowfield` are the limits of the region. The default values are the parameters `sp` and `sp+wp`, respectively.

`size` is a return value with the size of the largest integral. The size depends on the value of the parameter `is` and can be positive or negative.

`value` is a return argument with the value of the largest integral. This value depends on `ins`, `insref`, and `fn`, and is independent of `is`.

Examples: `integ:r1,r2`
`integ(500,1000):$height`
`integ(100+sp,300+sp):$ht,$val`

See also: *VNMR User Programming*

Related: **fn** Fourier number in directly detected dimension (P)
ins Integral normalization scale (P)
insref Fourier number scaled value of an integral (P)

<code>is</code>	Integral scale (P)
<code>rp</code>	Zero-order phase in directly detected dimension (P)
<code>sp</code>	Start of plot in directly detected dimension (P)
<code>wp</code>	Width of plot in directly detected dimension (P)

integrate **Automatically integrate 1D spectrum (M)**

Syntax: `integrate`

Description: A universal macro for selecting integral regions and adjusting the integrals in size and offset. Only if regions are not already selected, and if `intmod` is set to 'partial', will `integrate` call `region` to select integral regions. For proton spectra, the selection is done through the `hregions` macro; for ^{19}F and ^{31}P spectra (for wide spectral windows, multiplet spectra), `region` is called with optimized arguments, and for other nuclei (mostly decoupled, single-line spectra) other optimized parameters are used with `region`, such that lines consisting of a few data points only are recognized.

See also: *Getting Started*

Related:	<code>hregions</code>	Select integral regions in proton spectrum (M)
	<code>intmod</code>	Integral display mode (P)
	<code>isadj</code>	Adjust integral scale (M)
	<code>region</code>	Automatically select integral regions (C)

intmod **Integral display mode (P)**

Description: Controls display and plotting of the spectral integral.

Values: 'off' indicates that no integrals are displayed or plotted.

'full' indicates that all integral regions are displayed or plotted.

'partial' indicates that every other integral region is plotted (typically used to display integrals of only peaks and not of the baseline region).

See also: *Getting Started; User Guide: Liquids NMR*

Related:	<code>plc</code>	Plot carbon spectrum (M)
	<code>plh</code>	Plot proton spectrum (M)
	<code>plp</code>	Plot phosphorus spectrum (M)

intvast **Produces a text file of integral regions (M)**

Applicability: Systems with VAST accessory.

Syntax: `intvast (last)`

Description: `intvast` produces a text file, `integ.out` in the current experiment, containing the integrals of the partial regions of each spectra from wells 0 to `last`.

Arguments: `last` is the number last sample well. The default is 96.

See also: *User Guide: Liquids NMR*

Related:	<code>pintvast</code>	Plot the integrals (M)
----------	-----------------------	------------------------

iplan **Open interactive image planning tools (M)**

Applicability: Systems with imaging capabilities.

Description: `iplan` is an interactive image planning server loop with drawn-on screen control buttons. It captures mouse control in VNMR so that you click the screen Exit button to leave. The server opens the `tbox` transverse slice specification tool. By choosing a button in the graphics area, `tbox` can be stretched, tilted,

and moved. The number of slices and the area that they cover can also be adjusted. The Exit button calls the `rsliceplan` macro to load these setting for the next images.

See also: *User Guide: Imaging*

Related: `sliceplan` Set slice parameters for target slice (M)
`tbox` Draw a tilted box (C)

io Integral offset (P)

Description: Offset of the integral with respect to the spectrum.

Values: 0 to 200, in mm.

See also: *Getting Started*

ir Inversion recovery mode (P)

Applicability: Systems with imaging capabilities.

Description: Specifies whether to run in inversion recovery mode or in normal mode. In inversion recovery mode, the parameters `pipat`, `tpwri`, `pi`, and `ti` become active, providing a prepulse and delay for inversion recovery experiments.

Values: 'n' specifies normal mode and 'y' specifies inversion recovery mode.

See also: *User Guide: Imaging*

Related: `pi` Width of an inversion pulse (P)
`pipat` Shape of an inversion pulse (P)
`ti` Second delay in an inversion recovery sequence (P)
`tpwri` Intensity of an inversion pulse in dB (P)

is Integral scale (P)

Description: Multiplier that adjusts height of the displayed integral trace. Note that the `ins` parameter controls the integral value, and that `is` has no effect on integral value.

Values: 1 to 1e9

See also: *Getting Started*

Related: `ins` Integral normalization scale (P)
`ins2` 2D volume value (P)
`insref` Fourier number scaled value of an integral (P)
`integ` Find largest integral in a specified region (C)

isadj Automatic integral scale adjustment (M)

Syntax: `isadj<(height<,neg_height)>`

Description: Adjusts the height of the integrals in a display to make the tallest integral fit the paper. Optionally, the height of the maximum integral can be specified by an argument. Negative integrals, if present, are given a limit of 10 mm if parameter `io` is less than 10; otherwise, they are set so they end 5 mm above the spectrum. Negative integrals can also be given a height. Whichever part of the integrals (positive or negative) runs into the given limit will be used to scale `is`.

Arguments: `height` is the size, in mm, of the maximum integral on display. The default is the height that makes the tallest integral fit the paper.

`neg_height` is the desired height, in mm, of the largest negative integral. If `io` is less than 10, the default is 10; otherwise, the default height is 5 mm above the spectrum.

Examples: `isadj`
`isadj(100)`
`isadj(100,100)`

See also: *Getting Started*

Related: `io` Integral offset (P)
`is` Integral scale (P)
`isadj2` Automatic integral scale adjustment by powers of two (M)

isadj2 Automatic integral scale adjustment by powers of two (M)

Syntax: `isadj2<(height<,neg_height>):scaling_factor`

Description: Functionally the same as `isadj` except that `isadj2` adjusts the integral height by powers of two and returns the scaling factor to the calling macro.

Arguments: `height` is the size, in mm, of the maximum integral on display.
`neg_height` is the desired height, in mm, of the maximum negative integral on display.

`scaling_factor` is a return value giving the ratio of the new integral size to the old value (`new_is/old_is`).

Examples: `isadj2`
`isadj2(100)`
`isadj2(100,100)`
`isadj2(50):r1`

See also: *Getting Started*

Related: `is` Integral scale (P)
`isadj` Automatic integral scale adjustment (M)

iterate Parameters to be iterated (P)

Description: Contains parameters to be iterated during iterative spin simulations. If the Set Params button is used in setting up spin simulation parameters, `iterate` is initialized to a string containing all parameters appropriate to the current spin system.

Values: List of parameters, separated by commas (e.g., `iterate='A,B,JAB'`).

See also: *User Guide: Liquids NMR*

Related: `initialize_iterate` Set `iterate` string to contain relevant parameters (M)

jdesign Start Plot Designer Program (M)

Syntax: `jdesign`

Description: Opens the Plot Designer program, which provides mechanisms for positioning spectra, parameters, axes, and other plot output on a page. Text annotation and drawing features are available.

See also: *Getting Started*

Related: `jplot` Plot from Plot Designer program (C)

jexp Join existing experiment (C)

Syntax: (1) `jexp(exp_number)`
 (2) `jexp:$current_exp_number,$current_exp_name`

Description: Joins an existing experiment (syntax 1) or returns the current experiment number and experiment name (syntax 2). After entering this command, until another “join experiment” command or macro is entered, all actions (including changes of parameters, acquisition of data, and display of data) apply to the parameters and data of the experiment joined.

The `jexp` command does not refresh the display or display new experiment parameters. Use one of the macros `jexp1`, `jexp2`, etc. to join an experiment and have the screen refreshed and new parameters displayed.

Arguments: `exp_number` is a number from 1 to 9999 for existing experiment to be joined.

`$current_exp_number` is a return value with the current experiment number.

`$current_exp_name` is a return value with the current experiment name.

Examples: `jexp(3)`
`jexp:$expp`
`jexp:r1,n1`

Alternate: Exp# button (for example, Exp1) in the Workspace Menu.

See also: *Getting Started, User Guide: Liquids NMR*

Related: `cexp` Create an experiment (M)
`delexp` Delete an experiment (M)
`jexp1-jexp9` Join existing experiment and display new parameters (M)
`unlock` Remove inactive lock and join experiment (C)

jexp1-jexp9999 Join existing experiment and display new parameters (M)

Syntax: `jexp1, jexp2, jexp3, ..., jexp9999`

Description: Joins an existing experiment, refreshes the screen, and displays the main menu and the new experiment parameters. After entering this macro, until another “join experiment” command or macro is entered, all actions (including changes of parameters, acquisition of data, and display of data) apply to the parameters and data of the experiment joined.

To join an experiment without refreshing the screen and displaying new parameters, use the `jexp` command.

Examples: `jexp8`
`jexp354`

See also: *Getting Started; User Guide: Liquids NMR*

Related: `cexp` Create an experiment (M)
`delexp` Delete an experiment (M)
`jexp` Join existing experiment (C)
`unlock` Remove inactive lock and join experiment (C)

jplot Plot from Plot Designer program (C)

Syntax: `jplot(<'-setup'><,template>)`

Description: Starts plotting from the Plot Designer program to the current plotter.

Arguments: `'-setup'` is a keyword to start `jdesign`, the Plot Designer program, to allow interactive design and plotting.

`template` is the name of a file that will be used to make a plot of the current experiment. The default is a saved file chosen by the user.

Examples: `jplot`
`jplot('t1')`

See also: *Getting Started*

Related: `jdesign` Start Plot Designer program (M)
`jplotscale` Scale plot parameters (M)
`jplotunscale` Restore current experiment parameters (M)

jplotscale Scale plot parameters (M)

Applicability: Plot Designer program

Syntax: `jplotscale`

Description: Scales parameters of plotting area and an imported plot. When a region is drawn in Plot Designer, `jplotscale` automatically changes the plotting area parameters `wcmax` and `wc2max`. The parameters `io`, `is`, `vs`, `wc`, and `wc2` of a plot imported into a region are adjusted according to `wcmax` and `wc2max`.

See also: *Getting Started*

Related: `jplot` Plot from Plot Designer program (C)
`jplotunscale` Restore current experiment parameters (M)

jplotunscale Restore current experiment parameters (M)

Applicability: Plot Designer program

Syntax: `jplotunscale`

Description: Restores the current experiment parameters (`io`, `is`, `vs`, `wc`, and `wc2`) to a plot within a region that was created in Plot Designer. For example, entering `jplotunscale jexp2 jplotscale` restores the parameters of experiment 2 to a plot and then `jplotscale` applies the adjusted parameters to the plot.

See also: *Getting Started*

Related: `jplot` Plot from Plot Designer program (C)
`jplotscale` Scale plot parameters (M)

J

jumpret **Set up parameters for JUMPRET pulse sequence (M)**

Applicability: Sequence is not supplied with *MERCURY-VX*, *MERCURY* and *GEMINI 2000*.

Syntax: `jumpret`

Description: Sets up parameters for a jump-and-return water suppression sequence.

See also: *User Guide: Liquids NMR*

jwin **Activate and record activity in current window (M)**

Syntax: `jwin (pane_number)`

Description: Activates and records the activity in a specific window pane, created by `setgrid`, in the VNMR graphics window. `jwin` is executed when you double-click the left mouse button in a multiple-paned graphics window.

Arguments: `pane_number` is the number of the pane to join.

Examples: `jwin (2)`

See also: *Getting Started*

Related:	<code>curwin</code>	Current window (P)
	<code>fontselect</code>	Open FontSelect window (C)
	<code>mapwin</code>	List of experiment numbers (P)
	<code>setgrid</code>	Activate selected window (M)
	<code>setwin</code>	Activate selected window (C)

K

killft3d **Terminate any ft3d process started in an experiment (M,U)**

Syntax: `killft3d (exp_number)`

Description: Terminates any `ft3d` program that has been started in the specified VNMR experiment. `killft3d` can be executed from any experiment. For each `ft3d` process terminated, the relevant 3D data subdirectory is also deleted. Remote `ft3d` processes, denoted by the call name `ft3d` in the process table (displayed by the UNIX command `ps -azx`), are not directly terminated by `killft3d` but die of their own accord due to the deletion of the 3D data subdirectory.

The `killft3d` command can also be run as a shellscrip from UNIX. Its function is analogous to the associated VNMR macro.

Arguments: `exp_number` is a number from 1 to 9 that identifies the experiment that started the `ft3d` program.

Examples: `killft3d (4)`

See also: *User Guide: Liquids NMR*

Related: `ft3d` Perform a 3D Fourier transform (M,U)

killplot **Stop plot jobs and remove from plot queue (M)**

Syntax: `killplot`

Description: Kills all current plot jobs in the plot queue for the active plotter in VNMR, then removes the jobs from the plot queue. Unless the user executing `killplot` is `root`, only that user's plot jobs are deleted from the plot queue. To kill a plot that is in progress (i.e., a plot in which you have not entered `page`), use the `page ('clear')` command.

The plotter may have to be reinitialized after `killplot` is executed. To reinitialize the plotter, turn it off and then back on after a few seconds. Hewlett-Packard (HP) pen plotters appear to be more susceptible to this problem than the other HP output devices supported by VNMR.

If one port is configured to be both a printer and a plotter, `killplot` can cause both plot *and* print jobs to that port to be deleted. For example, if `printer='LaserJet_300'`, `plotter='LaserJet_300R'`, and a plot command `pl pscale page` is followed by a print command `p text (vnmruser+' /psglib/noesy.c')`, entering `killplot` deletes both jobs.

See also: *Getting Started*

Related: `killprint` Stop print jobs and remove from print queue (M)
`page` Move plotter forward one or more pages (C)
`pl` Plot spectra (C)
`pscale` Plot scale below spectrum or FID (C)
`p text` Print out a text file (M)
`showplotq` Display plot jobs in plot queue (M)

killprint **Stop print jobs and remove from print queue (M)**

Syntax: `killprint`

Description: Kills all current print jobs in the print queue for the active printer in VNMR, then removes the jobs from the print queue. Unless the user executing `killprint` is `root`, only that user's print job is deleted from the print queue. It is slightly possible that the printer may have to be reinitialized after the execution of this macro. To reinitialize the printer, turn it off, wait a few seconds, and then turn it back on.

If one port is configured to be both a printer and a plotter, `killprint` can cause both print *and* plot jobs to that port to be deleted. For example, if `printer='LaserJet_300'`, `plotter='LaserJet_300R'`, and a plot command `pl pscale page` is followed by a print command `ptext(vnmruser+' /psglib/noesy.c')`, entering `killprint` deletes both jobs.

See also: *Getting Started*

Related: `killplot` Stop plot jobs and remove from plot queue (M)
`ptext` Print out a text file (M)
`showprintq` Display print jobs in print queue (M)

kind **Kinetics analysis, decreasing intensity (M)**

Syntax: `kind`

Description: If the signal decreases exponentially toward a limit, the output is matched by $I = A1 * EXP(-T/TAU) + A3$. This macro supplies the necessary keywords to the `analyze` command, which uses the output of `fp` (i.e., the file `fp.out`) as input. The results can be displayed with `expl`.

See also: *User Guide: Liquids NMR*

Related: `analyze` Generalized curve fitting (C)
`expl` Display exponential/polynomial curves (C)
`fp` Find peak heights (C)
`kinds` Kinetic analysis, decreasing intensity, short form (M)
`kini` Kinetics analysis, increasing intensity (M)
`kinis` Kinetic analysis, increasing intensity, short form (M)

kinds **Kinetics analysis, decreasing intensity, short form (M)**

Syntax: `kinds`

Description: Produces a summary of the results from `kind`.

See also: *User Guide: Liquids NMR*

Related: `kind` Kinetics analysis, decreasing intensity (M)

kini **Kinetics analysis, increasing intensity (M)**

Syntax: `kini`

Description: If the signal increases exponentially toward a limit, the output is matched by $I = -A1 * EXP(-T/TAU) + A3 - A1$. This macro supplies the necessary keywords to the `analyze` command, which uses the output of `fp` (i.e., the file `fp.out`) as input. The results can be displayed with `expl`.

See also: *User Guide: Liquids NMR*

Related: `kind` Kinetics analysis, decreasing intensity (M)
`kinis` Kinetic analysis, increasing intensity, short form (M)

kinis **Kinetics analysis, increasing intensity, short form (M)**

Syntax: `kinis`

Description: Produces a summary of the results from `kini`.

See also: *User Guide: Liquids NMR*

Related: `kind` Kinetics analysis, decreasing intensity (M)
`kini` Kinetics analysis, increasing intensity (M)

large Use large graphics window (C)

Syntax: `large`

Description: Sets the Sun graphics window to use the full screen with the text window overlaid. The `large` command is only executed after any other commands have been processed; any current display is lost and has to be recalculated.

Alternate: Large button in the Permanent Menu.

See also: *Getting Started*

Related: `flip` Flip between large and small windows (C)
`small` Use small graphics window (C)

lastlk Last lock solvent used (P)

Description: Contains the name of the last lock solvent. Intended for use with the optional sample changer, this parameter is a user global variable (stored in the user's `global` file) and is not accessible to multiple users simultaneously. On a multiuser automation run, you should preferably access the last lock solvent from the file `/vnmr/acqqueue/lastlk`.

Values: String containing the name of the solvent.

See also: *User Guide: Liquids NMR*

Related: `solvent` Lock solvent (P)

lastmenu Menu to display when Return button is selected (P)

Description: Contains the name of the menu to display when the Return button is clicked on certain menus. For example, if the Phase F2 button in the 2D Processing menu (controlled by the file `process_2D`) is clicked, `lastmenu` is set to `'process_2D'`, the `ft` and `aph` commands are executed, the `ds` window is opened, and the Interactive 1D Spectrum Display menu (`ds_1` file) is displayed. Appearing in this menu is a Return button. Because `lastmenu` is still set to `'process_2D'`, clicking on the Return button redisplay the 2D Processing menu. `lastmenu` is stored in the `$vnmrsys/global` file.

Values: String containing the name of a menu (e.g., `'process_2D'`).

See also: *VNMR User Programming*

Related: `menu` Change status of menu system (C)
`newmenu` Select a menu without immediate activation (C)

latch Frequency synthesizer latching (P)

Applicability: All systems except *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Description: Configuration parameter for whether the PTS frequency synthesizer has latching capabilities (all digits of the frequency value are sent to the synthesizer at once). The value for each channel is by the Latching label in the CONFIG window (opened from `config`).

Values: `'n'` indicates the synthesizers do not have latching capabilities (Not Present choice from the CONFIG window).

'y' indicates the synthesizers have latching capabilities (Present choice from the CONFIG window). This value is used with all *UNITY/NOVA* and *UNITYplus* systems (all synthesizers on these systems have latching capabilities) and with *UNITY* and *VXR-S* systems with synthesizers that have latching capability.

See also: *VNMR and Solaris Software Installation*

Related: `config` Display current configuration and possibly change it (M)

1b Line broadening in directly detected dimension (P)

Description: Sets line broadening and exponential weighting along the directly detected dimension. This dimension is often referred to as the f_2 dimension in 2D data sets, the f_3 dimension in 3D data sets, etc.

Values: A positive value gives the desired line broadening, in Hz, which is then used to calculate a decaying exponential function of the form $\exp(-t*\pi*1b)$.

A negative value gives a resolution enhancement function (increasing exponential) of the form $\exp(-t*\pi*1b)$.

'n' turns off line broadening and exponential weighting.

See also: *Getting Started*

Related: `exp` Find exponential value of a number (C)
`1b1` Line broadening in 1st indirectly detected dimension (P)
`1b2` Line broadening in 2nd indirectly detected dimension (P)

1b1 Line broadening in 1st indirectly detected dimension (P)

Description: Sets line broadening and exponential weighting along the first indirectly detected dimension. This dimension is often referred to as the f_1 dimension in multidimensional data sets. `1b1` works analogously to the parameter `1b`. The "conventional" parameters (`1b`, `gf`, etc.) operate on the detected FIDs, while this "2D" parameter is used during processing of the interferograms.

Values: A positive value gives the desired line broadening, in Hz, which is then used to calculate a decaying exponential function of the form $\exp(-t*\pi*1b1)$. A typical value is between 0.0001 to 1000 Hz.

A negative value gives a resolution enhancement function (increasing exponential) of the form $\exp(-t*\pi*1b1)$.

'n' turns off line broadening and exponential weighting.

See also: *User Guide: Liquids NMR*

Related: `exp` Find exponential value of a number (C)
`1b` Line broadening in directly detected dimension (P)
`1b2` Line broadening in 2nd indirectly detected dimension (P)

1b2 Line broadening in 2nd indirectly detected dimension (P)

Description: Sets line broadening and exponential weighting along the second indirectly detected dimension. This dimension is often referred to as the f_2 dimension in multidimensional data sets. `1b2` works analogously to the parameter `1b`. `1b2` can be set with `wti` on the 2D interferogram data.

Values: A positive value gives the desired line broadening, in Hz, which is then used to calculate a decaying exponential function of the form $\exp(-t*\pi*1b2)$.

A negative value gives a resolution enhancement function (increasing exponential) of the form $\exp(-t*\pi*1b2)$.

'n' turns off line broadening and exponential weighting.

See also: *User Guide: Liquids NMR*

Related: **exp** Find exponential value of a number (C)
lb Line broadening in directly detected dimension (P)
wti Interactive weighting (C)

lc1d Pulse sequence for LC-NMR (M)

Applicability: Systems with LC-NMR accessory.

Syntax: `lc1d`

Description: Creates parameters to set up a pulse sequence that can be used to start an LC-NMR run, including triggering the injection of a sample, and can be used also to obtain multiple solvent-suppressed spectra using multifrequency Shifted Laminar Pulses (SLP) and gradients. The sequence is coded without a **d2** variable, thus allowing **ni** to be used to obtain a series of spectra without resulting in any delay in the sequence being incremented.

The sequence requires a phase table, `lc1d`, to be found in the `tablib` directory. Phases of the selective pulses, the observe pulse, and the receiver and separately controlled by phase variables.

Note that the `lc1d` sequence uses power scaling of shaped pulses, which is supported starting in VNMR 5.2. Because of this feature, this sequence *will not run* in earlier versions of VNMR.

See also: *User Guide: Liquids NMR*

lcpar2d Create 2D LC-NMR acquisition parameters (M)

Applicability: Systems with LC-NMR accessory.

Syntax: `lcpar2d`

Description: Creates the acquisition parameters **ni**, **sw1**, and **phase**, which can be used to acquire a 2D LC-NMR data set. `lcpar2d` is functionally the same as `addpar('2d')`.

See also: *User Guide: Liquids NMR*

Related: **addpar** Add selected parameters to current experiment (M)
lcset2d General setup for 2D LC-NMR experiments (M)

lcpeak Peak number (P)

Applicability: Systems with LC-NMR accessory.

Description: Contains the number of the peak being sensed or the loop being flushed.

See also: *User Guide: Liquids NMR*

lcplot Plot LC-NMR data (M)

Applicability: Systems with LC-NMR accessory.

Syntax: `lcplot`

Description: Plots LC-NMR data. This macro is executed with the Plot LC-NMR button on the Spare pane when LC-NMR is active.

See also: *User Guide: Liquids NMR*

lcpsgset **Set up parameters for various LC-NMR pulse sequences (M)**

Applicability: Systems with LC-NMR accessory.

Syntax: `lcpsgset(file,parameter1,parameter2,...,parameterN)`

Description: Sets up parameters for various LC-NMR pulse sequences using information in a `parlib` file. Rather than returning the entire parameter file, `lcpsgset` returns the parameters listed. `lcpsgset`, in general, is never entered from the keyboard but is used as part of experiment setup macros.

Arguments: `file` is the file from the user or system `parlib` that provides information on setting up parameters listed. The parameters `seqfil` and `pslabel` are set to the supplied file name.

`parameter1,parameter1,...,parameterN` are 1 to 11 parameters to be returned from the `parlib` file.

Examples: `lcpsgset('lccosy','ds','ap','ss','d1','axis','phase')`

See also: *User Guide: Liquids NMR*

Related: `pslabel` Pulse sequence label (P)
`seqfil` Pulse sequence name (P)

lcset2d **General setup for 2D LC-NMR experiments (M)**

Applicability: Systems with LC-NMR accessory.

Syntax: `lcset2d(experiment<,F2_dig_res<,F1_dig_res>>)`

Description: Runs the macro `lcpar2d` to create new parameters needed for 2D LC-NMR experiments, then selects starting values for a number of parameters. The `lcset2d` macro is “internal” and not normally entered directly by the user.

Arguments: `experiment` is the name of a 2D LC-NMR experiment.

`F2_dig_res` is the f_2 digital resolution desired, in Hz/pt.

`F1_dig_res` is the f_1 digital resolution desired, in Hz/pt.

Examples: `lcset2d('lcnoesy')`

See also: *User Guide: Liquids NMR*

Related: `lcpar2d` Create 2D LC-NMR acquisition parameters (M)

left **Set display limits to left half of screen (C)**

Syntax: `left`

Description: Sets the horizontal control parameters `sc` and `wc` to produce a display (and subsequent plot) in the left half of a screen (and page). For 2D data, space is left for the scales.

Alternate: Left button on the 1D Display Size Selection Menu, or
 Left button on the 2D Display Size Selection Menu.

See also: *Getting Started; User Guide: Liquids NMR*

Related: `center` Set display limits for center of screen (C)
`full` Set display limits for a full screen (C)
`fullt` Set display limits for full screen with room for traces (C)
`right` Set display limits for right half of screen (C)
`sc` Start of chart (P)
`wc` Width of chart (P)

legrelay Independent control of magnet leg relay (P)

Applicability: All systems except *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Description: Gives override capability over the magnetic leg high and low (broad) band rf signal routing. This parameter does not normally exist but can be created by the user with the command `create('legrelay','string')`.

The `legrelay` override is operational only on standard systems shipped starting in November 1990 and on certain special systems shipped before that date. A system includes the override capability if it uses N-type connectors instead by BNC connectors on the magnet leg.

Values: 'n' indicates normal logic is used to set the leg relay.

'h' indicates the leg relay is set to the high band

'l' indicates the leg relay is set to the low (broad) band.

Any other value results in an error message and an abort of pulse sequence generation.

See also: *VNMR User Programming*

Related: `create` Create new parameter in a parameter tree (C)

length Determine length of a string (C)

Syntax: `length(string):$string_length`

Description: Returns the length in characters of a specified string.

Arguments: `string` is zero or more characters enclosed in single quotes.

`string_length` is the number of characters (a real number) in `string`.

Examples: `length('abc'):r1`
`length(solvent):$len`

See also: *VNMR User Programming*

Related: `substr` Select a substring from a string (C)

lf List files in directory (C)

Syntax: `lf<(directory)>`

Description: Lists the files in a directory, with output on the text output window. Directories are suffixed by "/", executable files by "*", and links by "@".

Arguments: `directory` is the name of a directory. The default is the current working directory. `lf` is equivalent to the UNIX command `ls -F` and uses the same options (e.g., `-l` for a long listing such as `lf('-l *.fid')`).

Examples: `lf`
`lf('data')`
`lf('-l *.fid')`

See also: *Getting Started*

Related: `dir` List files in directory (C)

`ls` List files in directory (C)

liamp Amplitudes of integral reset points (P)

Description: Stores the integral amplitudes at the integral reset points for a list of integrals. To display the values of `liamp`, enter `display('liamp')`. Values of `liamp` can also be accessed in MAGICAL macros using, for example, `liamp[$i]`. Values are stored as absolute numbers (summations of data point

values) and, as such, are a function of the parameter `fn`. The values displayed by the `dli`, `pir`, and `dpir` programs are related to `liamp` values by the relationship:

$$\text{Displayed or plotted integral} = \text{liamp}[i] * \text{is} / (\text{fn} / 128) * \text{ins}$$

See also: *Getting Started*

Related:	<code>display</code>	Display parameters and their attributes (C)
	<code>dli</code>	Display list of integrals (C)
	<code>dpir</code>	Display integral amplitudes below spectrum (C)
	<code>fn</code>	Fourier number in directly detected dimension (P)
	<code>lifrq</code>	Frequencies of integral reset points (P)
	<code>pir</code>	Plot integral amplitudes below spectrum (C)

`lifrq` **Frequencies of integral reset points (P)**

Description: Stores the frequencies of integral reset points for a list of integrals. The frequencies are stored in Hz and are *not* adjusted by the reference parameters `rfl` and `rfp`.

See also: *Getting Started*

Related:	<code>liamp</code>	Amplitudes of integral reset points (P)
	<code>rfl</code>	Ref. peak position in directly detected dimension (P)
	<code>rfp</code>	Ref. peak frequency in directly detected dimension (P)

`listenoff` **Disable receipt of messages from send2Vnmr (M)**

Syntax: `listenoff`

Description: Deletes the file `$vnmruser/.talk`, thereby disallowing `send2Vnmr` to send commands to VNMR

See also: *VNMR User Programming*

Related:	<code>listenon</code>	Enable receipt of messages from <code>send2Vnmr</code> (M)
	<code>send2vnmr</code>	Send a command to VNMR (U)

`listenon` **Enable receipt of messages from send2Vnmr (M)**

Syntax: `listenon`

Description: Writes files with the VNMR port number that `/vnmr/bin/send2Vnmr` needs to talk to VNMR. The command then to send commands to VNMR is `/vnmr/bin/send2Vnmr $vnmruser/.talk` command.

See also: *VNMR User Programming*

Related:	<code>listenoff</code>	Disable receipt of messages from <code>send2Vnmr</code> (M)
	<code>send2vnmr</code>	Send a command to VNMR (U)

`lkof` **Track changes in lock frequency (P)**

Description: Tracks changes in the lock frequency resulting from changes in the solvent, and minor changes caused by the magnet drifting. The frequency units for `lkof` are in Hz, analogous to `sfrq` and `tof`, or `dfrq` and `dof`. `lkof` affects two components of the system: autolock on the console and `acqi` on the host computer. On ^{UNITY}INOVA systems, if `lkof` exists, it offsets the current value of the `lockfreq` parameter.

See also: *Getting Started*

Related:	<code>lockfreq</code>	Lock frequency (P)
----------	-----------------------	--------------------

112d Automatic and interactive 2D peak picking (C)

Syntax: (1) `112d<(options)><:$num>`
 (2) `112d('info'<, #>):$peak_number, $f1, $f2, $amplitude, $volume, $label, $comment, $FWHH1, $FWHH2, $f1_min, $f1_max, $f2_min, $f2_max`

Description: Automatically finds and integrates peaks that are above the threshold `th` in a 2D spectrum or a 2D plane of a 3D spectrum, and writes the peak location, volume, full-width at half-height (FWHH), volume, and the boundaries of the integrated region to a file in the `112d` subdirectory of the current experiment directory. For 2D spectra, the file name is `peaks.bin`, and for 2D planes of 3D spectra, the file name is `peaks_f##f##_#.bin`, where `f##f##` gives the plane direction (e.g., `f1f3`) and the final `#` gives the number of the plane. For easy import and export of peak data, `112d` also allows insertion and deletion of peaks interactively as well as reading and writing of text peak files.

Two-dimensional volumes are scaled in a manner analogous to 1D integrals, using the parameters `ins2` and `ins2ref`. The `ins2ref` parameter is the Fourier number scaled value of a selected volume. The reported value of a peak volume is $(\text{unscaled volume}) \times \text{ins2} / \text{ins2ref} / \text{fn} / \text{fn1}$. The unscaled volume of a peak can be obtained from the command

`112d('info', peak#).ins2ref` can be set to the unscaled value divided by `fn` and `fn1`. The report volume for that peak is then the value of `ins2`.

Arguments: `options` (syntax 1) are any of the following (`dconi` is not necessarily active):

- `'adjust'` is a keyword to adjust the bounds of all peaks in the displayed area so that no boundaries overlap, and then to recalculate peak volumes.
- `'draw'` is a keyword to draw the peaks, boxes, numbers, and labels on the spectrum based on the value of the parameter `112dmode`.
- `'info'`, `'total'` displays the total number of peaks in the current peak table. If a single return value is requested, printing is suppressed and the total number of peaks is returned.
- `'peaks'` is a keyword to find all peaks in the displayed area above a threshold `th`. If `dconi` is active and in the box mode, `112d` finds peaks only in the area defined by the cursors. The `'peaks'` option is the default if no arguments are entered.
- `'pos'` or `'neg'` keywords can be used in addition to `'peak'`, `'volume'`, or `'clear'` to operate only on positive or negative peaks.
- `'read'<, file >` reads in a binary peak file, where `file` is the name of the peak file. If a full path is not specified, the file is searched for first in the current working directory and then in the `112d` subdirectory of the current experiment directory.
- `'readtext'<, file>` reads in a text peak file, where `file` is the name of the peak file. If a full path is not specified, the file is searched for first in the current working directory and then in the `112d` subdirectory of the current experiment directory.
- `'reset'` is a keyword to delete all peaks in the peak table.
- `'volume'` is a keyword to find the bounds of each peak in the displayed area and integrate this area.
- `'writetext'<, file>` writes a peak file to a text file, where `file` is the name of the text file written. If a full path is not specified, the file is written in the current working directory.

options (syntax 1) can also be any of the following (`dconi` must be active):

- `'clear'` is a keyword to delete all peaks in the displayed region if in the `dconi` cursor mode, or to delete all peaks within the cursors if in the `dconi` box mode.
- `'combine'` is a keyword to combine all peaks within the area defined by the cursors into a single peak (in `dconi` box mode only). The center of the new peak is at the average of all combined peaks' centers, and the bounds of this peak contains the maximum extents of the combined peaks' bounds. If all combined peaks have the same label, this label is assigned to the new peak. **CAUTION: All individual peaks to be combined are deleted prior to the creation of the new combination peak, and there is no automatic way to restore the original peaks. Therefore, it is recommended that you make a backup copy of the peak file prior to using this option.**
- `'comment'` is a keyword to prompt for an 80-character comment. The comment is assigned to the nearest peak in the `dconi` cursor mode or to all peaks within the cursors in the `dconi` box mode.
- `'comment', text` executes the `'comment'` option using the string entered for `text` instead of prompting for a comment.
- `'label'` is a keyword to prompt for a 15-character label. The label is assigned to the nearest peak in `dconi` cursor mode or assigned to all peaks within the cursors in `dconi` box mode. To erase an existing label, enter a label consisting of one or more spaces.
- `'label', text` executes the `'label'` option using the string entered for `text` instead of prompting for a label.
- `'mark'` is a keyword to insert a peak at the current cursor position if in the `dconi` cursor mode. If in the `dconi` box mode, `'mark'` is a keyword to integrate the area within the cursors and assign that area to all peaks within the cursors that do not have their bounds already defined. If there are no peaks within the area defined by the cursors, using `'mark'` finds the highest point within this area, marks that as a peak, integrates the area within the cursors, and assigns that area to the peak. The displayed values of the volume integrals are scaled by `ins2` and `ins2ref` and the Fourier number of the 2D experiment.
- `'unmark'` is a keyword to delete the nearest peak if in `dconi` cursor mode. If in the `dconi` box mode, `'unmark'` deletes all peak bounds that are completely within the area defined by the cursors. Peaks are not deleted in the box mode.

options (syntax 1) also can be any of the following (`dconi` does not have to be active because `l12d` is executed on a peak number):

- `'combine', #1, #2, ...` executes the `'combine'` option on the list of peak numbers that follow the `'combine'` keyword. If a single return value is requested, the peak number of the new combination peak is returned.
- `'comment', text, #` executes the `'comment'` option on peak `#` using the string entered for `text` instead of prompting for a comment.
- `'label', text, #` executes the `'label'` option on peak `#` using the string entered for `text` instead of prompting for a label.
- `'unmark', #` deletes peak number `#`.

\$num (syntax 1) is a return value set to the total number of peaks that have been picked unless the arguments 'combine', #1, #2, . . . are used, in which case \$num is the number of the newly created combination peak.

Syntax 2 arguments are the following:

- 'info' <, #> displays information in the text window about peak number #. If no peak number is included, **dconi** must be active and the default is the peak nearest to the cursor. If return values are requested, the display is suppressed.
- \$peak_number is a return value set to the number of the peak, either the second argument # or, if no value is given for #, the peak nearest to the cursor in **dconi**.
- \$f1 and \$f2 are return values set to the peak frequencies in f_1 and f_2 of peak \$peak_number.
- \$amp is a return value set to the amplitude of peak \$peak_number.
- \$vol is a return value set to the unscaled volume of \$peak_number peak. This value can be used to set the **ins2ref** parameter.
- \$label is a return value set to the label of peak \$peak_number.
- \$comment is a return value set to the comment about \$peak_number.
- \$FWHH1 and \$FWHH2 are return values set to full-width at half-height of \$peak_number.
- \$f1_min, \$f1_max, \$f2_min, \$f2_max are return values set to the bounds of \$peak_number.

Examples: `l12d`
`l12d:$npeaks`
`l12d('volume')`
`l12d('read','peaklist.inp')`
`l12d('mark')`
`l12d('label','Peak 1')`
`l12d('info','total'):$npeaks`
`l12d('combine',3,4,5,6):$cpn`
`l12d('info',3):$num,$f1,$f2,$amp,$vol,$label`

Alternate: 2D Line List button in the Automatic COSY Analysis Menu.

See also: *User Guide: Liquids NMR*

Related:	dconi	Interactive 2D contour display (C)
	ins2	2D volume value (P)
	ins2ref	Fourier number scaled volume of a peak (P)
	l12dbackup	Copy current l12d peak file to another file (M)
	l12dmode	Control display of peaks picked by l12d (P)
	parl12d	Create parameters for 2D peak picking (M)
	p112d	Plot results of 2D peak picking (C)
	th	Threshold (P)
	th2d	Threshold for integrating peaks in 2D spectra (P)
	xdiag	Threshold for excluding diagonal peaks when peak picking (P)

l12dbackup Copy current l12d peak file to another file (M)

Syntax: `l12dbackup<(file)>`

Description: Backs up the current **l12d** peak file by copying it to a file with a different file name. The default **l12d** peak file is `peaks.bin` for 2D data.

Arguments: `file` is the name to be given to the backup file. If a full path is not specified, the file is written to the current working directory. If no argument is provided, the system prompts for a file name. If no file name is specified at the prompt, the default `112d` peak file name with `.bck` appended is used.

See also: *User Guide: Liquids NMR*

Related: `112d` Automatic and interactive 2D peak picking (C)

112dmode Control display of peaks picked by 112d (P)

Description: Sets the display attributes of peaks picked by the `112d` command

Values: A string variable composed of 4 characters, with each character taking the value 'y' (display the peak attribute) or 'n' (do not display the attribute). The first character determines if a "+" is drawn on the screen in `dconi` displays to mark peaks, the second character controls the drawing of the peak number, the third character controls drawing of the peak bounds box, and the last character controls drawing of the peak label.

See also: *User Guide: Liquids NMR*

Related: `112d` Automatic and interactive 2D peak picking (C)

11amp List of line amplitudes (P)

Description: Stores a list of line amplitudes above the threshold set by `th`.

See also: *Getting Started*

Related: `dll` Display listed line frequencies and intensities (C)
`11frq` List of line frequencies (P)
`th` Threshold (P)

11frq List of line frequencies (P)

Description: Stores a list of line frequencies above the threshold set by `th`. Frequencies are stored in Hz and are *not* adjusted by reference parameters `rfl` and `rfp`.

See also: *Getting Started*

Related: `11amp` List of line amplitudes (P)
`rfl` Ref. peak position in directly detected dimension (P)
`rfp` Ref. peak frequency in directly detected dimension (P)
`th` Threshold (P)

ln Find natural logarithm of a number (C)

Syntax: `ln(value) <:n>`

Description: Finds the natural logarithm (base e) of a number. To convert the value to base 10, use $\log_{10}x = 0.43429 * \ln(x)$.

Arguments: `value` is a number.

`n` is the return value giving the logarithm of `value`. The default is to display the logarithmic value in the status window.

Examples: `ln(.5)`
`ln(val):ln_val`

See also: *VNMR User Programming*

Related: `atan` Find arc tangent of a number (C)
`cos` Find cosine value of an angle (C)

<code>exp</code>	Find exponential value of a number (C)
<code>sin</code>	Find sine value of an angle (C)
<code>tan</code>	Find tangent value of an angle (C)

load **Load status of displayed shims (P)**

Description: Sets whether shim values are used. `load` is automatically set to 'y' by the `rts` and is automatically set to 'n' by `su`, `go`, `au`, and `shim`. On ^{UNITY}*INOVA* systems, shim DAC values are automatically loaded after the console is rebooted (the last values returned before the console was rebooted).

Values: 'y' begins any noninteractive shimming process or data acquisition after loading the shim DACs with the shim values from the current experiment. It also prevents `acqi` from delivering shim values to that experiment.

'n' begins any noninteractive shimming process or data acquisition with the current values stored in the shim DACs. Shim values in the current experiment are ignored.

See also: *Getting Started*

Related:	<code>acqi</code>	Interactive acquisition display process (C)
	<code>au</code>	Submit experiment to acquisition and process data (C)
	<code>go</code>	Submit experiment to acquisition (C)
	<code>rts</code>	Retrieve shim coil settings (C)
	<code>shim</code>	Submit an autoshim experiment to acquisition (C)
	<code>su</code>	Submit a setup experiment to acquisition (M)

loadcolors **Load colors for graphics window and plotters (M)**

Syntax: `loadcolors<(color_file)>`

Description: Loads the color table for VNMR graphics window and plotters. `loadcolors` is generated by the `color` program and includes a series of `setcolor` commands. On bootup, the `bootup` macro calls `loadcolors` to set the graphics and plotter colors.

The `loadcolors` macro checks the value of `maxpen` to decide if the plotter supports colors. If `maxpen` is greater than 1, a color printer is configured.

Arguments: `color_file` is the name of the file to load. `loadcolors` first searches for this file in the directory `$vnmruser/templates/` directory. If not found there, `loadcolors` then searches the `user_templates/vnmr` directory. The default is a color table with the same name as the value of the plotter parameter that `loadcolors` searches for in the same two directories.

Examples: `loadcolors`
`loadcolors('mycolortable')`

See also: *User Guide: Imaging*

Related:	<code>bootup</code>	Macro executed automatically when VNMR activated (M)
	<code>color</code>	Select plotting colors from a graphic interface (M)
	<code>maxpen</code>	Maximum number of pens to use (P)
	<code>setcolor</code>	Set colors for graphics window and for plotters (C)

loc **Location of sample in tray (P)**

Description: Indicates whether a sample changer is present and enabled, present but disabled, or not present. If the changer is present and enabled, the value of `loc` sets the location in the tray of the sample in use or to be used. The `loc` parameter is stored in the global tree. When an acquisition is started, certain global

parameters, including `loc`, are saved with the experiment parameters. The `saveglobal` parameter specifies which global parameters are saved.

The `auto_au` macro controls most of the automation features, including setting the value of `loc`.

Values: A number between 1 and `traymax` indicates the sample location.

0 indicates the changer is not present or disabled.

See also: *Getting Started; User Guide: Liquids NMR*

Related: `auto_au` Controlling macro for automation (M)
`saveglobal` Save selected parameters from global tree (P)
`traymax` Sample changer tray size (P)

location **Get coordinate information from an image display (M)**

Applicability: Systems with imaging capabilities.

Syntax: `location`

Description: Provides coordinate information from an image display using the 2D cursor package. This program can be used, along with the interactive image viewing program `dconi`, to provide coordinate data. You should position the 2D cursor at the desired point and enter `location` in the input window. Coordinates are printed on line 3 in the VNMR status window. Coordinate values are supplied in both the magnet frame (X, Y, Z) and logical frame (R, P, S), where the letters R, P, and S denote read, phase encode, and slice select axes, respectively. A typical use for `location` is to set the value of the parameter `pro` for FOV position of the image center. Position the cursor at the point desired to become the new image center, enter `location`, and set the value of `pro` to the R coordinate for the logical frame.

See also: *User Guide: Imaging*

Related: `dconi` Interactive 2D contour display (C)
`pro` Position of image center on the readout axis (P)

lock **Submit an Autolock experiment to acquisition (C)**

Syntax: `lock`

Description: Performs an automatic locking operation using the acquisition computer, optimizing lock power, phase, and gain. If necessary, `lock` obtains lock through a software-controlled search (required on ^{UNITY}*INOVA*, *MERCURY-VX*, *MERCURY*, *UNITYplus*, and *GEMINI 2000*). `lock` is the only method to automatically adjust lock phase (usually needed only after probe change or lock channel tuning). `lock` also sets the rf frequencies, decoupler status, and temperature.

See also: *Getting Started*

Related: `au` Submit experiment to acquisition and process data (C)
`change` Submit a change sample experiment to acquisition (M)
`ga` Submit experiment to acquisition and FT the result (C)
`go` Submit experiment to acquisition (C)
`sample` Submit change sample, autoshim experiment to acquisition (M)
`shim` Submit an Autoshim experiment to acquisition (C)
`spin` Submit a spin setup experiment to acquisition (C)
`su` Submit a setup experiment to acquisition (M)

lockacqtc Lock loop time constant during acquisition (P)

Applicability: All systems except *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Description: Controls time constant of lock loop during acquisition (i.e., time constant by which the lock feedback corrects disturbances of the magnetic field).

Values: On *UNITYINOVA* and *UNITYplus*: 1, 2, 3, or 4 (where 1 sets 1.2 seconds, 2 sets 4.7 seconds, 3 sets 12 seconds, and 4 sets 48 seconds).
On *UNITY* and *VXR-S*: 1 or 2 (where 1 sets 1 second and 2 sets 200 seconds).
If `lockacqtc` does not exist, it is set to 48 seconds on a *UNITYINOVA* or *UNITYplus*, and 200 seconds on a *UNITY* or *VXR-S*. All systems are designed to work well with the default settings, and there should rarely be a reason to alter the lock time constant. However, to experiment with other values, create `lockacqtc` and set a new value:

```
create('lockacqtc','integer','global')
setlimit('lockacqtc',4,1,1,'global') lockacqtc=n
```

where *n* is the new value.

See also: *Getting Started*

Related: `create` Create new parameter in a parameter tree (C)
`locktc` Lock time constant (P)
`setlimit` Set limits of a parameter in a tree (C)

lockfreq Lock frequency (P)

Description: Sets system lock frequency. The value is entered using the Lock Frequency label in CONFIG window (opened from `config`). **The value of `lockfreq` must be set correctly in order to observe NMR signals.**

On *UNITYINOVA* systems, `lockfreq` can find the lock signal or resonance. Traditionally, Varian spectrometers have used the parameter `z0` for this purpose; however, using `lockfreq` can require less shimming when switching solvents and less adjustment to the lock phase. To use `lockfreq`, set `z0='n'`.

Values: 1 to 160 (in MHz), 'n'

UNITYINOVA, *MERCURY-VX*, *MERCURY*, *UNITYplus*, and *GEMINI 2000* use the true ²H frequency. Typical values of `lockfreq` are shown in the chart below. On *UNITYINOVA* and *UNITYplus*, step size is approximately 2.384 Hz; on *MERCURY-VX* and *MERCURY*, step size is 0.05 Hz; on *GEMINI 2000*, step size is about 76 Hz.

On *UNITY* and *VXR-S* (except 200-MHz systems), the lock transmitter is equipped with a series of thumbwheel switches that adjust the lock frequency if the field drifts out of the range of the field offset control (the `z0` parameter). Typical starting values of the switches are shown in the chart. As the field decays, the number is set downward to lower the lock frequency.

¹ H Frequency	<i>UNITYINOVA</i> , <i>UNITYplus</i>	<i>MERCURY</i>	<i>GEMINI 2000</i>	<i>UNITY</i> , <i>VXR-S</i>
200	30.710	30.6976	30.697612	...
300	46.044	46.0625	46.062489	1.206
400	61.395	61.471	61.463000	1.145
500	76.729	1.479
600	92.095	153.845
750	115.250

For all systems, refer to the manual *VNMR and Solaris Software Installation* for details on finding the correct lock frequency.

Commands such as `go`, `lock`, `shim`, and `su` reset the lock frequency in the console to the current value of `lockfreq`. On *UNITYINOVA MERCURY-VX* and *MERCURY*, lock frequency in the console can be set with the `sethw` command.

Note that on the *UNITYINOVA* only, `lockfreq` is offset by the value of `lkof`, if that parameter exists, but `sethw` directly uses its numeric argument, without any offset by `lkof`.

See also: *VNMR and Solaris Software Installation; Getting Started*

Related:	<code>config</code>	Display current configuration and possibly change it (M)
	<code>go</code>	Submit experiment to acquisition (M)
	<code>lkof</code>	Track changes in lock frequency (P)
	<code>lock</code>	Submit an Autolock experiment to acquisition (C)
	<code>sethw</code>	Set values for hardware in acquisition system (C)
	<code>setlockfreq</code>	Set lock frequency on a <i>UNITYINOVA</i> or <i>UNITYplus</i> system (C)
	<code>shim</code>	Submit an Autoshim experiment to acquisition (C)
	<code>su</code>	Submit a setup experiment to acquisition (M)
	<code>z0</code>	Z0 field position (P)

lockgain **Lock gain (P)**

Description: Contains the current lock gain value as set by computer control. The value is stored in `vnmr/sys/global` and can be examined by typing `lockgain?`.

Values: On *UNITYINOVA* and *UNITYplus*, 0 to 48 dB, in 1-dB steps.
 On *MERCURY-VX* and *MERCURY*, 0 to 38 dB, in 1-dB steps.
 On *GEMINI 2000*, 0 to 30 dB, in 10-dB steps.
 On *UNITY* and *VXR-S*, 0 to 70 dB, in 1-dB steps.

See also: *Getting Started*

lockphase **Lock phase (P)**

Description: Contains the current lock phase. The value is stored in `vnmr/sys/global` and can be examined by typing `lockphase?`.

Values: 0 to 360, in degrees, in 1.4-degree steps.

See also: *Getting Started*

lockpower **Lock power (P)**

Description: Contains the current lock power value as set by computer control. The value is stored in `vnmr/sys/global` and can be examined by typing `lockpower?`.

Values: On *UNITYINOVA* and *UNITYplus*, 0 to 68 dB, in 1-dB steps, 68 is full power.
 On *MERCURY-VX* and *MERCURY*, 0 to 48 dB, in 1-dB steps, 48 is full power.
 On *GEMINI 2000*, 0 to 40 dB, in 1-dB steps.
 On *UNITY* and *VXR-S*: 0 to 63 dB, in 1-dB steps, 63 is full power

See also: *Getting Started*

locktc **Lock time constant (P)**

Applicability: All systems except *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Description: Controls lock loop time constant when system is not performing acquisition (idle, lock display, shim display, FID display, autoshim, autolock, etc.).

Values: On *UNITYINOVA* and *UNITYplus*: 1, 2, 3, or 4 (where 1 corresponds to 1.2 seconds, 2 to 4.7 seconds, 3 to 12 seconds, and 4 to 48 seconds). On *UNITY* and *VXR-S*: 1 or 2 (where 1 corresponds to 1 second and 2 to 200 seconds). If `locktc` does not exist, the system uses a value of 1, the fastest value. To experiment with other value, create `locktc` and set a value (e.g.,
`create('locktc','integer','global')`
`setlimit('locktc',4,1,'global') locktc=2`).

See also: *Getting Started*

Related: `create` Create new parameter in a parameter tree (C)
`lockacqtc` Lock acquisition time constant (P)
`setlimit` Set limits of a parameter in a tree (C)

logate Transmitter local oscillator gate (P)

Applicability: *UNITYINOVA* and *UNITYplus* systems.

Description: Specifies whether the transmitter local oscillator (L.O.) is gated with the transmitter rf output or with the transmitter I.F. (intermediate frequency).

The `logate` parameter does not exist in most parameter sets; the system internally sets it to '1'. To use the value 's', create `logate` and change the value by entering: `create('logate','string')`
`setenumeral('logate',2,'1','s') logate='s'`.

Values: '1' makes the transmitter L.O. gate with the rf output, producing better signal-to-noise, usually most important in liquids NMR.

's' makes the transmitter L.O. gate with the I.F. signal, producing sharper pulses, especially important in solid-state NMR.

See also: *User Guide: Solid-State NMR*

Related: `create` Create new parameter in a parameter tree (C)
`setenumeral` Set values of a string variable in a tree (C)

lookup Look up words and lines from a text file (C)

Syntax: `lookup(options):return1,return2,...,number_returned`

Description: Searches a text file from top to bottom for a word and returns to the user subsequent words or lines. In this context, *word* is defined as any string of characters delimited by “whitespace.” By default, *whitespace* includes the space character, a tab, a newline, a carriage return, and a comma. The whitespace characters can also be specified. Therefore, a word can be a string a digits, a string of letters, or a combination of letters and digits. Punctuation marks, unless defined as whitespace (as the comma is by default), can also form words or be part of a word. A *line* is any string of characters from the current word to the next carriage return. A line includes all whitespace characters except the carriage return. Note that word searches are case-insensitive.

Arguments: `options` is one or more of the seven keywords ('file', 'seek', 'skip', 'read', 'readline', 'count', and 'delimiter') and other arguments used as follows:

- 'file' is a keyword to specify that the next argument is the name of the text file to be searched. If the 'file' keyword is used, it *must* be the first argument and the name of the file *must* be the second argument. 'file' resets the start of a search to the top of the text file, and subsequent searches

through the file continue from where the previous search stopped, provided the 'file' keyword is not used again. Using 'file' as an argument also resets the whitespace characters back to default values.

- 'seek' is a keyword to search the text file for words that match those supplied as arguments following the 'seek' argument. When `lookup` is executed the first time, an implicit 'seek' is assumed as an argument. `lookup` maintains a pointer to the word following the last successful 'seek'. The first argument following an explicit 'seek' argument is interpreted as a word to search for, not a potential keyword. The second or later argument following an explicit 'seek' is interpreted as a keyword if it matches one of the seven `lookup` keywords. For example, you can search for the word `file` without having it interpreted as a keyword by having 'file' immediately follow the 'seek' keyword in the argument list.
- 'skip' is a keyword to move the word pointer to the next word in the text file. 'skip' can optionally be followed by a number specifying how many words to skip.
- 'read' is a keyword to return to the user the word currently being pointed to and then move the pointer to the next word. 'read' can optionally be followed by a number specifying how many words to return.
- 'readline' is a keyword to return to the user the word currently being pointed to and all the following words until the end of the current line. The pointer is then moved to the first word of the next line. 'readline' can optionally be followed by a number specifying how many lines to return.
- 'count' is a keyword to return to the user the number of times words in the text file match the subsequent argument. The count starts at the current word pointer and proceeds to the end of the file.
- 'delimiter' is a keyword to specify that the next supplied argument is a list of characters identifying the whitespace used to delimit words. Characters are specified by `\n` (newline), `\t` (tab), `\r` (carriage return), `\\` (backslash), and `\'` (single quote). The arguments 'delimiter', '`\t\n\r,`' reselect the default whitespace. The 'file' keyword also reselects the default whitespace. The distinction is that using 'file' restarts the search from the beginning of the file while using 'delimiter' continues from the current search position. Following the 'delimiter' keyword and its argument, an implicit 'seek' is assumed.

`return1, return2, ...` are words or lines returned from the search.

`number_returned` is the number of arguments returned from the file.

```
Examples: lookup('file', systemdir + '/manual/lookup')
lookup('user', 'skip', 2, 'read', 2, 'readline')
        :$n1, $n2, $n3, $ret
lookup('skip', 8, 'read', 'skip', 3, 'read', 2, 'seek',
        'comma'):$n3, $n4, $n5
lookup('delimiter', '\. \n \t', 'seek', 'file',
        'must', 'skip', 6, 'read'):$n
```

For a more detailed example of using `lookup`, see the text file `/manual/lookup` in the VNMR system directory (`systemdir`).

See also: *VNMR User Programming*

Related: `dialog` Display a dialog box from a macro (C)
`systemdir` VNMR system directory (P)

lp First-order phase in directly detected dimension (P)

Description: Specifies the first-order phase-correction angles along the directly detected dimension according to the formula

$$\text{absorption spectrum}(\omega) = \text{real channel}(\omega) * \sin \theta + \text{imaginary channel}(\omega) * \cos \theta$$

where the phase angle θ is a function of frequency, i.e.

$$\theta = \text{rp} + (\omega - \omega_0) * \text{lp}$$

ω_0 is defined to be the right end of the spectrum (i.e., `lp` has zero effect at the right edge of the spectrum and a linearly increasing effect going to the left). In multidimensional data sets, `lp` controls the phase of the directly detected dimension: f_2 dimension in 2D data sets, f_3 dimension in 3D data sets, etc.

Values: -3600 to $+3600$, in degrees. Typical values are between 0 and -180 .

See also: *Getting Started*

Related: `aph` Automatic phase adjustment of spectra (C)
`lp1` First-order phase in 1st indirectly detected dimension (P)
`lp2` First-order phase in 2nd indirectly detected dimension (P)
`rp` Zero-order phase in directly detected dimension (P)

lp1 First-order phase in 1st indirectly detected dimension (P)

Description: Controls the first-order phase constant along the first indirectly detected dimension during the process of phase-sensitive 2D transformation. The first indirectly detected dimension is often referred to as the f_1 dimension of a multidimensional data set.

See also: *User Guide: Liquids NMR*

Related: `lp` First-order phase in directly detected dimension (P)
`lp2` First-order phase in 2nd indirectly detected dimension (P)
`rp1` Zero-order phase in 1st indirectly detected dimension (P)

lp2 First-order phase in 2nd indirectly detected dimension (P)

Description: Controls the first-order phase constant along the second indirectly detected dimension during a `ds`, `dconi`, or equivalent display operation on the 2D data or a 1D trace therein. The second indirectly detected dimension is often referred to as the f_2 dimension of a 3D (or higher dimensionality) data set.

See also: *User Guide: Liquids NMR*

Related: `dconi` Interactive 2D contour display (C)
`ds` Display a spectrum (C)
`lp` First-order phase in directly detected dimension (P)
`rp2` Zero-order phase in 2nd indirectly detected dimension (P)

lpalg LP algorithm in np dimension (P)

Description: Specifies the linear prediction (LP) algorithm to use in the `np` dimension. The resulting LP coefficients are used to appropriately extend the complex time-domain data prior to a normal Fourier transform. The LP algorithms work both

on complex t_2 FIDs and on hypercomplex or complex t_1 interferograms. Enter `addpar('lp')` to create `lpalg` and other `np` dimension LP parameters in the current experiment

Values: `'lpfft'` does a least-squares calculation of `lpfilt` complex LP coefficients using `lpnupts` complex time-domain data points. Eigenvalue decomposition of the least-squares matrix is done using Householder tridiagonalization followed by the QL method with implicit shifts.

`'lparfft'` does a non-least-squares calculation of `lpfilt` complex LP coefficients using `(lpfilt+1)` complex, autoregressive (AR) matrix elements. These AR matrix elements are calculated from the raw, complex time-domain data using `lpnupts` points.

Note that the `'lpfft'` algorithm is preferred by far. While `'lparfft'` can model broad lines and can extend data sets when mostly noise exists, it cannot model narrow lines.

See also: *Getting Started*

Related:	<code>addpar</code>	Add selected parameters to the current experiment (M)
	<code>dglp</code>	Display group of linear prediction parameters (M)
	<code>lpalg1</code>	LP algorithm in <code>ni</code> dimension (P)
	<code>lpalg2</code>	LP algorithm in <code>ni2</code> dimension (P)
	<code>lpext</code>	LP data extension in <code>np</code> dimension (P)
	<code>lpfilt</code>	LP coefficients to calculate in <code>np</code> dimension (P)
	<code>lpnupts</code>	LP number of data points in <code>np</code> dimension (P)
	<code>lpopt</code>	LP algorithm data extension in <code>np</code> dimension (P)
	<code>lpprint</code>	LP print output in <code>np</code> dimension (P)
	<code>lptrace</code>	LP output spectrum in <code>np</code> dimension (P)
	<code>np</code>	Number of data points (P)
	<code>proc</code>	Type of processing on <code>np</code> FID (P)
	<code>strtlp</code>	Starting point for LP calculation in <code>np</code> dimension (P)
	<code>strtext</code>	Starting point for LP data extension in <code>np</code> dimension (P)

`lpalg1` LP algorithm in `ni` dimension (P)

Description: Specifies the LP (linear prediction) algorithm to use in the `ni` dimension. `lpalg1` functions analogously to `lpalg`. Enter `addpar('lp',1)` to create `lpalg1` and other `ni` dimension LP parameters in the current experiment.

Values: `'lpfft'` or `'lparfft'`

See also: *User Guide: Liquids NMR*

Related:	<code>addpar</code>	Add selected parameters to the current experiment (M)
	<code>lpalg</code>	LP algorithm in <code>np</code> dimension (P)
	<code>ni</code>	Number of increments in 1st indirectly detected dimension (P)

`lpalg2` LP algorithm in `ni2` dimension (P)

Description: Specifies the LP (linear prediction) algorithm to use in the `ni2` dimension. `lpalg2` functions analogously to `lpalg`. Enter `addpar('lp',2)` to create `lpalg2` and other `ni2` dimension LP parameters in the current experiment.

Values: `'lpfft'` or `'lparfft'`

See also: *User Guide: Liquids NMR*

Related:	<code>addpar</code>	Add selected parameters to the current experiment (M)
	<code>lpalg</code>	LP algorithm in <code>np</code> dimension (P)
	<code>ni2</code>	Number of increments in 2nd indirectly detected dimension (P)

lpe **Field of view size for phase-encode axis (P)**

Applicability: Systems with imaging capabilities.

Description: Specifies the actual size of the image field of view (FOV) for phase encode axis, in cm. The size and shape of the FOV is set through the selection of the parameters `sw`, `gro`, `lro`, `sw1`, `gpe`, and `lpe`. The size of the FOV in frequency units is `sw*sw1`, in terms of distance measure (in cm) is `lro*lpe`. The values of these parameters are related by the following equalities, where `gcal` is the appropriate calibration constant.

$$sw = (gcal*sfrq*1000000*gro*lro)$$

$$sw1 = (gcal*sfrq*1000000*gpe*lpe)$$

See also: *User Guide: Imaging*

Related: `gcal` Gradient calibration constant (P)
`gpe` Phase encoding gradient increment (P)
`gro` Readout gradient strength (P)
`lpe2` Field of view size for 2nd phase-encode axis (P)
`lro` Field of view parameter for read out in cm (P)
`sw` Spectral width in directly detected dimension (P)
`sw1` Spectral width in 1st indirectly detected dimension (P)

lpe2 **Field of view size for 2nd phase-encode axis (P)**

Applicability: Systems with imaging capabilities.

Description: Specifies the size of the field of view (FOV) along a second phase-encode dimension, in cm. Higher order phase-encode dimensions are found in 3D volume imaging, and Chemical Shift Imaging (CSI) experiments with two spatial dimensions.

See also: *User Guide: Imaging*

Related: `lpe` Field of view size for phase-encode axis (P)

lpext **LP data extension in np dimension (P)**

Description: Specifies number of complex time-domain data points for LP (linear prediction) in the `np` dimension by which the original data is to be extended (or altered) in either the forward or backward direction. `lpext` is constrained by $(strtext - lpext) \geq 0$ for `lpopt = 'b'` and by $(strtext + lpext - 1) \leq fn/2$ for `lpopt = 'f'`. In the `np` direction, if $(strtext - lpext) = 0$ and `lpopt = 'b'` (backwards linear prediction with calculation of the first point), `fpmult` defaults to the theoretical value of 0.5 instead of 1.0. Enter `addpar('lp')` to create `lpext` and other `np` dimension LP parameters in the current experiment.

See also: *Getting Started*

Related: `addpar` Add selected parameters to the current experiment (M)
`lpalg` LP algorithm in `np` dimension (P)
`lpext1` LP data extension in `ni` dimension (P)
`lpext2` LP data extension in `ni2` dimension (P)
`lpopt` LP algorithm data extension in `np` dimension (P)
`np` Number of data points (P)
`strtext` Starting point for LP data extension in `np` dimension (P)

lpext1 LP data extension in ni dimension (P)

Description: Specifies number of complex time-domain data points for LP (linear prediction) in the **ni** dimension by which the original data is to be extended (or altered) in either the forward or backward direction. `lpext1` functions analogously to `lpext`. Enter `addpar('lp', 1)` to create `lpext1` and other **ni** dimension LP parameters in the current experiment.

See also: *User Guide: Liquids NMR*

Related: `addpar` Add selected parameters to the current experiment (M)
`lpext` LP data extension in **np** dimension (P)
`ni` Number of increments in 1st indirectly detected dimension (P)

lpext2 LP data extension in ni2 dimension (P)

Description: Specifies number of complex time-domain data points for LP (linear prediction) in the **ni2** dimension by which the original data is to be extended (or altered) in either the forward or backward direction. `lpext2` functions analogously to `lpext`. Enter `addpar('lp', 2)` to create `lpext2` and other **ni2** dimension LP parameters in the current experiment.

See also: *User Guide: Liquids NMR*

Related: `addpar` Add selected parameters to the current experiment (M)
`lpext` LP data extension in **np** dimension (P)
`ni2` Number of increments in 2nd indirectly detected dimension (P)

lpfilt LP coefficients to calculate in np dimension (P)

Description: Specifies number of complex LP (linear prediction) coefficients in the **np** dimension to be calculated from a specified region of the time-domain data. `lpfilt` should be greater than `nsignals`, where `nsignals` is the number of sinusoidal signals contained in that FID (or interferogram). Enter `addpar('lp')` to create `lpfilt` and other **np** dimension LP parameters in the current experiment.

See also: *Getting Started*

Related: `addpar` Add selected parameters to the current experiment (M)
`lpalg` LP algorithm in **np** dimension (P)
`lpfilt1` LP coefficients to calculate in **ni** dimension (P)
`lpfilt2` LP coefficients to calculate in **ni2** dimension (P)
`np` Number of data points (P)

lpfilt1 LP coefficients to calculate in ni dimension (P)

Description: Specifies number of complex LP (linear prediction) coefficients in the **ni** dimension to be calculated from a specified region of the time-domain data. `lpfilt1` functions analogously to `lpfilt`. Enter `addpar('lp', 1)` to create `lpfilt1` and other **ni** dimension LP parameters in the current experiment.

See also: *User Guide: Liquids NMR*

Related: `addpar` Add selected parameters to the current experiment (M)
`lpfilt` LP coefficients to calculate in **np** dimension (P)
`ni` Number of increments in 1st indirectly detected dimension (P)

lpfilt2 LP coefficients to calculate in ni2 dimension (P)

Description: Specifies number of complex LP (linear prediction) coefficients in the **ni2** dimension to be calculated from a specified region of the time-domain data. `lpfilt2` functions analogously to `lpfilt`. Enter `addpar('lp',2)` to create `lpfilt1` and other **ni2** dimension LP parameters in the current experiment.

See also: *User Guide: Liquids NMR*

Related: `addpar` Add selected parameters to the current experiment (M)
`lpfilt` LP coefficients to calculate in **np** dimension (P)
`ni` Number of increments in 1st indirectly detected dimension (P)

lpnupts LP number of data points in np dimension (P)

Description: Specifies number of complex time-domain data points in the **np** dimension to be used in constructing the autoregressive (`lpalg='lparfft'`) or least-squares (`lpalg='lpnefft'`) matrix from which the complex LP (linear prediction) coefficients are calculated. Note that `lpnupts` greater than or equal to $2 * \text{lpfilt}$ is required for both algorithms. Enter `addpar('lp')` to create `lpnupts` and other **np** dimension LP parameters in the current experiment.

See also: *Getting Started*

Related: `addpar` Add selected parameters to the current experiment (M)
`lpalg` LP algorithm in **np** dimension (P)
`lpfilt` LP coefficients to calculate in **np** dimension (P)
`lpnupts1` LP number of data points in **ni** dimension (P)
`lpnupts2` LP number of data points in **ni2** dimension (P)
`np` Number of data points (P)

lpnupts1 LP number of data points in ni dimension (P)

Description: Specifies number of complex time-domain data points in the **ni** dimension to be used in constructing the autoregressive (`lpalg1='lparfft'`) or least-squares (`lpalg1='lpnefft'`) matrix from which the complex LP (linear prediction) coefficients are calculated. `lpnupts1` functions analogously to `lpnupts`. Enter `addpar('lp',1)` to create `lpnupts1` and other **ni** dimension LP parameters in the current experiment.

See also: *User Guide: Liquids NMR*

Related: `addpar` Add selected parameters to the current experiment (M)
`lpalg1` LP algorithm in **ni** dimension (P)
`lpnupts` LP number of data points in **np** dimension (P)
`ni` Number of increments in 1st indirectly detected dimension (P)

lpnupts2 LP number of data points in ni2 dimension (P)

Description: Specifies number of complex time-domain data points in the **ni2** dimension to be used in constructing the autoregressive (`lpalg2='lparfft'`) or least-squares (`lpalg2='lpnefft'`) matrix from which the complex LP (linear prediction) coefficients are calculated. `lpnupts2` functions analogously to `lpnupts`. Enter `addpar('lp',2)` to create `lpnupts2` and other **ni2** dimension LP parameters in the current experiment.

See also: *User Guide: Liquids NMR*

Related: **addpar** Add selected parameters to the current experiment (M)
lpalg2 LP algorithm in **ni2** dimension (P)
lpnupts LP number of data points in **np** dimension (P)
ni2 Number of increments in 2nd indirectly detected dimension (P)

lpopt LP algorithm data extension in **np** dimension (P)

Description: Specifies how the specific LP (linear prediction) algorithm is to extend (or alter) forward or backward the time-domain data in the **np** dimension. Enter **addpar** (' lp ') to create **lpopt** and other **np** dimension LP parameters in the current experiment.

Multiple LP operations, extended forward or backward, can be performed on each FID or interferogram. This is accomplished by arraying the LP processing parameters (e.g., **lpopt**= ' b ' , ' f ' , ' b '). The number of LP operations is determined by the LP processing parameter with the largest array size. LP parameters having a smaller array size are padded out with their last value. The most common use for this capability is to back-calculate the first 1 to 2 points in an FID or interferogram and subsequently to extend the length of the time-domain data by LP.

A printout can be obtained for each LP operation on an individually definable FID or interferogram. For example, if **lpprint**=30, 30 and **lptrace**=1, 2, the text file **lpanalyz.out.1** contains the LP printout for the first LP operation on FID 1 and **lpanalyz.out.2** contains the LP printout for the second LP operation on FID 2.

Values: ' b ' indicates the LP coefficients are to be used in the back-calculation of a specified number of time-domain data points.

' f ' indicates the LP coefficients are to be used in the forward extension of the time-domain data by a specified number of points. The characteristic polynomial in **z** space, derived from the complex LP coefficients, is set up and rooted. Any root found to lie outside the unit circle is reflected back into the unit circle. New complex LP coefficients are then calculated from these adjusted complex roots.

See also: *Getting Started*

Related: **addpar** Add selected parameters to the current experiment (M)
lpalg LP algorithm in **np** dimension (P)
lpopt1 LP algorithm data extension for **ni** dimension (P)
lpopt2 LP algorithm data extension for **ni2** dimension (P)
lpprint LP print output for **np** dimension (P)
lptrace LP output spectrum for **np** dimension (P)
np Number of data points (P)

lpopt1 LP algorithm data extension in **ni** dimension (P)

Description: Specifies how the specific LP (linear prediction) algorithm is to extend (or alter) forward or backward the time-domain data in the **ni** dimension. **lpopt1** functions analogously to **lpopt**. Enter **addpar** (' lp ' , 1) to create **lpopt1** and other **ni** dimension LP parameters in the current experiment.

See also: *User Guide: Liquids NMR*

Related: **addpar** Add selected parameters to the current experiment (M)
lpopt LP algorithm data extension for **np** dimension (P)
ni Number of increments in 1st indirectly detected dimension (P)

lpopt2 LP algorithm data extension in ni2 dimension (P)

Description: Specifies how the specific LP (linear prediction) algorithm is to extend (or alter) forward or backward the time-domain data in the **ni2** dimension. **lpopt2** functions analogously to **lpopt**. Enter **addpar** (' lp ' , 2) to create **lpopt2** and other **ni2** dimension LP parameters in the current experiment.

See also: *User Guide: Liquids NMR*

Related: **addpar** Add selected parameters to the current experiment (M)
lpopt LP algorithm data extension for **np** dimension (P)
ni2 Number of increments in 2nd indirectly detected dimension (P)

lpprint LP print output for np dimension (P)

Description: Controls LP (linear prediction) print output for the **np** dimension and creates an output file in the current experiment directory (**curexp**) with the name **lpanalyz.out.1**. Enter **addpar** (' lp ') to create **lpprint** and other **np** dimension LP parameters in the current experiment.

Values: Comprised of sum of decimal values of the following bit fields, in which each bit field controls an independent output option:

- Bit 0 (decimal value 1) writes out the LP matrix and Y vector from which the LP coefficients are calculated.
- Bit 1 (decimal value 2) writes out the LP coefficients that have been obtained using either of the two supported algorithms.
- Bit 2 (decimal value 4) writes out the LP roots obtained from the characteristic polynomial derived from the LP coefficients; this only applies for **lpalg**= ' lpfilt ' and **lpopt**= ' f '.
- Bit 3 (decimal value 8) writes out the original and recalculated values for each LP extended (or altered) complex time-domain data point.
- Bit 4 (decimal value 16) writes out the internal LP parameter structure.

For example, **lpprint**=12 and **lptrace**=1 yields the following information in the file **curexp/lpanalyz.out.1** for spectrum 1 along f_2 : the values for all **lpfilt** complex LP coefficients and the original and recalculated values for each of the **lpext** LP extended (or altered) complex time-domain data points.

See also: *Getting Started*

Related: **addpar** Add selected parameters to the current experiment (M)
curexp Current experiment directory (P)
lpalg LP algorithm in **np** dimension (P)
lpext LP data extension in **np** dimension (P)
lpfilt LP coefficients to calculate in **np** dimension (P)
lpopt LP algorithm data extension for **np** dimension (P)
lpprint1 LP print output for **ni** dimension (P)
lpprint2 LP print output for **ni2** dimension (P)
lptrace LP output spectrum in **np** dimension (P)
np Number of data points (P)

lpprint1 LP print output for ni dimension (P)

Description: Controls LP (linear prediction) print output for the **ni** dimension and creates an output file in the current experiment directory (**curexp**) with the name **lpanalyz1.out.1**. **lpprint1** functions analogously to **lpprint**. Enter

`addpar('lp', 1)` to create `lpprint1` and other `ni` dimension LP parameters in the current experiment.

See also: *User Guide: Liquids NMR*

Related: `addpar` Add selected parameters to the current experiment (M)
`lpprint` LP print output for `np` dimension (P)
`ni` Number of increments in 1st indirectly detected dimension (P)

lpprint2 LP print output for ni2 dimension (P)

Description: Controls LP (linear prediction) print output for the `ni2` dimension and creates an output file in the current experiment directory (`curexp`) with the name `lpanalyz2.out.1`. `lpprint2` functions analogously to `lpprint`. Enter `addpar('lp', 2)` to create `lpprint2` and other `ni2` dimension LP parameters in the current experiment.

See also: *User Guide: Liquids NMR*

Related: `addpar` Add selected parameters to the current experiment (M)
`lpprint` LP print output for `np` dimension (P)
`ni2` Number of increments in 2nd indirectly detected dimension (P)

lptrace LP output spectrum in np dimension (P)

Description: Specifies for which spectrum LP (linear prediction) output in the `np` dimension is produced in accordance with the parameter `lpprint`. Enter `addpar('lp')` to create `lptrace` and other `np` dimension LP parameters in the current experiment.

See also: *Getting Started*

Related: `addpar` Add selected parameters to the current experiment (M)
`lpalg` LP algorithm in `np` dimension (P)
`lpprint` LP print output in `np` dimension (P)
`lptrace1` LP output spectrum in `ni` dimension (P)
`lptrace2` LP output spectrum in `ni2` dimension (P)
`np` Number of data points (P)

lptrace1 LP output spectrum in ni dimension (P)

Description: Specifies for which spectrum or trace LP (linear prediction) output in the `ni` dimension is produced in accordance with the parameter `lpprint1`. `lptrace1` functions analogously to `lptrace`. Enter `addpar('lp', 1)` to create `lpprint2` and other `ni` dimension LP parameters in the current experiment.

See also: *User Guide: Liquids NMR*

Related: `addpar` Add selected parameters to the current experiment (M)
`lpprint1` LP print output in `ni` dimension (P)
`lptrace` LP output spectrum in `np` dimension (P)
`ni` Number of increments in 1st indirectly detected dimension (P)

lptrace2 LP output spectrum in ni2 dimension (P)

Description: Specifies for which spectrum or trace LP (linear prediction) output in the `ni2` dimension is produced in accordance with the parameter `lpprint2`. `lptrace2` functions analogously to `lptrace`. Enter `addpar('lp', 2)` to create `lptrace2` and other `ni2` dimension LP parameters in the current experiment.

See also: *User Guide: Liquids NMR*

Related: **addpar** Add selected parameters to the current experiment (M)
lpprint2 LP print output in **ni2** dimension (P)
lptrace LP output spectrum in **np** dimension (P)
ni2 Number of increments in 2nd indirectly detected dimension (P)

lro Field of view size for readout axis (P)

Applicability: Systems with imaging capabilities.

Description: Specifies the actual size of the image field of view (FOV) for readout axis, in cm. The size and shape of the image FOV is set through the selection of the parameters **sw**, **gro**, **lro**, **sw1**, **gpe**, and **lpe**. The size of the FOV in frequency units is **sw*sw1**, or in terms of distance measure (cm) is **lro*lpe**. The values of these parameters are related by the following equalities, where **gcal** is the appropriate calibration constant:

$$\begin{aligned} \text{sw} &= (\text{gcal} * \text{sfrq} * 1000000 * \text{gro} * \text{lro}) \\ \text{sw1} &= (\text{gcal} * \text{sfrq} * 1000000 * \text{gpe} * \text{lpe}) \end{aligned}$$

See also: *User Guide: Imaging*

Related: **gcal** Gradient calibration constant (P)
gpe Phase encoding gradient increment (P)
gro Readout gradient strength (P)
lpe Field of view size for phase encode axis (P)
sw Spectral width in directly detected dimension (P)
sw1 Spectral width in 1st indirectly detected dimension (P)

ls List files in directory (C)

Syntax: `ls<(directory)>`

Description: Lists the names of files in a directory on the text output window. **ls** is identical to **dir** and **lf**.

Arguments: **directory** is the name of a directory. The default is the current working directory. **ls** is equivalent to the UNIX command **ls** and uses the same options (e.g., **-l** for a long listing such as `ls('-l * .fid')`).

Examples: `ls`
`ls('data')`
`ls('-l * .fid')`

See also: *Getting Started*

Related: **dir** List files in directory (C)
lf List files in directory (C)

lsfid Number of complex points to left-shift the **np** FID (P)

Description: Specifies number of complex points (not real points) that the **np** FID is to be either left-shifted (**lsfid**>0) or right-shifted (**lsfid**<0). A right shift adds zeros to the front of the FID. **lsfid** (and related parameters **phfid** and **lsfrq**) operate on complex **np** FID data, referred to as the **t₂** dimension in a 2D experiment or as the **t₃** dimension in a 3D experiment. **lsfid** is in the processing group and is properly handled by a **wti** operation (display).

Values: **-fn/2** to **np/2** (or **-fn/2** to **fn/2** if **fn**<**np**), 'n'

See also: *Getting Started*

Related:	<code>dfid</code>	Display a single FID (C)
	<code>ds</code>	Display a spectrum FID (C)
	<code>fn</code>	Fourier number in directly detected dimension (P)
	<code>ft</code>	Fourier transform 1D data (C)
	<code>ft1d</code>	Fourier transform along f_2 dimension (C)
	<code>ft2d</code>	Fourier transform 2D data (C)
	<code>lsfid1</code>	Number of complex points to left-shift ni interferogram(P)
	<code>lsfid2</code>	Number of complex points to left-shift $ni2$ interferogram (P)
	<code>lsfrq</code>	Frequency shift of the fn spectrum in Hz (P)
	<code>np</code>	Number of data points (P)
	<code>phfid</code>	Zero-order phasing constant for the np FID (P)
	<code>wft</code>	Weight and Fourier transform 1D data (C)
	<code>wft1d</code>	Weight and Fourier transform f_2 of 2D data (C)
	<code>wft2d</code>	Weight and Fourier transform 2D data (C)
	<code>wti</code>	Interactive weighting (C)

lsfid1 **Number of complex points to left-shift ni interferogram (P)**

Description: Specifies number of hypercomplex (for hypercomplex interferogram data) or complex (for complex interferogram data) points that the ni interferogram is to be either left-shifted ($lsfid1 > 0$) or right-shifted ($lsfid1 < 0$). A right shift adds zeros to the front of the FID. $lsfid1$ (and related parameters `phfid1` and `lsfrq1`) operate on ni interferogram data, both hypercomplex and complex. ni interferogram data are referred to as the t_1 dimension in both a 2D and a 3D experiment. $lsfid1$ is in the processing group and is properly handled by a `wti` operation (display); that is, a `wti` operation on an ni interferogram applies the parameters `phfid1`, $lsfid1$, and `lsfrq1`, if selected, to the time-domain data prior to the Fourier transformation.

Values: $-fn1/2$ to ni (or $-fn1/2$ to $fn1/2$ if $fn1 < 2 * ni$), 'n'

See also: *User Guide: Liquids NMR*

Related:	<code>fn1</code>	Fourier number in 1st indirectly detected dimension (P)
	<code>lsfid</code>	Number of complex points to left-shift np FID (P)
	<code>lsfid2</code>	Number of complex points to left-shift $ni2$ interferogram (P)
	<code>lsfrq1</code>	Frequency shift of the $fn1$ spectrum in Hz (P)
	<code>ni</code>	Number of increments in 1st indirectly detected dimension (P)
	<code>phfid1</code>	Zero-order phasing constant for ni interferogram (P)
	<code>wti</code>	Interactive weighting (C)

lsfid2 **Number of complex points to left-shift $ni2$ interferogram (P)**

Description: Specifies the number of hypercomplex (for hypercomplex interferogram data) or complex (for complex interferogram data) points that the $ni2$ interferogram is to be either left-shifted ($lsfid2 > 0$) or right-shifted ($lsfid2 < 0$). A right shift adds zeros to the front of the FID. $lsfid2$ (and related parameters `phfid2` and `lsfrq2`) operate on $ni2$ interferogram data, both hypercomplex and complex. $ni2$ interferogram data are referred to as the t_2 dimension in a 3D experiment. $lsfid2$ is in the processing group and is properly handled by a `wti` operation (display).

Values: $-fn2/2$ to $ni2$ (or $-fn2/2$ to $fn2/2$ if $fn2 < 2 * ni2$), 'n'

See also: *User Guide: Liquids NMR*

Related:	<code>fn2</code>	Fourier number in 2nd indirectly detected dimension (P)
	<code>lsfid</code>	Number of complex points to left-shift np FID (P)

<code>lsfid1</code>	Number of complex points to left-shift <code>ni</code> interferogram(P)
<code>lsfrq2</code>	Frequency shift of the <code>fn2</code> spectrum in Hz (P)
<code>ni2</code>	Number of increments in 2nd indirectly detected dimension (P)
<code>phfid2</code>	Zero-order phasing constant for <code>ni2</code> interferogram (P)
<code>wti</code>	Interactive weighting (C)

`lsfrq` Frequency shift of the `fn` spectrum (P)

Description: Sets a frequency shift of spectral data, in Hz. `lsfrq` is the time-domain equivalent of `lp` within VNMR. `lsfrq` (and related parameters `phfid` and `lsfid`) operate on complex `np` FID data, referred to as the t_2 dimension in a 2D experiment or as the t_3 dimension in a 3D experiment. `lsfrq` is in the processing group and is properly handled by a `wti` operation (display).

Values: A positive value results in peaks being shifted downfield (to the left). A negative value results in peaks being shifted upfield (to the right).

See also: *Getting Started*

Related:	<code>dfid</code>	Display a single FID (C)
	<code>ds</code>	Display a spectrum FID (C)
	<code>fn</code>	Fourier number in directly detected dimension (P)
	<code>ft</code>	Fourier transform 1D data (C)
	<code>ft1d</code>	Fourier transform along f_2 dimension (C)
	<code>ft2d</code>	Fourier transform 2D data (C)
	<code>lp</code>	First-order phase in directly detected dimension (P)
	<code>lsfid</code>	Number of complex points to left-shift <code>np</code> FID (P)
	<code>lsfrq1</code>	Frequency shift of the <code>fn1</code> spectrum in Hz (P)
	<code>lsfrq2</code>	Frequency shift of the <code>fn2</code> spectrum in Hz (P)
	<code>phfid</code>	Zero-order phasing constant for <code>np</code> FID (P)
	<code>wft</code>	Weight and Fourier transform 1D data (C)
	<code>wft1d</code>	Weight and Fourier transform f_2 of 2D data (C)
	<code>wft2d</code>	Weight and Fourier transform 2D data (C)
	<code>wti</code>	Interactive weighting (C)

`lsfrq1` Frequency shift of the `fn1` spectrum (P)

Description: Sets a frequency shift of spectral data, in Hz. `lsfrq1` is the time-domain equivalent of `lp1` within VNMR. `lsfrq1` (and related parameters `phfid1` and `lsfid1`) operate on `ni` interferogram data, both hypercomplex and complex. `ni` interferogram data are referred to as the t_1 dimension in both a 2D and a 3D experiment. `lsfrq1` is in the processing group and is properly handled by a `wti` operation (display); that is, a `wti` operation on an `ni` interferogram applies the parameters `phfid1`, `lsfid1`, and `lsfrq1`, if selected, to the time-domain data prior to the Fourier transformation.

Values: A positive value results in peaks being shifted downfield (to the left).
A negative value results in peaks being shifted upfield (to the right).

See also: *User Guide: Liquids NMR*

Related:	<code>fn1</code>	Fourier number in 1st indirectly detected dimension (P)
	<code>lp1</code>	First-order phase in 1st indirectly detected dimension (P)
	<code>lsfid1</code>	Number of complex points to left-shift <code>ni</code> interferogram(P)
	<code>lsfrq</code>	Frequency shift of the <code>fn</code> spectrum in Hz (P)
	<code>lsfrq2</code>	Frequency shift of the <code>fn2</code> spectrum in Hz (P)
	<code>ni</code>	Number of increments in 1st indirectly detected dimension (P)
	<code>phfid1</code>	Zero-order phasing constant for <code>ni</code> interferogram (P)
	<code>wti</code>	Interactive weighting (C)

lsfrq2 **Frequency shift of the fn2 spectrum (P)**

Description: Sets a frequency shift of spectral data in Hz. `lsfrq2` is the time-domain equivalent of `lp2` within VNMR. `lsfrq2` (and related parameters `phfid2` and `lsfid2`) operate on `ni2` interferogram data, both hypercomplex and complex. `ni2` interferogram data is referred to as the t_2 dimension in a 3D experiment. `lsfrq2` is in the processing group and is properly handled by a `wti` operation (display).

Values: A positive value results in peaks being shifted downfield (to the left).
A negative value results in peaks being shifted upfield (to the right).

See also: *User Guide: Liquids NMR*

Related: `fn2` Fourier number in 2nd indirectly detected dimension (P)
`lp2` First-order phase in 2nd indirectly detected dimension (P)
`lsfid1` Number of complex points to left-shift `ni` interferogram (P)
`lsfid2` Number of complex points to left-shift `ni2` interferogram (P)
`lsfrq` Frequency shift of the `fn` spectrum in Hz (P)
`ni2` Number of increments in 2nd indirectly detected dimension (P)
`phfid2` Zero-order phasing constant for `ni2` interferogram (P)
`wti` Interactive weighting (C)

lv1 **Zero-order baseline correction (P)**

Description: When spectral display is active, the command `dc` turns on a linear drift correction (baseline correction). The result of this operation includes calculating a zero-order baseline correction parameter `lv1`. This is done by averaging of a small number of points at either end of the display and drawing a straight line baseline between them.

See also: *Getting Started*

Related: `cdc` Cancel drift correction (C)
`dc` Calculate spectral drift correction (C)
`lv1tlt` Control sensitivity of `lv1` and `tlt` adjustments (P)
`tlt` First-order baseline correction (P)

lv1tlt **Control sensitivity of lv1 and tlt adjustments (P)**

Description: Controls the sensitivity of the interactive `lv1` and `tlt` adjustments. `lv1tlt` is in the “current” parameter set and is basically a multiplier for the sensitivity. If this parameter does not exist, it can be created by commands `create('lv1tlt')` `setgroup('lv1tlt', 'display')`.

Values: The default value is 1.0. Larger values make the adjustments larger. Smaller values make the adjustments smaller.

See also: *Getting Started*

Related: `create` Create new parameter in a parameter tree (C)
`ds` Display a spectrum (C)
`lv1` Zero-order baseline correction (P)
`setgroup` Set group of a variable in a tree (C)
`tlt` First-order baseline correction (P)

M

maclibpath **Path to user's macro directory (P)**

Description: Contains an absolute path to a user's macro files directory. If `maclibpath` exists for a user, it must be defined in the global parameter file for the user. Enter the command `create('maclibpath','string','global')` to create `maclibpath`.

See also: *VNMR User Programming*

Related: `create` Create new parameter in a parameter tree (C)
`exists` Determine if a parameter, file, or macro exists (C)

macro **Macro name (P)**

Description: A string parameter, available in each experiment, similar to the `n1`, `n2`, and `n3` parameters. Certain macros, such as `h1p`, need to know which macro invoked them. This parameter is used to pass that information.

See also: *VNMR User Programming*

Related: `h1p` Process simple proton spectra from h1 macro (M)
`n1,n2,n3` Name storage for macros (P)

macrocat **Display a user macro file in text window (C)**

Syntax: `macrocat(file1<,file2><,...>)`

Description: Displays one or more user macro files in the text window.

Arguments: `file1`, `file2`, ... are the names of macros in the user macro library.

Examples: `macrocat('build')`
`macrocat('dan','george')`

See also: *VNMR User Programming*

Related: `macrodir` List user macros (C)
`macrosyscat` Display a system macro file in text window (C)

macrocp **Copy a user macro file (C)**

Syntax: `macrocp(from_file,to_file)`

Description: Makes a copy of the existing user macro file and places the copy in the user's macro library. Using `macrocp` to make a backup copy is the recommended procedure to modify a macro but still be able to revert to the previous version if you are unsure about the modification. `macrocp` can also be useful for writing a new macro that is very similar to an existing macro.

Arguments: `from_file` is the name of an existing user macro file to be copied. The file must be in the user's macro library.

`to_file` is the file name to be given to the copy. This name must be different from the name of the original macro.

Examples: `macrocp('dan','dan.old')`

See also: *VNMR User Programming*

Related: **macrocat** Display a user macro file in text window (C)
macrodir List user macros (C)
macrosyscp Copy a system macro to become a user macro (C)

macrodir List user macro files (C)

Syntax: `macrodir`

Description: Lists the names of user macro files in the user's macro library.

See also: *VNMR User Programming*

Related: **macrosysdir** Lists system macros (C)

macroedit Edit a macro with user-selectable editor (M)

Syntax: `macroedit (file)`

Description: Opens a MAGICAL macro file from a user's personal macro library for editing (if you want to edit a system macro, copy it to a personal library and then use `macroedit`).

The default editor is `vi`. To select another editor, first set UNIX environmental variable `vnmreditor` to the name of the editor; that is, in the `.login` file, change the line

```
setenv vnmreditor old_ed
```

to become

```
setenv vnmreditor new_ed (e.g., setenv vnmreditor emacs).
```

Second, make sure a script with the prefix `vnmr_` followed by the name of the editor is placed in the `bin` subdirectory of the VNMR system directory (e.g., `vnmr_emacs`).

The script file makes adjustments for the type of graphic interface in use. Scripts provided in the software include `vnmr_vi` and `vnmr_textedit`. To create other scripts, refer to the `vnmr_vi` script for non-window editor interfaces or refer to `vnmr_textedit` for window-based editor interfaces.

Arguments: `file` is the name of the macro file you wish to edit.

Examples: `macroedit ('pa')`

See also: *VNMR User Programming*

Related: **paramedit** Edit a parameter and its attributes with user-selected editor (C)
paramvi Edit a parameter and its attributes with `vi` editor (M)
edit Edit a file with user-selectable editor (C)
macrovi Edit a user macro with `vi` editor (M)
menuvi Edit a menu with the `vi` editor (M)
textvi Edit text file of current experiment with `vi` editor (M)

macrold Load a macro into memory (C)

Syntax: `macrold (file) <:dummy>`

Description: Loads a macro, user or system, into memory. If the macro already exists in memory, it is overwritten by the new macro. Loading a macro into memory increases the execution speed of the macro. The trade-off is that the macro uses memory. The `mstat` command displays macros that have been loaded into memory. One or more individual macros, or all the macros loaded in memory, can be removed from memory with the `purge` command.

If a macro already loaded into memory is edited using `macrovi` or `macroedit`, the changed macro automatically is loaded by those macros. This overwrites the previous macro. However, if a macro is edited or created some other way (with `macrocp` perhaps), the changed version is not automatically loaded. If the macro already exists in memory, the previous version executes unless the user runs `macrold`.

Arguments: `file` is the name of the macro file to be loaded into memory. For loading macros, the same search path is used as when deciding which macro to execute. That is, the user's private `maclib` directory is searched first, then a directory specified by `maclibpath`, and finally the system `maclib`. If an absolute path is supplied as the `file` argument, that macro is loaded. This allows macros not in a `maclib` to be loaded and executed from VNMR.

`dummy` is any throwaway variable. Requesting a return value suppresses the message in the status window (line 3) that the macro is loaded.

Examples: `macrold('pa')`
`macrold('_sw'):$noline3`

See also: *VNMR User Programming*

Related:

<code>maclibpath</code>	Path to user's macro directory (P)
<code>macrocp</code>	Copy a user macro file (C)
<code>macroedit</code>	Edit a macro with user-selectable editor (M)
<code>macrovi</code>	Edit a user macro with the vi text editor (M)
<code>mstat</code>	Display memory usage statistics (C)
<code>purge</code>	Remove macros from memory (C)

macrorm Remove a user macro (C)

Syntax: `macrorm(file)`

Description: Removes a user macro from the user's macro directory. If the macro has already been loaded in memory, it remains in memory until a new macro of the same name is loaded or the program exits.

Arguments: `file` is the name of the user macro to be removed.

Examples: `macrorm('pa')`

See also: *VNMR User Programming*

Related:

<code>delcom</code>	Delete a user macro (M)
<code>macrodir</code>	List user macros (C)
<code>macrosysrm</code>	Remove a system macro (C)
<code>purge</code>	Remove all macros from memory (C)

macrosyscat Display a system macro file in text window (C)

Syntax: `macrosyscat(file1<,file2><,...>)`

Description: Displays one or more system macro files in the text window.

Arguments: `file1`, `file2`, ... are names of macros in the system macro library.

Examples: `macrosyscat('build')`
`macrosyscat('dan','george')`

See also: *VNMR User Programming*

Related:

<code>macrocat</code>	Display a user macro file in text window (C)
<code>macrosysdir</code>	Lists system macros (C)

macrosyscp Copy a system macro to become a user macro (C)

Syntax: `macrosyscp(from_file, to_file)`

Description: Makes a copy of the existing system macro file and places the copy in the user's macro library. This is the recommended way to modify a system macro for personal use.

Arguments: `from_file` is the name of an existing system macro file to be copied. The file must be in the system macro library.

`to_file` is the file name to be given to the copy. In this case, the name of the copied macro can be the same as the original macro. In many cases, it is the same, allowing the user to have a personal macro of the same name as the system macro but which will override the system macro.

Examples: `macrosyscp('pa', 'pa')`
`macrosyscp('pa', 'mypa')`

See also: *VNMR User Programming*

Related: `macrocp` Copy a user macro file (C)
`macrosyscat` Display a system macro file in text window (C)
`macrosysdir` Lists system macros (C)

macrosysdir List system macros (C)

Syntax: `macrosysdir`

Description: Lists the names of system macros in the system macro library.

See also: *VNMR User Programming*

Related: `macrodir` List user macros (C)

macrosysld Load a system macro into memory (obsolete)

Description: This command is no longer part of VNMR. It has been replaced by the `macrold` command, which has been changed to load both user and system macros into memory.

Related: `macrold` Load a user macro into memory (C)

macrosysrm Remove a system macro (C)

Syntax: `macrosysrm(file)`

Description: Removes a system macro file from the system macro directory. If the macro has already been loaded in memory, it remains in memory until a new macro of the same name is loaded or the program exits.

Arguments: `file` is the name of the system macro file to be removed.

Examples: `macrosysrm('pa')`

See also: *VNMR User Programming*

Related: `macrorm` Remove a user macro (C)
`macrosysdir` Lists system macros (C)
`purge` Remove all macros from memory (C)

macrosysvi **Edit a system macro with the vi text editor (obsolete)**

Description: This macro is no longer part of VNMR. To edit a system macro, first copy it to a personal library, and then edit it using **macroedit** or **macrovi**.

Related: **macroedit** Edit a macro with a user-selectable editor (M)
macrovi Edit a user macro with the *vi* text editor (M)

macrovi **Edit a user macro with the vi text editor (M)**

Syntax: `macrovi (file)`

Description: Initiates creating a new user macro or modifying an existing user macro using the UNIX *vi* text editor. On the Sun workstation, a pop-up window contains the edit. On the GraphOn, the edit is done on the entire terminal. To edit a system macro, first copy the macro to a personal library and then edit it using **macroedit** or **macrovi**.

Arguments: `file` is the name of an existing user's macro to be edited or the name of a new user's macro to be created.

Examples: `macrovi ('pa')`

See also: *VNMR User Programming*

Related: **macroedit** Edit a macro with a user-selectable editor (C)
vi Edit text file with *vi* text editor (C)

make3dcoef **Make a 3D coefficients file from 2D coefficients (M)**

Syntax: `make3dcoef<('t1t2' | 't2t1')>`

Description: Makes a 3D coefficients file from 2D coefficients and writes the file in the path stored by **curexp**. 2D coefficients are supplied as strings in the parameters **f2coef** and **f1coef**. This macro is capable of handling 3D data collected with any number of data sets (e.g., TPPI, Hypercomplex, Rance SE, Kay SE, and phase-sensitive gradient in one or both dimensions). **make3dcoef** is called by the **ft3d** macro.

The 2D coefficients are supplied as strings in **f1coef** and **f2coef**. These coefficients are the same as found by processing with **wft2d(2dcoefs)**. Note that **wft2da** (for States-Hypercomplex method) is equivalent to **wft2d(1,0,0,0,0,0,-1,0)**, and that **wft2d** (for absolute-value mode) is equivalent to **wft2d(1,0,0,-1)**.

Coefficients are separated by spaces and not commas. For example, if a 3D data set collected by the States-Hypercomplex method in both **ni** and **ni2** dimensions, **f1coef**='1 0 0 0 0 0 -1 0' and **f2coef**='1 0 0 0 0 0 -1 0'. And if a 3D data set collected in absolute-value mode in both **ni** and **ni2** dimensions, **f1coef**='1 0 0 -1' and **f2coef**='1 0 0 -1'.

The **f1coef** and **f2coef** parameters are created by the **par3d** macro. Execution of **make3dcoef** when **f1coef** and **f2coef** have no value or inconsistent values causes the macro to abort, which enables the user to enter these values and reexecute the macro. For example, the value of **f1coef** when the F1 dimension can be processed with **wft2da** is '1 0 0 0 0 0 -1 0'. The value of **f2coef** when the F2 dimension can be processed with **wft2d(1,0,1,0,0,-1,0,1)** is '1 0 1 0 0 -1 0 1'.

The parameters **f1coef** and **f2coef** must be 2D coefficients that give proper **ni** and **ni2** first planes with the same **rp** (assuming **lp** is 0 by using **calfa**) values. For example, processing the phase-sensitive gradient dimension should not be done with 1 0 0 1 0 1 1 0 and applying 45° phase shifts to **rp**, but with

1 0 1 0 0 1 0 -1, or its variant, that gives the same **rp** value as the other dimension. This also applies to Rance-type or Kay-type sensitivity-enhanced dimensions.

Note that sensitivity-enhanced sequences (gradient or otherwise) can be processed two different ways to give “orthogonal” data sets. The coefficients must be picked so that they have the same **rp** as the other dimension.

This macro can also handle coefficients that are not 1s or 0s. For example, if processing requires that a data set contributes to the interferogram after a 30° phase shift, $\cos(30)$ and $\sin(30)$ can be selected as the real and imaginary contributions, respectively, during the construction of the interferogram.

Arguments: 't1t2' means **array**= 'phase, phase2' in simple hypercomplex data sets. It means **array**= 't1related', 't2related' with multiple sets in general.

't2t1' means **array**= 'phase2, phase' in simple hypercomplex data sets. It means **array**= 't2related', 't1related' with multiple sets in general.

If no argument is used and if **array**= 'phase, phase2' or **array**= 'phase2, phase', the macro automatically decides on 't1t2' or 't2t1', respectively.

See also: *User Guide: Liquids NMR*

Related:	array	Parameter order and precedence (P)
	calfa	Recalculate alfa so that first-order phase is zero (M)
	curexp	Current experiment directory (P)
	f1coef	Coefficient to construct F1 interferogram (P)
	f2coef	Coefficient to construct F2 interferogram (P)
	ft3d	Perform a 3D Fourier transform on a 3D FID data set (M)
	lp	First-order phase in directly detected dimension (P)
	ni	Number of increments in 1st indirectly detected dimension (P)
	ni2	Number of increments in 2nd indirectly detected dimension (P)
	ntype3d	Specify whether f_1 or f_2 display expected to be N-type (P)
	rp	Zero-order phase in directly detected dimension (P)
	wft2d	Weight and Fourier transform 2D data (C)
	wft2da	Weight and Fourier transform phase-sensitive data (M)

makedosyparams Create parameters for DOSY processing (M)

Syntax: `makedosyparams(dosytimecubed, dosyfreq)`

Description: This macro is automatically called by the `Dbppste`, `DgcsteSL`, `Doneshot`, `Dbppsteinept`, `Dgcstecosy`, and `Dgcstehmqc` sequences to create the parameters **dosyfreq**, **dosygamma**, and **dosytimecubed**, which are necessary for the **dosy** analysis. Do not manually run `makedosyparams`.

See also: *User Guide: Liquids NMR*

Related:	dosy	Process DOSY experiments (M)
	dosyfreq	Larmor frequency of phase encoded nucleus in DOSY (P)
	dosygamma	Gyromagnetic constant of phase encoded nucleus in DOSY (P)
	dosytimecubed	Gyromagnetic constant of phase encoded nucleus in DOSY (P)

makefid Make a FID element using numeric text input (C)

Syntax: `makefid(file<, element_number<, format>)`

Description: Creates FID files that can be used to introduce computed data into an experiment. The number of points comes from the number of numeric values

read from the input file. If the current experiment already contains a FID, you will not be able to change either the format or the number of points from that present in the FID file. Use `rm(curexp+'acqfil/fid')` to remove the FID.

The `makefid` command does not look at parameter values when establishing the format of the data or the number of points in an element. Thus, if the FID file is not present, it is possible for `makefid` to write a FID file with a header that does not match the value of `dp` or `np`. Because the active value is in the processed tree, you need to use the `setvalue` command if any changes are required.

Arguments: `file` is the name of the input file. It contains numeric values, two per line. The first value is assigned to the X (or real) channel; the second value on the line is assigned to the Y (or imaginary) channel.

`element_number` is the number of the element or FID and is any integer larger than 0. The default is the first element or FID. If the FID element already exists in the FID file, the program overwrites the old data.

`format` is a character string with the precision of the resulting FID file and can be specified by one of the following strings:

'dp=n'	single-precision (16-bit) data
'dp=y'	double-precision (32-bit) data
'16-bit'	single-precision (16-bit) data
'32-bit'	double-precision (32-bit) data

If an FID file exists, `makefid` uses the same `format` string for precision; otherwise, the default is double-precision (32-bit) data.

`element_number` and `format` arguments can be entered in any order.

Examples: `makfid('fid.in',2,'32-bit')`

See also: *Getting Started; VNMR User Programming*

Related:

<code>cp</code>	Copy a file (C)
<code>curexp</code>	Current experiment directory
<code>dp</code>	Double precision (P)
<code>mv</code>	Move and/or rename a file (C)
<code>np</code>	Number of data points (P)
<code>rm</code>	Delete file (C)
<code>setvalue</code>	Set value of any parameter in a tree (C)
<code>writfid</code>	Write numeric text file using a FID element (C)

makephf Transform and save images as phasefiles (M)

Applicability: Systems with imaging capabilities.

Syntax: `makephf`

Description: Transforms and saves images as phasefiles.

See also: *User Guide: Imaging*

Related:

<code>imcalc</code>	Calculate 2D phasefiles (M,U)
<code>imfit</code>	Fit arrayed imaging data to T_1 or T_2 exponential data (M,U)

makeslice Synthesize 2D projection of 3D DOSY experiment (C)

Syntax: `makeslice(<option>,lowerlimit,upperlimit)`

Arguments: `option` is either 'i' or 's'.

'i' includes the “tails” of diffusion peaks that lie outside the range between `lowerlimit` and `upperlimit`. The default is 'i'.

's' only includes the integration peaks whose diffusion coefficient lies between the specified limits.

`lowerlimit` is the lower diffusion limit (in units of 10^{-10} m²/s) to be displayed.

`upperlimit` is the upper diffusion limit (in units of 10^{-10} m²/s) to be displayed

Description: Synthesizes an integral projection between specified diffusion limits of a 3D DOSY spectrum onto the frequency-frequency plane. `makeslice` requires the first 2D increment of the 3D DOSY data to have been transformed.

See also: *User Guide: Liquids NMR*

Related: `dosy` Process DOSY experiments (M)
`showoriginal` Restore first 2D spectrum in 3D DOSY spectrum (M)

man Display online description of command or macro (M)

Syntax: `man(file)`

Description: Displays in the text window a description of commands and system macros from files in the directory `/vnmr/manual`.

Arguments: `file` is the name of a command or system macro in `/vnmr/manual`.

Examples: `man('mark')`

See also: *Getting Started; VNMR User Programming*

Related: `manvi` Edit online description of a command or macro (M)
`manualpath` Path to user's manual directory (P)

manualpath Path to user's manual directory (P)

Description: Contains the absolute path to a user's directory of VNMR manual entries. If `manualpath` exists for a user, it must be defined in the user's global parameter file. Enter `create('manualpath','string','global')` to create the `manualpath` parameter.

See also: *VNMR User Programming*

Related: `man` Display online description of a command or macro (M)

manvi Edit online description of a command or macro (M)

Syntax: `manvi(file)`

Description: Enables editing the online description of commands and system macros stored in the directory `/vnmr/manual`. You must have write permission to this directory in order to edit the files.

Arguments: `file` is the name of a command or system macro in `/vnmr/manual`.

Examples: `manvi('mark')`

See also: *VNMR User Programming*

Related: `man` Display online description of command or macro (M)

mapwin List of experiment numbers (P)

Syntax: `mapwin`

Description: Arrayed global parameter that maintains a list of experiment numbers for the window panes in the VNMR graphics window.

See also: *Getting Started*

Related: `curwin` Current window (P)
`fontselect` Open FontSelect window (C)
`jwin` Activate current window (M)
`setgrid` Activate selected window (M)
`setwin` Activate selected window (C)

mark Determine intensity of spectrum at a point (C)

Syntax: (1) `mark<(f1_position)><:intensity>`
(2) `mark<(left_edge,region_width)><:intensity, integral>`
(3) `mark<(f1_position,f2_position)><:intensity>`
(4) `mark<(f1_start,f1_end,f2_start,f2_end)><:intensity,integral,c1,c2>`
(5) `mark<('trace',<options>)><:intensity,integral,c1,c2>`
(6) `mark('reset')`

Description: Find the intensity of a spectrum at a point. Either 1D or 2D operations can be performed in the cursor or box mode for a total of four separate functions: 1D operations in cursor mode (syntax 1), 1D operations in box mode (syntax 2), 2D operations in cursor mode (syntax 3) and 2D operations in box mode (syntax 4).

In the *cursor mode*, the intensity at a particular point is found. In the *box mode*, the integral over a region is calculated. The displayed integral is scaled in the same way as output from `dli` is scaled; that is, by the `ins` and `insref` parameters. For 2D operations, this is the volume integral and the volume is scaled by `ins2` and `ins2ref`. In addition, the `mark` command in the box mode finds the maximum intensity and the coordinate(s) of the maximum intensity.

The `mark` command requires that transformed data be present in the current experiment. If required, it recomputes the phase file from the complex data (i.e., it rephases the data if required); however, the `mark` command requires parameters from the command line if no data is displayed (i.e., if `ds` or `dconi` has not been executed).

Note that 2D operations require that 2D data be present. This not only means that `ni` must be larger than 1, but also that the data was transformed using `ft1d`, `ft2d` or an equivalent (and not `ft` or its equivalents).

The `mark` command, as well as the MARK button of `ds`, writes output to a file in the current experiment. For 1D operations, the file is named `mark1d.out`; for 2D operations, it is `mark2d.out`. If this file already exists, VNMR appends output from the current `mark` operation to the end of the file. (Older versions of VNMR used `ds.out` and `dconi.out` as files for output from the MARK button). Either file can be read by other programs at any time between operations.

The following criteria establish the exact function. The command checks them in the following order until it determines the exact function:

1. Number of numeric parameters.
2. Number of return values called out.

3. Which display command (`ds` or `dconi`) was last used.

4. Nature of the data in the experiment.

The first two criteria only serve to distinguish between box mode and cursor mode. The nature of the data in the experiment and the last display command entered determines whether a 1D or a 2D operation is selected.

Arguments: `f1_position` defines the position, in Hz, along the f_1 axis in the 1D and 2D cursor modes. The default is `cr` (1D) or `cr1` (2D).

`left_edge` defines the position of the left edge of the region, in Hz, to be integrated in 1D box mode. The default is `cr`.

`region_width` defines the width, in Hz, of the region, which extends to the right of `left_edge`, in 1D box mode. The default is `delta`.

`f2_position` defines the position, in Hz, along the f_2 axis in the 2D cursor mode. The default is `delta1`.

`f1_start` and `f1_end` define region along the f_1 axis in the 2D box mode.

`f2_start` and `f2_end` define region along the f_2 axis in the 2D box mode.

'`trace`' is a keyword to select a 1D operation if 2D data is present. It must be either the first or the last argument (e.g., `mark('trace', 400)` determines the intensity at 400 Hz in the current trace).

'`reset`' is a keyword to erase the output files from the `mark` command. No other argument can be used with this keyword. Use `rename` to rename the current `mark` output files (e.g., `rename(curexp+' /mark1d.out' , curexp+' /mark.16.01.89')`)

`intensity` is a return value set to the intensity of the spectrum at the point for either 1D or 2D operations (the maximum if cursor mode was selected).

`integral` is a return value set to the integral of the spectrum at the point. `integral` is not returned in the cursor mode.

`c1`, `c2` are return values set to the coordinates where the maximum intensity was found in 2D mode. `c1` and `c2` are not returned in the cursor mode.

Examples: 1D data sets:

<code>mark(cr)</code>	cursor mode for 1D data
<code>mark(cr,delta)</code>	box mode for 1D data

2D data sets (2D mode): In this mode, the order of the arguments to `mark` is independent of the `trace` parameter.

<code>mark(cr1,cr)</code>	cursor mode for 2D data
<code>mark(cr1,delta1,cr,delta)</code>	box mode for 2D data

2D data sets (1D mode): In this mode, the selection of the arguments to `mark` is dependent on the `trace` parameter. If `trace='f2'`, then `cr`, `delta`, `sp`, or `wp` are appropriate. If `trace='f1'`, then `cr1`, `delta1`, `sp1`, and `wp1` are appropriate.

<code>mark('trace',cr)</code>	cursor mode for selected 2D trace
<code>mark('trace',cr1,delta1)</code>	box mode for selected 2D trace

Alternate: MARK button in the `ds` program.

See also: *User Guide: Liquids NMR; VNMR User Programming*

Related: <code>cr</code>	Cursor position in directly detected dimension (P)
<code>cr1</code>	Cursor position in 1st indirectly detected dimension (P)

<code>curexp</code>	Current experiment directory (P)
<code>dconi</code>	Interactive 2D contour display (C)
<code>delta</code>	Difference of two frequency cursors (P)
<code>dli</code>	Display list of integrals (C)
<code>ds</code>	Display a spectrum (C)
<code>ft1d</code>	Fourier transform along f_2 dimension (C)
<code>ft2d</code>	Fourier transform 2D data (C)
<code>ins</code>	Integral normalization scale (P)
<code>ins2</code>	2D volume value (P)
<code>insref</code>	Fourier number scaled value of an integral (P)
<code>ins2ref</code>	Fourier number scaled volume of a peak (P)
<code>mv</code>	Move and/or rename a file (C)
<code>ni</code>	Number of increments in 1st indirectly detected dimension (P)

masvt **Type of variable temperature system (P)**

Applicability: All systems except *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Description: Identifies the type of VT system in use: the standard Oxford VT controller or the Oxford-Sorenson or solids VT controller system (used with the Varian VT CP/MAS probe). `masvt` is a global parameter that is active on all of each user's experiments on a per user account basis. The current value of the parameter can be displayed by typing `masvt?`.

Note that the VT Controller option displayed by `config` must be set to Present for either VT controller system to be active. If `masvt` does not exist, it can be created with the command `create('masvt','string','global')`.

The new Highland VT controller is autosensing, making `masvt` superfluous for systems with this controller.

Values: 'y' indicates the solids VT system is in use.

'n', any other value but 'n' and 'y', or if `masvt` does not exist, indicate that the Oxford Varian VT controller, if present, is in use.

See also: *VNMR and Solaris Software Installation*

Related:	<code>config</code>	Display current configuration and possibly change values (M)
	<code>create</code>	Create a new parameter in a parameter tree (C)
	<code>vttype</code>	Variable temperature controller present (P)

math **Fourier transform mathematics (obsolete)**

Description: This parameter is no longer part of VNMR. Fourier transform math is now always done in floating point.

maxpen **Maximum number of pens to use (P)**

Description: Controls the maximum number of pens that will be used.

Values: 1 to the number of pens in the system plotter. If `maxpen=x` and the software attempts to use pen $x+y$, it uses pen y instead.

See also: *Getting Started*

Related:	<code>pen</code>	Select a pen or color for drawing (C)
	<code>setpen</code>	Set maximum number of HP plotter pens (M)

maxsw_loband **Maximum spectral width of Input board (P)**

Applicability: Systems with imaging capabilities.

Description: Stores the maximum spectral width of the Input board. The system value is set using the Max. Narrowband Width label in the CONFIG window (opened from `config`).

See also: *VNMR and Solaris Software Installation; User Guide: Imaging*

Related: `config` Display current configuration and possibly change it (M)

md Move display parameters between experiments (C)

Syntax: `md(<from_exp,>to_exp)`

Description: Moves the saved display parameters from one experiment to another. These parameters must have been saved with the `s` command (e.g., `s2`).

Arguments: `from_exp` specifies the number of the experiment, 1 through 9, from which the parameters are to be taken. The default is that the parameters are moved from the currently active experiment.

`to_exp` specifies to which experiment the parameters are to be moved.

Examples: `md(4)`
`md(2,3)`

See also: *User Guide: Liquids NMR*

Related: `mf` Move FIDs between experiments (C)
`mp` Move parameters between experiments (C)
`s` Save display parameters as a set (M)

menu Change status of menu system (C)

Syntax: (1) `menu(menu_name)`
(2) `menu('off')`

Description: The VNMR menu system allows up to eight buttons to be active at a time, enabling the user to perform most actions with the mouse rather than typing in commands. All menus are stored in the library `menulib` in the system directory or in the user's `menulib`. See `menuvi` to change these menus.

If the menu system becomes deactivated for some reason, select the Menu On button in the Permanent Menu to reactivate it. Entering `menu('main')` also works.

Arguments: `menu_name` is the name of the file controlling the menu (e.g., 'main'). Including this argument activates the menu system and displays the menu controlled by `menu_name`.

'off' is a keyword to turn off the menu system.

Examples: `menu`
`menu('fitspec')`
`menu('off')`

See also: *Getting Started; VNMR User Programming*

Related: `menuvi` Edit a menu with the `vi` text editor (M)
`mlabel` Menu label (P)
`newmenu` Select a menu without immediate activation (C)

menulibpath Path to user's menu directory (P)

Description: Contains an absolute path to a user's directory of VNMR menu files. If `menulibpath` exists for a user, it must be defined in the user's global

parameter file. To create `menulibpath`, enter the command `create('menulibpath','string','global')`.

See also: *VNMR User Programming*

menuvi **Edit a menu with vi text editor (M)**

Syntax: `menuvi (menu)`

Description: Edits a VNMR menu file using the UNIX `vi` text editor. On the Sun workstation, a pop-up window contains the edit. On the GraphOn, the edit is done on the entire terminal.

Arguments: `menu` is the name of file controlling a menu.

Examples: `menuvi('display_1D')`

See also: *VNMR User Programming*

Related: `menu` Change status of menu system (C)
`newmenu` Select a menu without immediate activation (C)
`vi` Edit text file with `vi` text editor (C)

method **Autoshim method (P)**

Description: Selects the method for automatic shimming. Refer to the manual *Getting Started* for information on how to write or alter methods.

Values: Name of file in the `/vnmr/shimmethods` library for one of the defined shim methods in the system. To display all available methods, enter `ls('/vnmr/shimmethods')`. Standard methods include `'z1z2'` (selects shimming of the Z1 and Z2 gradients) and `'allzs'` (selects shimming all spinning gradients, Z1 to Z4 or Z5, depending on the magnet type). Shim methods can also be stored in a user's `shimmethods` directory (e.g., `/home/vnmr1/vnmrsys/shimmethods`).

See also: *Getting Started*

Related: `ls` List files in current directory (C)
`newshm` Interactively create a shim method with options (M)
`stdshm` Interactively create a shim method (M)

mF **Move FIDs between experiments (C)**

Syntax: `mF (<from_exp,>to_exp)`

Description: Moves the last acquired FID, as well as its associated parameters, from one experiment to another. The text, the processed acquisition parameters and the current display and processing parameters are also moved to the specified experiment.

Arguments: `from_exp` specifies number of the experiment from which the FID is to be taken. The default is the FID is moved from the currently active experiment.
`to_exp` specifies to which experiment the FID is to be moved.

Examples: `mF(4)`
`mF(3,2)`

See also: *User Guide: Liquids NMR*

Related: `md` Move display parameters between experiments (C)
`mp` Move parameters between experiments (C)

mfblok **Copy FID block (C)**

Syntax: `mfblok(<src_expno, >src_blk_no, dest_expno, dest_blk_no)`

Description: Copies data from a source FID block specified by `src_blk_no` to a destination FID block specified by `dest_expno` and `dest_blk_no`, using memory-mapped input and output.

`mfblok` searches for the source and destination FID file in the directory `$vnmruser/expN/acqfil`, where `N` is the requested experiment number or the current experiment number. If the FID file is not open, `mfblok` opens the file, copies the data, and closes the file. If a number of blocks need to be copied, explicitly opening and closing the files with the commands `mfopen` and `mfclose` can significantly speed up the data reformatting process.

`mfblok` can also be used to append blocks of data to a FID file by specifying that the `dest_blk_no` is greater than the number of blocks in a file.

Be aware that `mfblok` can modify data returned to an experiment with the `rt` command. To avoid modification, enter the following sequence of VNMR commands before running `mfblok`:

```
cp(curexp+' /acqfil/ fid' , curexp+' /acqfil/ fidtmp' )
rm(curexp+' /acqfil/ fid' )
mv(curexp+' /acqfil/ fidtmp' , curexp+' /acqfil/ fid' )
```

Arguments: `src_expno` specifies the experiment number of the source FID file. The default is the FID file of the current experiment.

`src_blk_no` specifies the source block of data to be copied. Block numbers start at 1 and run from 1 to the number of blocks in a file.

`dest_expno` specifies the experiment number of the destination FID file.

`dest_blk_no` specifies the destination block to send the copied data.

Examples: `mfblok(1 , 2 , 1)` copies current experiment, block 1 to exp 2, block 1.
`mfblok(3 , 2 , 6 , 2)` copies exp 2, block 2 to exp 6, block 2.

See also: *VNMR User Programming*

Related:	<code>mfclose</code>	Memory map close FID file (C)
	<code>mfdata</code>	Move FID data (C)
	<code>mfopen</code>	Memory map open FID file (C)
	<code>mftrace</code>	Move FID trace (C)

mfclose **Close memory map FID (C)**

Syntax: `mfclose`

Description: Closes experiment source and destination FID files that have been explicitly opened with `mfopen`.

See also: *VNMR User Programming*

Related:	<code>mfblok</code>	Move FID block (C)
	<code>mfdata</code>	Move FID data (C)
	<code>mfopen</code>	Memory map open FID file (C)
	<code>mftrace</code>	Move FID trace (C)
	<code>rfblk</code>	Reverse FID block (C)
	<code>rfdata</code>	Reverse FID data (C)
	<code>rftrace</code>	Reverse FID trace (C)

mfdata **Move FID data (C)**

Syntax: `mfdata (<src_expno,>src_blk_no,src_start_loc, \ dest_expno,dest_blk_no,dest_start_loc,num_points)`

Description: Copies data specified by `src_start_loc` from a FID block specified by `src_blk_no` to a destination location specified by `dest_expno`, `dest_blk_no`, and `dest_start_lo`, using memory-mapped input and output. The data point locations and the `num_points` to be copied are specified by data points corresponding to the `np` parameter, not bytes or complex points.

`mfdata` searches for the source and destination FID file in the directory `$vnmruser/expN/acqfil`, where `N` is the requested experiment number or the current experiment number. If the FID file is not open, `mfdata` opens the file, copies the data, and closes the file. If a number of blocks need to be copied, explicitly opening and closing the files with the commands `mfopen` and `mfclose` can significantly speed up the data reformatting process.

Be aware that `mfdata` can modify data returned to an experiment with the `rt` command. To avoid modification, enter the following sequence of VNMR commands before running `mfdata`:

```
cp(curexp+ '/acqfil/fid', curexp+ '/acqfil/fidtmp' )
rm(curexp+ '/acqfil/fid' )
mv(curexp+ '/acqfil/fidtmp', curexp+ '/acqfil/fid' )
```

Arguments: `src_expno` specifies the experiment number of the source FID file. The default is the FID file of the current experiment.

`src_blk_no` specifies the source block of data to be copied. Block numbers start at 1 and run from 1 to the number of blocks in a file.

`src_start_loc` specifies the starting data location within the specified block to copy the data. Data locations start from 0 and are specified as data points corresponding to the `np` parameter.

`dest_expno` specifies the experiment number of the destination FID file.

`dest_blk_no` specifies the destination block to send the copied data.

`dest_start_loc` specifies the starting data destination location within the specified block to send the copied data.

Examples: `mfdata (1, 0, 2, 1, (nv-1) * np, np)` copies `np` points of data from the starting location 0 of block 1 of the current experiment to the data location `(nv-1) * np` of block 1 of experiment 2.

See also: *VNMR User Programming*

Related:	<code>mfblk</code>	Move FID block (C)
	<code>mfclose</code>	Memory map close FID file (C)
	<code>mfdata</code>	Move FID data (C)
	<code>mfopen</code>	Memory map open FID file (C)
	<code>mftrace</code>	Move FID trace (C)
	<code>rfblk</code>	Reverse FID block (C)
	<code>rftrace</code>	Reverse FID trace (C)

mfopen **Memory map open FID file (C)**

Syntax: `mfopen(< <src_expno,>dest_expno>)>`

Description: Explicitly opens experiment source and destination FID files for using memory-mapped input and output. Opening a file explicitly can significantly speed up the data reformatting process.

`mfopen` searches for the FID file to be opened in the directory `$vnmruser/expN/acqfil`, where `N` is the requested experiment number or the current experiment number. Without arguments, `mfopen` assumes the source and destination files are the same and are in the current experiment.

After a file is open, the data reformatting commands `mfblk`, `mfdata`, `mftrace`, `rfblk`, `rfdata`, and `rftrace` can be used for moving around data. The `mfclose` must be used to close the file when data reformatting has been completed.

Arguments: `src_expno` specifies the experiment number of the source FID file. The default is the FID file of the current experiment.

`dest_expno` specifies the experiment number of the destination FID file. The default is the FID file of the current experiment.

If only one argument is provided, `mfopen` uses that as the experiment number of the destination FID file and assumes the source is the FID file of the current experiment.

Examples: `mfopen`
`mfopen(3)`
`mfopen(1,2)`

See also: *VNMR User Programming*

Related:	<code>mfblk</code>	Move FID block (C)
	<code>mfclose</code>	Memory map close FID file (C)
	<code>mfdata</code>	Move FID data (C)
	<code>mftrace</code>	Move FID trace (C)
	<code>rfblk</code>	Reverse FID block (C)
	<code>rfdata</code>	Reverse FID data (C)
	<code>rftrace</code>	Reverse FID trace (C)

mftrace **Move FID trace (C)**

Syntax: `mftrace(<src_expno,>src_blk_no,src_trace_no, \`
`dest_expno,dest_blk_no,dest_trace_no)`

Description: Copies FID traces specified by `src_trace_no` from a FID block specified by `src_blk_no` to a destination location specified by `dest_expno`, `dest_blk_no`, and `dest_trace_no`, using memory-mapped input and output. If a number of blocks need to be copied, explicitly opening and closing the files with the commands `mfopen` and `mfclose` can significantly speed up the data reformatting process.

`mftrace` searches for the source and destination FID file in the directory `$vnmruser/expN/acqfil`, where `N` is the requested experiment number or the current experiment number. If the FID file is not open, `mftrace` opens the file, copies the data, and closes the file.

`mftrace` cannot be used to append data to a FID file. Its purpose is for moving around data.

Be aware that `mftrace` can modify data returned to an experiment with the `rt` command. To avoid modification, enter the following sequence of VNMR commands before running `mftrace`:

```
cp(curexp+' /acqfil/fid',curexp+' /acqfil/fidtmp' )
rm(curexp+' /acqfil/fid' )
mv(curexp+' /acqfil/fidtmp',curexp+' /acqfil/fid' )
```

Arguments: `src_expno` specifies the experiment number of the source FID file. The default is the FID file of the current experiment.

`src_blk_no` specifies the source block of data to be copied. Block numbers start at 1 and run to the number of blocks in a file.

`src_trace_no` specifies the source trace of data within the specified block to be copied. Trace numbers run from 1 to number of traces in a file.

`dest_expno` specifies the experiment number of the destination FID file.

`dest_blk_no` specifies the destination block to send the copied data.

`src_trace_no` specifies the destination trace of data within the specified block to be copied. Trace numbers run from 1 to the number of traces in a file.

Examples: `mftrace(1,1,2,1,nv)` copies trace 1 from block 1 of the current experiment to trace `nv` of block 1 of experiment 2.

See also: *VNMR User Programming*

Related:	<code>mfblk</code>	Move FID block (C)
	<code>mfclose</code>	Memory map close FID file (C)
	<code>mfdata</code>	Move FID data (C)
	<code>mfopen</code>	Memory map open FID file (C)
	<code>rftrace</code>	Reverse FID trace (C)
	<code>rfblk</code>	Reverse FID block (C)
	<code>rfdata</code>	Reverse FID data (C)

minsw **Reduce spectral width to minimum required (M)**

Syntax: `minsw`

Description: Searches the spectrum for peaks, sets new limits accordingly, and then calls `movesw` to calculate a new transmitter offset `tof` and spectral width `sw`.

See also: *Getting Started*

Related:	<code>movesw</code>	Move spectral window according to cursors (M)
	<code>movetof</code>	Move transmitter offset (M)
	<code>sw</code>	Spectral width in directly detected dimension (P)
	<code>tof</code>	Frequency offset for transmitter offset (P)

mkdir **Create new directory (C)**

Syntax: `mkdir(directory)`

Description: Creates a new UNIX directory. The function of the VNMR `mkdir` command is similar to the UNIX `mkdir` command.

Arguments: `directory` is the name of the new directory to be created.

Examples: `mkdir('tests')`
`mkdir('/home/george')`

See also: *Getting Started*

Related:	<code>rmdir</code>	Remove directory (C)
----------	--------------------	----------------------

mlabel **Menu label (P)**

Description: Stores the label for a menu button. Usually this parameter is arrayed, with one label for each button in the menu. This parameter is stored in a user's global file and is set whenever a menu is called.

See also: *VNMR User Programming*

Related:	<code>menu</code>	Change status of menu system (C)
	<code>mstring</code>	Menu string (P)

move **Move to an absolute location to start a line (C)**

Syntax: `move(<'graphics' | 'plotter'>, x, y)`

Description: Moves the start of a line to an absolute location with the coordinates given as an argument. `move` is part of a line drawing capability that includes the `pen` and `draw` commands. `pen` selects the pen number of the plotter ('pen1', 'pen2', etc.) or the color ('red', 'green', 'blue', etc.). `move` sets the point from which to start drawing the line. `draw` draws a line from that point to the point given by the `draw` arguments. Refer to the description of the `draw` command for examples of using the line drawing capability.

Arguments: 'graphics' and 'plotter' are keywords selecting output to the graphics window or a plotter device. The default is 'plotter'. The output selected is passed to subsequent `pen`, `move`, or `draw` commands, remaining unchanged until different output is specified.

`x`, `y` are the absolute coordinates, in mm, of a point to move to. The range of `x` is 0 at the left edge of the chart and `wcmax` at the right edge of the chart. The range of `y` is -20 at the bottom of the chart and `wc2max` at the top.

See also: *Getting Started*

Related:	<code>draw</code>	Draw line from current location to another location (C)
	<code>gin</code>	Return current mouse position and button values (C)
	<code>pen</code>	Select a pen or color for drawing (C)
	<code>wcmax</code>	Maximum width of chart (P)
	<code>wc2max</code>	Maximum width of chart in second direction (P)

movedssw **Set downsampling parameters for selected spectral region (M)**

Syntax: `movedssw`

Description: Sets the parameters `ds`, `dslsfrq` and `downsamp` to appropriate values for digital filtering and downsampling in a cursor-selected spectral region. To accomplish this, Fourier transform an oversampled data set, and then run the `ds` program. In the resulting spectral display, enclose the desired region with the cursors, and then run `movedssw`.

See also: *Getting Started*

Related:	<code>downsamp</code>	Downsampling factor applied after digital filtering (P)
	<code>ds</code>	Display a spectrum (C)
	<code>dslsfrq</code>	Bandpass filter offset for downsampling (P)

moveossw **Set oversampling parameters for selected spectral region (M)**

Syntax: `moveossw`

Description: Sets the parameters `os`, `oslsfrq` and `sw` to appropriate values for oversampling and digital filtering in a cursor-selected spectral region. To accomplish this, acquire a data set without digital filtering, and then run the `ds` program. In the resulting spectral display, enclose the desired region with the cursors, and then run `moveossw`. The value of `oversamp` is manually set.

See also: *Getting Started*

Related:	<code>ds</code>	Display a spectrum (C)
	<code>oslsfrq</code>	Bandpass filter offset for oversampling (P)
	<code>oversamp</code>	Oversampling factor for acquisition (P)
	<code>sw</code>	Spectral width in directly detected dimension (P)

movepro **Move the imaging readout position (C)**

Syntax: `movepro`

Description: Sets the readout position for an image or image projection to a point defined by the position of the cursor (the `cr` parameter).

`movepro` works with a 1D display (a projection or trace along F2) or 2D display, in either single cursor or box modes (only the position of the cursor in the F2 readout dimension is used; the position of the cursor in the F1 phase-encode dimension does not matter).

`movepro` determines the position of the cursor relative to the gradient origin and sets the parameter `pro` to this value, independent of image orientation. Because `pro` is measured in dimensional units like mm or cm, and the cursor position is stored internally in hertz, `movepro` works in Hz, accounting for any spectral referencing that may have been set, and converts to cm or mm to assign the value of `pro`.

To use `movepro`, display an image, image projection or trace, move the cursor to the position along the readout axis you desire to be at the center of the next image acquisition, and type `movepro`. This command has no effect on the value of `tof` (which is normally not used to define any positional information in imaging). Unlike `movetof`, the image or projection display will be unchanged, and no redisplay in “full” mode should be necessary.

To accurately center an image or projection, move the box cursors to the edges of the imaged object. Then use the macro `split` to place the cursor at the exact midpoint of the box, and type `movepro`.

See also: *User Guide: Imaging*

Related:	<code>cr</code>	Cursor position in directly detected dimension (P)
	<code>lro</code>	Field of view parameter for read out in cm (P)
	<code>movetof</code>	Move transmitter offset (M)
	<code>pro</code>	Position of image center on the readout axis (P)
	<code>resto</code>	NMR resonance offset frequency (P)
	<code>tof</code>	Frequency offset for observe transmitter (P)

movesw **Move spectral window according to cursors (M)**

Syntax: `movesw<(width)>`

Description: Uses the parameters `cr` and `delta` to calculate a new transmitter offset `tof` and a new spectral width `sw`. If referencing was used, it is also adjusted. The `movesw` macro also sets `sp` and `wp` to display the spectral window.

Arguments: `width` specifies the spectral width `sw`. The default is to use a value calculated from the parameter `delta`.

Examples: `movesw`
`movesw(5000)`

See also: *Getting Started*

Related:	<code>cr</code>	Cursor position in directly detected dimension (P)
	<code>delta</code>	Cursor difference in directly detected dimension (P)
	<code>minsw</code>	Reduce spectral width to minimum required (M)
	<code>movetof</code>	Move transmitter offset (M)
	<code>sp</code>	Start of plot (P)
	<code>sw</code>	Spectral width in directly detected dimension (P)
	<code>tof</code>	Frequency offset for observe transmitter (P)
	<code>wp</code>	Width of plot (P)

movetof **Move transmitter offset (M)**

Syntax: `movetof<(frequency)>`

Description: Moves the transmitter offset parameter `tof` so that the current cursor position, defined by `cr`, becomes the center of the spectrum. If referencing was used, `movetof` maintains the referencing.

Arguments: `frequency` specifies the transmitter frequency rather than using the cursor position to define the frequency. This provides a convenient method of moving the transmitter frequency outside the current spectral window.

See also: *Getting Started*

Related:	<code>cr</code>	Cursor position in directly detected dimension (P)
	<code>minsw</code>	Reduce spectral width to minimum required (M)
	<code>movesw</code>	Move spectral window according to cursors (M)
	<code>tof</code>	Frequency offset for observe transmitter (P)

mp **Move parameters between experiments (C)**

Syntax: `mp(<from_exp,>to_exp)`

Description: Moves text and the current display, processing, and acquisition parameters from one experiment to another. No FID is transferred.

Arguments: `from_exp` specifies the number of the experiment from which the parameters are to be taken; default is the parameters are moved from the currently active experiment.

`to_exp` specifies to which experiment the parameters are to be moved.

Examples: `mp(4)`
`mp(2,3)`

See also: *User Guide: Liquids NMR*

Related:	<code>md</code>	Move display parameters between experiments (C)
	<code>mf</code>	Move FIDs between experiments (C)

mqcosy **Set up parameters for MQCOSY pulse sequence (M)**

Applicability: All systems, except sequence not supplied with *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Syntax: `mqcosy<(level)>`

Description: Sets up a multiple-quantum filtered COSY experiment.

Arguments: `level` is the desired quantum level of filtration.

Examples: `mqcosy`
`mqcosy(3)`

See also: *User Guide: Liquids NMR*

mrev8 **Set up parameters for MREV8 pulse sequence (M)**

Applicability: Systems with a solids module. This sequence not supplied with *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Syntax: `mrev8`

Description: Converts FLIPFLOP, BR24, or S2PUL parameter set into the MREV8 multiple-pulse line narrowing sequence.

See also: *User Guide: Solid-State NMR*

Related: **br24** Set up parameters for BR24 pulse sequence (M)
cylmrev Set up parameters for cycled MREV8 pulse sequence (M)
flipflop Set up parameters for FLIPFLOP pulse sequence (M)
s2pul Set up parameters for standard two-pulse sequence (M)

mrfb Set the filter bandwidths for multiple receivers (P)

Applicability: Systems with multiple receivers

Description: An array of 'fb' settings to apply to individual receivers in a multiple receiver system. If it exists and is active, these settings override the setting specified by the 'fb' parameter; otherwise, 'fb' is used as the filter bandwidth setting for all receivers.

Note that not all multiple receiver systems have the hardware set up to provide individual receiver control. In that case, the filter setting for receiver 1 is used on receivers 1 and 2 and the setting for receiver 3 is used on receivers 3 and 4. Also note that 'mrfb' is not set automatically when 'sw' is changed.

Normally, you can leave 'mrfb' inactive and let 'fb' be used for all receivers.

Examples: `mrfb=fb,fb/2` sets the filter bandwidth of the first receiver to fb and of the rest to fb/2.

Related: **fb** Filter bandwidth (P)
sw Spectral width in directly detected dimension (P)

mrgain Set the gain for multiple receivers (P)

Applicability: Systems with multiple receivers

Description: An array of 'gain' settings to apply to individual receivers in a multiple receiver system. If it exists and is active, these settings override the setting specified by the 'gain' parameter; otherwise, 'gain' is used as the gain setting for all receivers.

Note that not all multiple receiver systems have the hardware set up to provide individual receiver control. In that case, the gain setting for receiver 1 is used on receivers 1 and 2 and the setting for receiver 3 is used on receivers 3 and 4.

Examples: `mrgain=30,40,20` sets the gains of receiver 1 to 30, receiver 2 to 40 and receivers 3 and 4 to 20.

Related: **gain** Receiver gain (P)

mstat Display memory usage statistics (C)

Syntax: `mstat<(program_id)>`

Description: Displays statistics on memory usage by programs that use the procedures `allocateWithId` and `release`.

Arguments: `program_id` is the program ID, usually the same name as the program. The default is to display all program IDs and associated memory statistics.

Examples: `mstat`
`mstat('proc2d')`

See also: *VNMR User Programming*

mstring **Menu string (P)**

Description: Stores command strings to be executed when a VNMR menu button is clicked. Usually the `mstring` parameter is arrayed, with one string for each button in the menu. The string can be any string of commands that can otherwise appear in a macro or on the command line. This parameter is stored in a user's global file and is set whenever a menu is called.

See also: *VNMR User Programming*

Related: `menu` Change status of menu system (C)
`mlabel` Menu label (P)

mv **Move and/or rename a file (C)**

Syntax: `mv(from_file,to_file)`

Description: Renames and/or moves a file or directory. `mv` functions the same as the command `rename`.

Arguments: `from_file` is the name of the file to be moved and/or renamed.
`to_file` is the new name of the file and/or the new location. If the `from_file` argument has an extension such as `.fid` or `.par`, be sure the `to_file` argument has the same extension.

Examples: `mv(' /home/vnmr1/vnmrsys/seqlib/d2pul ',`
`' /vnmr/seqlib/d2pul ')`

See also: *Getting Started*

Related: `copy` Copy a file (C)
`cp` Copy a file (C)
`delete` Delete a file, parameter directory, or FID directory (C)
`rename` Move and/or rename a file (C)
`rm` Delete a file (C)

mxconst **Maximum scaling constant (P)**

Description: Before the start of data acquisition, noise is sampled to determine the number of bits of noise present. This number is used to set the maximum number of scaling operations on the data that can occur (essentially relevant only if `dp= 'n'`). `mxconst` is used to adjust this amount of scaling.

Increasing `mxconst` to 1, for example, permits additional scaling operations, allowing acquisition to proceed slightly longer in single-precision mode.

Decreasing `mxconst` to -1 allows fewer scaling operations before reaching the message "maximum transients accumulated".

One special case exists. If `mxconst` is set to less than -90 and single-precision acquisition is used (`dp= 'n'`), then scaling of the data is disabled. In this mode, reports of data overflowing the 16 bits is also disabled.

`mxconst` does not exist in standard parameter sets. If it does not exist, its value defaults to 0. To modify `mxconst`, first create it by entering `create('mxconst','integer')` and then enter the desired value.

CAUTION: Do not change `mxconst` unless you are fully aware of the consequences.

See also: *Getting Started*

Related: `create` Create new parameter in a parameter tree (C)
`dp` Double precision (P)

N

n1 , n2 , n3 Name storage for macros (P)

Description: Stores arbitrary character strings for macros. Each experiment has these three string parameters available.

See also: *VNMR User Programming*

Related: **dgs** Display group of special/automation parameters (M)
r1-r7 Real value storage for macros (P)

nactivercvrs Return number of receivers currently active (M)

Applicability: Systems with multiple receivers.

Syntax: `nactivercvrs`

Description: Calculates and returns the number of receivers currently active, based on the values of the 'rcvrs' and 'numrcvrs' parameters.

Examples: `nactivercvrs:$nact` sets '\$nact' to the number of currently active receivers.

Related: **rcvrs** Which receivers to use (P)
numrcvrs Number of receivers in the system (P)

nD Application dimension (P)

Applicability: Systems with the imaging capabilities.

Description: Defines the dimension of the experiment performed by the application code. The value of nD is the number of FFT (fast Fourier transform) operations used to reconstruct the data or the number of independent k space coordinates encoded in the data. The nD, **seqcon**, **plist**, **patlist**, **pwrlist**, **fliplist** and **sslist** parameters configure a particular parameter set for an application sequence defined by the value of the **seqfil** parameter.

Values: 1, 2, 3, or 4.

See also: *User Guide: Imaging*

Related: **fliplist** Standard flip angle list (P)
patlist Active pulse template parameter list (P)
plist Active pulse length parameter list (P)
pwrlist Active pulse power level parameter list (P)
seqcon Acquisition loop control (P)
seqfil Application object code name (P)
sslist Conjugate gradient list (P)

ne Number of echoes to be acquired (P)

Applicability: Systems with the imaging capabilities.

Description: Sets number of echoes to be acquired for multiecho sequences.

Values: 1 to desired number, in integer steps.

See also: *User Guide: Imaging*

Related: **ns** Number of slices to be acquired (P)

newmenu **Select a menu without immediate activation (C)**

Syntax: (1) `newmenu(menu_name)`
 (2) `newmenu:$current_menu`

Description: Selects a menu but does not activate it (syntax 1). This is most useful when picking which menu will be active when an interactive command exits. `newmenu` can also return the name of the currently active menu (syntax 2).

Arguments: `menu_name` is the name of the file controlling the menu selected. For example, the command string `newmenu('manipulate_1D')` `ds` causes the menu controlled by `manipulate_1D` to be displayed when the Return button in the `ds` menu is selected.

`$current_menu` returns the file name of the currently active menu.

Examples: `newmenu('display_1D')`
`newmenu:$name1`

See also: *VNMR User Programming*

Related: `menu` Change status of menu system (C)
`menuvi` Edit a menu with the *vi* text editor (M)

newshm **Interactively create a shim method with options (M)**

Syntax: `newshm`

Description: Interactively creates a *method* string to be used in autoshimming of the magnetic field homogeneity. The string may consist of a series of shimming operations. The command `dshim('method')` describes method strings. Any text editor may be used to make and modify the strings.

`newshm` provides for either lock shimming or FID shimming, permitting the user to choose whichever is best (FID shimming is not supported by the *GEMINI 2000*). Lock shimming is much faster, but FID shimming is frequently much more effective in improving the field. With FID shimming, the FID evaluation range limits are requested. The full range is 0 to 100. Sensitivity to higher order gradients is greatly increased by setting the finish limit to about 5 or 10 with the start limit at 0.

`newshm` begins by asking for the name of the user's new shim method. If the non-spin (transverse) controls are chosen for adjustment, the spinner is turned off; otherwise, it is turned on. If uncertain about the shim criteria, the "medium to medium" choice is suitable in most circumstances. The new method is found in `curexp+'.../shimmethods`.

To shim after running `newshm`, type `method='methodname'` and then enter `shim` or set the `wshim` parameter to `shim` before the start of acquisition. 'methodname' is the name supplied to `newshm`. For more information on shimming, see the manual *Getting Started*.

Compared to `stdshm`, the `newshm` macro is more flexible and provides for a shimming time and FID evaluation limits supplied by the user. The primary difference between the macros is that `stdshm` provides for determining an estimated shimming time for the selected shim controls. When no time limit is supplied, autoshim continues until the exit criteria is met or the number of cycles reaches a limit.

See also: *Getting Started*

Related: `curexp` Current experiment directory (P)
`dshim` Display a shim method string (M)
`method` Autoshim method (P)
`shim` Submit an Autoshim experiment to acquisition (C)
`stdshm` Interactively create a shim method (M)

`wshim` Conditions when shimming is performed (P)
`vi` Edit text file with `vi` text editor (C)

nextpl Display the next 3D plane (M)

Applicability: All systems; however, although `nextpl` is available on *GEMINI 2000* systems, such systems can only process 3D data and cannot acquire 3D data.

Syntax: `nextpl`

Description: Displays the 2D color map of the next 3D plane in the set of planes defined by the parameters `plane` and `path3d`. If `nextpl` immediately follows the command `dproj`, `nextpl` results in the display of the first 3D plane within that specified set and is therefore equivalent to the command `dplane (1)`. For example, if `dplane (40)` has just been executed, `nextpl` results in the display of 3D plane 41 of that set. The `nextpl` macro is more efficient than `dplane` or `dproj` because the 3D parameter set (`procp3d`) is not loaded into VNMR—it is assumed to have already been loaded by `dplane` or `dproj`, for example.

See also: *User Guide: Liquids NMR*

Related: `dplane` Display a 3D plane (M)
`dproj` Display a 3D plane projection (M)
`dsplanes` Display a series of 3D planes (M)
`getplane` Extract planes from a 3D spectral data set (M)
`path3d` Path to currently displayed 2D planes from a 3D data set (P)
`plane` Currently displayed 3D plane type (P)
`plplanes` Plot a series of 3D planes (M)
`prevpl` Display the previous 3D plane (M)

nf Number of FIDs (P)

Applicability: Systems with imaging capabilities.

Description: Number of FIDs acquired by explicit acquisition.

Values: Positive integer. For example, in the COSY-NOESY experiment, `nf` is 2.

See also: *User Guide Imaging*

Related: `cf` Current FID (P)

ni Number of increments in 1st indirectly detected dimension (P)

Description: Number of increments of the evolution time `d2`, and thus the number of FIDs that will comprise the first indirectly detected dimension of a multidimensional data set. To create parameters `ni`, `phase`, and `sw1` to acquire a 2D data set in the current experiment, enter `addpar (' 2d ')`.

Values: 8 is minimum; typical values range from 32 to 512. In microimaging, `ni` greater than 0 is the imaging mode and `ni` equal to 0 is the projection mode.

See also: *User Guide: Liquids NMR; User Guide: Imaging*

Related: `addpar` Add selected parameters to the current experiment (M)
`celem` Completed FID elements (P)
`d2` Incremented delay in 1st indirectly detected dimension (P)
`ni2` Number of increments in 2nd indirectly detected dimension (P)

- ni2** **Number of increments in 2nd indirectly detected dimension (P)**
- Description: Number of increments of the evolution time **d3**, and thus the number of FIDs that will comprise the second indirectly detected dimension of a multidimensional data set. To create parameters **d3**, **ni2**, **phase2**, and **sw2** to acquire a 3D data set in the current experiment, enter **addpar** (' 3d ').
- See also: *User Guide: Liquids NMR*
- Related: **addpar** Add selected parameters to the current experiment (M)
 d3 Incremented delay in 2nd indirectly detected dimension (P)
 ni Number of increments in 1st indirectly detected dimension (P)
 par3d Create 3D acquisition, processing, and display parameters (M)
 phase2 Phase selection for 3D acquisition (P)
 sw2 Spectral width in 2nd indirectly detected dimension (P)
- ni3** **Number of increments in 3rd indirectly detected dimension (P)**
- Description: Number of increments of the evolution time **d4**, and thus the number of FIDs that will comprise the third indirectly detected dimension of a multidimensional data set. To create parameters **d4**, **ni3**, **phase3**, and **sw3** to acquire a 4D data set in the current experiment, enter **addpar** (' 4d ').
- See also: *User Guide: Liquids NMR*
- Related: **addpar** Add selected parameters to the current experiment (M)
 d4 Incremented delay in 3rd indirectly detected dimension (P)
 ni Number of increments in 1st indirectly detected dimension (P)
 ni2 Number of increments in 2nd indirectly detected dimension (P)
 par4d Create 4D acquisition parameters (M)
 phase3 Phase selection for 4D acquisition (P)
 sw3 Spectral width in 3rd indirectly detected dimension (P)
- niter** **Number of iterations (P)**
- Description: Sets the maximum number of iterations in an iterative simulation.
- Values: 1 to 9999. The value is initialized to 20 if the Set Params button is used in setting up spin simulation parameters.
- See also: *User Guide: Liquids NMR*
- n1** **Position cursor at the nearest line (C)**
- Syntax: **n1**<:height<, frequency>>
- Description: Moves the cursor to the nearest calculated line position.
- Arguments: **height** is a return value set to the height of the line.
frequency is a return value set to the frequency of the line.
- Examples: **n1**
n1:r1,r2
- See also: *Getting Started*
- n1i** **Find integral values (C)**
- Syntax: **n1i**
- Description: Equivalent to the **d1i** command except that no screen display is produced. For a list of integrals, **n1i** stores the reset points in the parameter **lifrq** and stores the amplitudes in the parameter **liamp**.

See also: *Getting Started*

Related:	cz	Clear integral reset points (C)
	dli	Display list of integrals (C)
	dlni	Display list of normalized integrals (M)
	liamp	Amplitudes of integral reset points (P)
	lifrq	Frequencies of integral reset points (P)
	z	Add integral reset point at cursor position (C)

nlivast Produces a text file of integral regions without a sum region (M)

Applicability: Systems with VAST accessory.

Syntax: `nlivast (last)`

Description: Using predefined integral regions from the spectra for each well, `nlivast` writes a text file, `integ.out`, containing the integrals of the regions. The file is written into the current experiment. Does not add an additional region that is the sum of all the defined regions for each well (see `dlivast`).

Arguments: `last` is the number of the last well. The default is 96.

See also: *User Guide: Liquids NMR*

nlivast2 Produces a text file with normalized integral regions (M)

Applicability: Systems with VAST accessory.

Syntax: `nlivast (well)`

Description: Using predefined integral regions from the spectra for each well, `nlivast2` writes a text file, `integ.out`, containing the integrals of the regions. The file is written into the current experiment. Integrals are normalized to the integral specified by the argument `well`. The macro `nlivast2` does not add an additional region that is the sum of all the defined regions for each well (see `dlivast`). All of the spectra are integrated.

Arguments: `well` is the number of the reference sample well. The default reference is well 96.

See also: *User Guide: Liquids NMR*

nlivast3 Produces a text file with normalized integral regions (M)

Applicability: Systems with VAST accessory.

Syntax: `nlivast (well)`

Description: Using predefined integral regions from the spectra for each well, `nlivast3` writes a text file, `integ.out`, containing the integrals of the regions. The file is written into the current experiment. Integrals are referenced to the integral specified by the argument `well`. The integral of spectrum from the sample specified by `well` is set to 1000. The macro `nlivast3` does not add an additional region that is the sum of all the defined regions for each well (see `dlivast`). All of the spectra are integrated.

Arguments: `well` is the number of the reference sample well. Reference integral set to 1000. The default reference is well 96.

See also: *User Guide: Liquids NMR*

nll Find line frequencies and intensities (C)

Syntax: `nll (<'pos'<, noise_mult>) <:number_lines, scale>`

Description: Equivalent to the command `d11` except that the line listing is not displayed or printed. The results of this calculation are stored in `11frq` and `11amp`. The frequencies are stored as Hz and are not referenced to `rfl` and `rfp`. Amplitudes are stored as the actual data point value; they are not scaled by `vs`.

Arguments: `'pos'` is a keyword that causes only positive lines to be listed.

`noise_mult` is a numerical value that determines the number of noise peaks listed for broad, noisy peak. The default is 3. A smaller value results in more peaks, a larger value results in fewer peaks, and a value of 0.0 results in a line listing containing all peaks above the threshold `th`. Negative values of `noise_mult` are changed to 3.

`number_lines` is a return argument with the number of lines in the line list.

`scale` is a return argument with a scaling factor for line amplitudes. This scaling factor accounts for `vs` and whether the lines are listed in absolute intensity mode or normalized mode.

Examples: `n11:n1`
`n11('pos'):pn`
`n11(2.5),sc`

See also: *VNMR User Programming*

Related: `d11` Display listed line frequencies and intensities (C)
`11amp` List of line amplitudes (P)
`11frq` List of line frequencies (P)

n1ni Find normalized integral values (obsolete)

Description: Macro no longer used in VNMR

nm Select normalized intensity mode (C)

Syntax: `nm`

Description: Selects the normalized intensity mode in which spectra are scaled so that the largest peak in the spectrum is `vs` mm high. The alternative is the absolute intensity mode (selected by the `ai` command) in which the scale is kept constant from spectrum to spectrum to allow comparison of peak heights from one spectrum to another. The modes are mutually exclusive (i.e., the system is always in either `nm` or `ai` mode). Enter `aig?` to show which mode is currently active.

See also: *Getting Started*

Related: `ai` Select absolute intensity mode (C)
`aig` Absolute intensity group (P)
`vs` Vertical scale (P)

nm2d Select Automatic 2D normalization (M)

Syntax: `nm2d<(noisemult)>`

Description: Sets up parameters `th` and `vs2d` automatically for a 2D contour plot and color map display. `nm2d` measures the highest signal in the spectrum and sets `vs2d` so that the highest signal is in the range of the highest color level. It then calculates the noise threshold so that the number of points above the noise threshold is between 10% and 30% of all the points. At the same time, the difference between the mean value of all the points above the threshold (peak

points) and the mean value of all the points under the threshold (noise points) is maximized. This noise threshold is then multiplied by the noise multiplier.

`nm2d` works both with absolute-value and phase-sensitive spectra. `trace` can be set to 'f1' or 'f2'.

Arguments: `noisemult` specifies the noise multiplier number that multiplies the noise threshold:

- For ^1H , ^{19}F and ^{31}P (high dynamic range nuclei), and homonuclear spectra in general, the default value is 4.
- For HMQC/HSQC type spectra, the default value is also 4 but noise multipliers of 3 to 5 are often more adequate.
- For HETCOR and 2D-INADEQUATE spectra, the default value is 2.
- For “quick & dirty” COSY spectra with lots of t1 noise and other artifacts, a value of 8 and higher may be adequate for suppressing the artifacts.
- For 2D-INADEQUATE spectra, a value below 3 is appropriate to catch signals right above the noise level.
- If the multiplied noise threshold is below `th=1`, `vs2d` is scaled up; otherwise, `th` is increased to the desired level.
- Minimum value is 1.5 (if a lower value is entered, the value is set to 1.5).

Examples: `nm2d`
`nm2d (3)`

See also: *Getting Started*

Related:	<code>dconi</code>	Interactive 2D contour display (C)
	<code>noisemult</code>	Control noise multiplier for automatic 2D processing (M)
	<code>proc2d</code>	Process 2D spectra (M)
	<code>th</code>	Threshold (P)
	<code>trace</code>	Mode for <i>n</i> -dimensional data display (P)
	<code>vs2d</code>	Vertical scale for 2D displays (P)

noedif **Convert parameters for NOE difference experiment (M)**

Applicability: *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems only.

Syntax: `noedif`

Description: Converts a ^1H parameter set to perform the NOE (Nuclear Overhauser Enhancement) difference experiment.

See also: *User Guide: Liquids NMR*

Related:	<code>setup</code>	Set up parameters for basic experiments (M)
	<code>cyclenoe</code>	Set up parameters for CYCLENOE pulse sequence (M)

NOESY **Change parameters for NOESY experiment (M)**

Syntax: `NOESY< (' GLIDE ') >`

Description: Converts the current parameter set to a NOESY experiment.

Arguments: 'GLIDE' is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding proton spectrum for the experiment.

Related:	<code>noesy</code>	Set up parameters for NOESY experiment (M)
----------	--------------------	--

noesy **Set up parameters for NOESY pulse sequence (M)**

Syntax: `noesy`

Description: Sets up parameters for the laboratory frame Overhauser experiment or the 2D exchange experiment.

Alternate: NOESY button in the 2D Pulse Sequence Setup Menu.

See also: *User Guide: Liquids NMR*

Related: **foldt** Fold COSY-like spectrum along diagonal axis (C)

NOESY1D Change parameters for NOESY1D experiment (M)

Syntax: NOESY1D<('GLIDE')>

Description: Converts the current parameter set to a NOESY1D (also known as DPGSE-noe) experiment. A 1D proton spectrum is displayed with the `ds_selfrq` menu to do peak selection.

Arguments: 'GLIDE' is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding proton spectrum for the experiment.

Related: **TOCSY1D** Change parameters for TOCSY1D experiment (M)

noise Measure noise level of FID (C)

Syntax: noise<(excess_noise<,last_noise<,block_number>>>
:r1,r2,r3,r4,r5,r6

Description: Measures the noise level of a FID. By using `pw=0` so that no real signal is accumulated, one or more transients can be acquired. The value of `np` must be greater than 4096. `noise` then performs a statistical analysis of the noise, providing noise level, dc level, etc., for each channel. The noise level measurement can be repeated at various settings of `gain` and various settings of `fb`, etc., for a full system diagnosis.

Arguments: `excess_noise` is excess noise and is used to calculate the noise figure.

`last_noise` is the last measured mean square noise and is used to calculate the noise figure.

`block_number` is the block number. The default is 1.

`r1` returns the real dc offset.

`r2` returns the imaginary dc offset.

`r3` returns the real rms noise.

`r4` returns the imaginary rms noise.

`r5` returns the average rms noise.

`r6` returns the percentage channel imbalance.

`r7` returns the noise figure.

See also: *Getting Started*

Related: **ddf** Display data file in current experiment (C)
ddff Display FID file in current experiment (C)
ddfp Display phase file in current experiment (C)
fb Filter bandwidth (P)
gain Receiver gain (P)
np Number of data points (P)
pw Pulse width (P)

noisemult Control noise multiplier for automatic 2D processing (M)

Syntax: noisemult<(noise_multiplier)>

Description: Predetermines the noise multiplier used by the `nm2d` macro when starting automatic 2D experiments. This multiplier determines the threshold level in 2D spectra.

Arguments: `noise_multiplier` is a noise multiplier, the same as used in the `nm2d` macro. The default is 8 for homonuclear 2D spectra or 4 for other spectra.

Examples: `noisemult`
`noisemult(10)`

See also: *User Guide: Liquids NMR*

Related: `nm2d` Automatic 2D normalization (M)
`proc2d` Process 2D spectra (M)

noislm Limit noise in spectrum (M)

Syntax: `noislm<(max_noise)>`

Description: Limits the noise present in a spectrum by reducing the vertical scale `vs`. If the noise is smaller than the noise limit, `vs` is left untouched. The noise limit is in single root-mean-square noise size; the peak-to-peak noise (width of the noise band) is about twice that value. The noise is determined by taking the smallest value from four 5% regions at the left end of the spectrum. Any filter cutoff at the end will decrease the apparent noise in the spectrum, and therefore increase the noise limit in the central part of the spectrum. Because of the particular algorithm used in this macro, signals at the left end of the spectrum should not affect the result of `noislm`.

Arguments: `max_noise` is the maximum root-mean-square size, in mm, of the noise. The default is 2.

Examples: `noislm`
`noislm(5)`

See also: *Getting Started*

Related: `vs` Vertical scale (P)
`vsadj` Automatic vertical scale adjustment (M)
`vsadjc` Automatic vertical scale adjustment for ^{13}C spectra (M)
`vsadjh` Automatic vertical scale adjustment for ^1H spectra (M)

np Number of data points (P)

Description: Sets number of data points to be acquired. Generally, `np` is a *dependent* parameter and is calculated automatically when `sw` or `at` is changed. If a particular number of data points is desired, `np` can be entered, in which case `at` becomes the dependent parameter and is calculated based on `sw` and `np`.

Values: On *GEMINI 2000*, `np` is constrained to be a multiple of 64. Upper limit for `np` on broadband systems is 128,000, the limit on $^1\text{H}/^{13}\text{C}$ systems is 64,000. These limits can be doubled with the `setlimit` command if `dp='n'`.

On *MERCURY-VX* and *MERCURY*, 64 to 128,000, in steps of 64 (`dp` does not affect the limit because on *MERCURY-VX* and *MERCURY* `dp` is always 'y').

On systems other than the *GEMINI 2000*, `np` is constrained to be a multiple of 2 (Acquisition Controller or Pulse Sequence Controller board) or a multiple of 64 (Output board). (See the `acquire` statement in the manual *VNMR User Programming* for a description of these boards.)

See also: *Getting Started*

Related: `at` Acquisition time (P)
`dp` Double precision (P)

`setlimit` Set limits of a parameter in a tree (C)
`sw` Spectral width in directly detected dimension (P)

npoint **Number of points for fp peak search (P)**

Description: If `npoint` is defined in the current parameter set and has a value, it determines the range of data points over which the `fp` command searches for a maximum for each peak. To create `npoint` and give it a value other than the default, enter `create('npoint', 'integer') npoint=x`, where `x` is the new value.

Values: 1 to `fn/4`. The default is 2.

See also: *User Guide: Liquids NMR*

Related: `create` Create new parameter in a parameter tree (C)
`fn` Fourier number in directly detected dimension (P)
`fp` Find peak heights (C)

nrecords **Determine number of lines in a file (M)**

Syntax: `nrecords(file):$number_lines`

Description: Returns the number of lines (or records) in a file.

Arguments: `file` is the name of the file.

`$number_lines` returns the number of lines in the named file.

Examples: `nrecords(userdir+'markld.out'):$num`

See also: *VNMR User Programming*

ns **Number of slices to be acquired (P)**

Applicability: Systems with imaging capabilities.

Description: Sets the number of slices to be acquired for multislice sequences.

Values: 1 to desired number, in integer steps.

See also: *User Guide: Imaging*

`ne` Number of echoes to be acquired (P)

nscans **Number of scout scan or real scan repetitions (P)**

Applicability: Systems with LC-NMR accessory.

Description: For on-flow applications, `nscans` is set to the number of repetitions of the scout scan or real scan process to be performed (based on the time duration of the LC run). In stopped-flow applications, `nscans` must be set to a number that is greater than or equal to the number of peaks to be analyzed or detected. If `nscans` does not exist, the `parlc` macro can create it.

See also: *User Guide: Liquids NMR*

Related: `curscan` Scan currently in progress (P)
`parlc` Create LC-NMR parameters (M)

nt **Number of transients (P)**

Description: Sets the number of transients to be acquired (i.e., the number of repetitions or scans performed to make up the experiment or FID).

Values: 1 to $1e9$ (for *MERCURY-VX* and *MERCURY*, the hardware limits `nt` to 16e6). For an indefinite acquisition, set `nt` to a very large number such as $1e9$.

See also: *Getting Started; User Guide: Imaging*

ntrig **Number of trigger signals to wait before acquisition (P)**

Applicability: Systems with LC-NMR accessory.

Description: Sets the number of trigger signals from the LC to wait for on the external gate line before beginning acquisition. If `ntrig` is 0 or the parameter does not exist, the external gate signal is ignored. If `ntrig` does not exist, the `parlc` macro can create it. `ntrig` is not normally entered by the user.

See also: *User Guide: Liquids NMR*

Related: `parlc` Create LC-NMR parameters (M)

ntype3d **Specify whether f_1 or f_2 display expected to be N-type (P)**

Applicability: All systems; however, although `ntype3d` is available on *GEMINI 2000* systems, such systems can only process 3D data and cannot acquire 3D data.

Description: Indicates whether the f_1 or f_2 display is expected to be N-type, that is, opposite to the sense of precession defined by f_3 , under normal 3D processing conditions.

Values: 'yn' specifies that f_1 is expected to have an N-type display under normal 3D processing conditions.

'ny' specifies that f_2 is expected to have an N-type display under normal 3D processing conditions.

'yy' specifies that both f_1 and f_2 are expected to have N-type displays under normal 3D processing conditions. Setting `ntype3d = 'yy'` changes the sense of precession in f_1 and f_2 by negating the imaginary portion of the t_1 and t_2 interferograms prior to Fourier transformation.

See also: *Getting Started*

Related: `fiddc3d` 3D time-domain dc correction (P)
`ft3d` Perform a 3D Fourier transform on a 3D FID data set (M,U)
`ptspec3d` Region-selective 3D processing (P)
`specdc3d` 3D spectral dc correction (P)
`ssfilter` Full bandwidth of digital filter to yield a filtered FID (P)
`ssorder` Order of polynomial to fit digitally filtered FID (P)
`rftype` Type of rf generation

numrcvrs **Number of receivers in the system (P)**

Applicability: Systems with multiple receivers.

Description: An integer giving the number of receivers installed in the system. `numrcvrs` is set from the config panel by the `vnmr1` user.

Related: `rcvrs` Which receivers to use (P)

numreg **Return the number of regions in a spectrum (C)**

Syntax: `numreg: number_regions`

Description: Returns the number of regions in a spectrum previously divided by the `region` command, by manual means using the `z` command, or by the Resets button in `ds`. A *region* is the area between two reset points in integral mode, with every other reset point designating the start of a *baseline* region and not included in the count of regions.

Arguments: `number_regions` returns the number of peak regions in the spectrum.

Examples: `numreg:$num`

See also: *VNMR User Programming*

Related:	<code>ds</code>	Display a spectrum (C)
	<code>getreg</code>	Get frequency limits of a specified region (C)
	<code>region</code>	Divide spectrum into regions (C)
	<code>z</code>	Add integral reset point at cursor position (C)

numrfch **Number of rf channels (P)**

Description: Holds the number of rf channels available. The value is set with the Number of RF Channels label in the CONFIG window (opened from `config`). `numrfch` represents the hardware in the system. For example, if the last experiment used the second decoupler, `numrfch` is set to 2. The software then leaves the second decoupler on if it was on and leaves it off if it was off.

CAUTION: Do not reset `numrfch` to eliminate the use of a channel. See the description of `dn2` and `dn3` for the method to disable channels.

Values: On *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*: 2. On other systems: 1, 2, 3, 4, or 5. The value does not include the lock channel. For *UNITY INOVA*, the fifth channel can only be used with the deuterium decoupler channel.

See also: *VNMR and Solaris Software Installation*

Related:	<code>config</code>	Display current configuration and possibly change it (M)
	<code>dn2</code>	Nucleus for the second decoupler (P)
	<code>dn3</code>	Nucleus for the third decoupler (P)
	<code>dn4</code>	Nucleus for the fourth decoupler (P)

nv **Number of phase encode steps (P)**

Applicability: Systems with imaging capabilities.

Description: The number of phase encode steps for the first indirectly detected dimension in a multidimensional imaging or CSI experiment.

Values: 0 to the desired number, in powers of 2. Typical values are 0, 64, 128, and 256.

See also: *User Guide: Imaging*

off **Make a parameter inactive (C)**

Syntax: `off(parameter<,tree>)`

Description: Turns off an active parameter in any tree.

Arguments: `parameter` is the name of the parameter.

`tree` is type of parameter tree: 'current', 'global', 'processed', or 'systemglobal'. The default is 'current'. Refer to the `create` command for more information on the types of trees.

Examples: `off('gf')`
`off('n','global')`

See also: *VNMR User Programming*

Related: `create` Create new parameter in a parameter tree (C)
`on` Make a parameter active or test its state (C)

offset **Calculate frequency offset of cursor (M)**

Applicability: Systems with imaging capabilities.

Syntax: `offset('<silent>')<:parameter>`

Description: Reads value of the cursor parameter `cr`, and then calculates and displays the transmitter offset value, in Hz, that places the cursor position on resonance.

Arguments: '`silent`' is a keyword to not display the frequency offset value. The default is to display the value.

`parameter` is a variable (such as the parameter `tof` in the example below) that, if present, is loaded with the calculated offset frequency value.

Examples: `offset`
`offset('<silent>'):tof`

See also: *User Guide: Imaging*

Related: `cr` Current cursor position (P)

on **Make a parameter active or test its state (C)**

Syntax: `on(parameter<,tree>)<:$active>`

Description: Turns on an inactive parameter in any tree or tests if a parameter is active.

Arguments: `parameter` is the name of the parameter to make active or to test.

`tree` is type of parameter tree: 'current', 'global', 'processed', or 'systemglobal'. The default is 'current'. Refer to the `create` command for more information on the types of trees.

`$active` is 1 if the parameter is active, or is 0 if it is not active. Adding a return argument makes `on` conduct only a test of whether the specified parameter is active and does *not* turn on the parameter if it is inactive.

Examples: `on('lb'):$ison`
`on('gain','global')`

See also: *VNMR User Programming*

Related: **create** Create new parameter in a parameter tree (C)
off Make a parameter inactive (C)

opx Open shape definition file for Pbox (M)

Syntax: `opx<(name<.ext>)>`

Description: Opens the pulse shape/pattern definition input file `shapelib/Pbox.inp` for the Pbox software and writes the file header.

Arguments: `name` is the name of the output shape file.
`ext` is a file name extension that specifies the file type.

Examples: `opx`
`opx('newfile.DEC')`

See also: *User Guide: Liquids NMR*

Related: **Pbox** Pulse shaping software (U)

orient Slice plane orientation (P)

Applicability: Systems with imaging capabilities.

Description: Controls the orientation of the slice plane in the gradient reference frame.

Values: A three-character string with any permutation of the letters `x`, `y`, `z`, and `n`: `'xyz'`, `'zyx'`, `'nzx'`, etc. The permutation chosen determines the orientation of the slice plane. The first character is the identity of the readout gradient, the second character is the identity of the phase encoding gradient, and the third character is the identity of the slice selection gradient. The character `n` causes no gradient to be sent, which is used to avoid zeroing values.

For imaging modules, only `'sag'` (sagittal), `'trans'` (transverse), `'cor'`, and `'oblique'` are used. The choice `'oblique'` is not user-enterable. Only the macro **imprep** can set up oblique imaging.

See also: *User Guide: Imaging*

Related: **imprep** Set up rf pulses, imaging, and voxel selection gradients (M)

oscoef Digital filter coefficients for oversampling (P)

Description: Specifies number of coefficients used in the digital filter. If `oscoef` does not exist in the current experiment, enter **addpar** (`'oversamp'`) to add it. **addpar** (`'oversamp'`) creates digital filtering and oversampling parameters `def_osfilt`, `filtfile`, `oscoef`, `osfb`, `osfilt`, `oslsfrq`, and `oversamp`.

Values: For inline DSP (`dsp='i'`), the default is $7.5 * \text{oversamp}$. A larger number of coefficients gives a filter with sharper cutoffs; a smaller number gives a filter with more gradual cutoffs. The value of `oscoef` does not need to be changed when `oversamp` is changed because `oscoef` is automatically adjusted by VNMR to give filter cutoffs that are the same regardless of the value of `oversamp`.

For real-time DSP (`dsp='r'`), the number of coefficients is not adjustable but is determined by the hardware.

See also: *Getting Started*

Related: **addpar** Add selected parameters to current experiment (M)
dsp Type of DSP for data acquisition (P)

<code>filtfile</code>	File of FIR digital filter coefficients (P)
<code>osfb</code>	Digital filter bandwidth for oversampling (P)
<code>oslsfrq</code>	Bandpass filter offset for oversampling (P)
<code>oversamp</code>	Oversampling factor for acquisition (P)
<code>paros</code>	Create additional parameters used by oversampling (M)

osfb **Digital filter bandwidth for oversampling (P)**

Description: Specifies bandwidth of the digital filter used for oversampling. If `osfb` does not exist in the current experiment, enter `addpar('oversamp')` to add it. `addpar('oversamp')` creates digital filtering and oversampling parameters `def_osfilt`, `filtfile`, `oscoef`, `osfb`, `osfilt`, `oslsfrq`, and `oversamp`.

Values: Number, in Hz. A value less than `sw/2` rejects frequencies at the edges of the spectrum; a value greater than `sw/2` aliases noise and signals at frequencies outside of $\pm sw/2$.

'n' sets the bandwidth to `sw/2`.

See also: *Getting Started*

Related:	<code>addpar</code>	Add selected parameters to current experiment (M)
	<code>def_osfilt</code>	Default value of <code>osfilt</code> (P)
	<code>filtfile</code>	File of FIR digital filter coefficients (P)
	<code>oscoef</code>	Digital filter coefficients for oversampling (P)
	<code>osfilt</code>	Oversampling filter for real-time DSP (P)
	<code>oslsfrq</code>	Bandpass filter offset for oversampling (P)
	<code>oversamp</code>	Oversampling factor for acquisition (P)
	<code>paros</code>	Create additional parameters used by oversampling (M)
	<code>sw</code>	Spectral width in directly detected dimension (P)

osfilt **Oversampling filter for real-time DSP (P)**

Applicability: Systems with real-time DSP.

Description: Sets the type of real-time digital filter to be used on systems equipped with the real-time DSP hardware option. `osfilt` is normally set automatically by the software based on the user's global parameter `def_osfilt`, so that `osfilt` only needs to be changed if a particular experiment is to be run with a different digital filter than the default.

Values: 'a' or 'A' for the AnalogPlus™ digital filter.

'b' or 'B' for the brickwall digital filter.

' ' (null string) causes `osfilt` to be set to the value contained in the `def_osfilt` when an acquisition is initiated (with `go`, for example).

See also: *Getting Started*

Related:	<code>def_osfilt</code>	Default value of <code>osfilt</code> (P)
	<code>dsp</code>	Type of DSP for data acquisition (P)

oslsfrq **Bandpass filter offset for oversampling (P)**

Description: Selects a bandpass filter that is not centered about the transmitter frequency. In this way, `oslsfrq` works much like `lsfrq`. If `oslsfrq` does not exist in the current experiment, enter `addpar('oversamp')` to add it.

`addpar('oversamp')` creates digital filtering and oversampling parameters `def_osfilt`, `filtfile`, `oscoef`, `osfb`, `osfilt`, `oslsfrq`, and `oversamp`.

Values: Number, in Hz. A positive value selects a region upfield from the transmitter frequency; a negative value selects a downfield region.

See also: *Getting Started*

Related:	<code>addpar</code>	Add selected parameters to current experiment (M)
	<code>def_osfilt</code>	Default value of <code>osfilt</code> (P)
	<code>filtfile</code>	File of FIR digital filter coefficients (P)
	<code>fsq</code>	Frequency-shifted quadrature detection(P)
	<code>lsfrq</code>	Frequency shift of the <code>fn</code> spectrum in Hz (P)
	<code>oscoef</code>	Digital filter coefficients for oversampling (P)
	<code>osfb</code>	Digital filter bandwidth for oversampling (P)
	<code>osfilt</code>	Oversampling filter for real-time DSP (P)
	<code>oversamp</code>	Oversampling factor for acquisition (P)
	<code>paros</code>	Create additional parameters used for oversampling (M)

overrange **Frequency synthesizer overrange (P)**

Applicability: UNITY *INOVA*, UNITY *plus*, UNITY, and VXR-S systems with optional version X46 of the PTS frequency synthesizer.

Description: Configures whether an rf channel has version X46 of the PTS frequency synthesizer. The value for each channel is set using the label Frequency Overrange in the CONFIG window (opened from `config`).

Values: 0 indicates that channel does not have frequency overrange (Not Present choice from the CONFIG window).

10000 or 100000 indicates that channel has frequency overrange (10000 Hz or 100000 Hz choice from the CONFIG window). If Frequency Overrange is set to 10000 Hz or 100000 Hz, the value of Latching for that rf channel must also be set to Present in the CONFIG window. When set to either 10000 Hz or 100000 Hz, `overrange` guarantees a range of phase-continuous frequency jumps of at least 10 or 100 kHz in each jump direction. Continuous phase can be maintained over 20 or 200 kHz if the system doubles to derive the frequency.

See also: *VNMR and Solaris Software Installation*

Related:	<code>config</code>	Display current configuration and possibly change it (M)
	<code>latch</code>	Frequency synthesizer latching (P)

oversamp **Oversampling factor for acquisition (P)**

Description: Specifies the oversampling factor for the acquisition. With inline digital filtering (`dsp= ' i '`), `np*oversamp` data points are acquired at a rate of `sw*oversamp`. The data is then transferred to the host computer, digitally filtered, and downsampled to give `np` points and a spectral width of `sw`.

With real-time digital filtering (`dsp= ' r '`), the oversampling, digital filtering, and downsampling all occur as each data point is collected, so that only `np` data points are ever stored in the acquisition computer memory and subsequently transferred to the host computer.

If `oversamp` does not exist in the current experiment, enter the command `addpar(' oversamp ')` to add it. `addpar(' oversamp ')` creates digital filtering and oversampling parameters `def_osfilt`, `filtfile`, `oscoef`, `osfb`, `osfilt`, `oslsfrq`, and `oversamp`.

If `oversamp` is set to a number, then that number represents the amount of oversampling to apply when collecting the data. The `oversamp` value is automatically calculated whenever `sw` is changed, provided `oversamp` is not set to ' n '. That is the distinction between `oversamp= ' n '` and

`oversamp=1`. In both cases, no oversampling will be used. This occurs, for example, if the `sw` parameter is greater than half the maximum spectral width. However, if `sw` is reduced so that oversampling is possible, then if `oversamp` is set to 'n', `oversamp` will remain set to 'n' and oversampling will not occur. On the other hand, if `oversamp` is set to 1, then `oversamp` is recalculated and oversampling will occur. Therefore, the `oversamp` parameter accurately represents whether oversampling is performed for a data set. When `oversamp` is automatically determined based on a change to `sw`, it is set to the maximum possible oversampling factor. The value of `oversamp` can be manually reset.

Note that setting `oversamp` greater than 1 means oversampling is selected for the experiment. However, if the oversampling facility is not present in the system (i.e., `dsp='n'`), then the `oversamp` parameter is automatically reset to 1, indicating that no oversampling will be performed.

Two other experiment local parameters reflect whether DSP is used during the acquisition of a data set:

- `fb` is set to Not Active if DSP is used.
- `oscoef` reflects whether real-time (`dsp='r'`) or inline (`dsp='i'`) DSP was used. If real-time, `oscoef` is set to Not Active. If inline, `oscoef` is set to the value used by the inline algorithm.

Values: Number less than or equal to 68. For inline DSP, `sw*oversamp` and `np*oversamp` are limited by the values in the following table:

<i>System</i>	<i>Maximum sw*oversamp</i>	<i>Maximum np*oversamp</i>
UNITY INOVA	500 kHz	2M
MERCURY-VX & MERCURY	100 kHz	128K
UNITYplus, UNITY, VXR-S	100 kHz	512K
GEMINI 2000 Broadband	100 kHz	128K
GEMINI 2000 ¹ H/ ¹³ C	23 kHz	64K

The maximum `np*oversamp` is given for double precision data (`dp='y'`). For `dp='n'`, multiply this value by 2.

'n' causes normal acquisition to be done without digital filtering.

See also: *Getting Started*

Related:	<code>addpar</code>	Add selected parameters to current experiment (M)
	<code>def_osfilt</code>	Default value of <code>osfilt</code> parameter (P)
	<code>dp</code>	Double precision (P)
	<code>dsp</code>	Type of DSP for data acquisition (P)
	<code>fb</code>	Filter bandwidth (P)
	<code>filtfile</code>	File of FIR digital filter coefficients (P)
	<code>fsq</code>	Frequency-shifted quadrature detection (P)
	<code>np</code>	Number of data points (P)
	<code>oscoef</code>	Digital filter coefficients for oversampling (P)
	<code>osfb</code>	Digital filter bandwidth for oversampling (P)
	<code>osfilt</code>	Oversampling filter for real-time DSP (P)
	<code>oslsfrq</code>	Bandpass filter offset for oversampling (P)
	<code>paros</code>	Create additional parameters used by oversampling (M)
	<code>sw</code>	Spectral width in directly detected dimension (P)

P

- p1** **Enter pulse width for p1 in degrees (C)**
- Syntax: `p1 (flip_angle<, 90_pulse_width>)`
- Description: Calculates the flip time, in μs , given a desired flip angle and the 90° pulse. The value is entered into the pulse width parameter `p1`.
- Arguments: `flip_angle` is the desired flip angle, in degrees.
`90_pulse_width` is the 90° pulse, in μs . The default is the value of parameter `pw90` if it exists.
- Examples: `p1 (30)`
`p1 (90, 12.8)`
- See also: *Getting Started*
- Related: `ernst` Calculate the Ernst angle pulse (C)
`p1` First pulse width (P)
`pw90` 90° pulse width (P)
-
- p1** **First pulse width (P)**
- Description: Length of first pulse in the standard two-pulse sequence.
- Values: On *MERCURY-VX* and *MERCURY* systems: 0, 0.2 μs to 4095 μs . On *GEMINI 2000* systems: 0, 0.2 to 4095 μs , in 100-ns steps. On systems with a Data Acquisition Controller board: 0, 0.1 to 8190 μs , in 12.5-ns steps. On systems with Pulse Sequence Controller or Acquisition Controller boards: 0, 0.2 to 8190 μs , in 25-ns steps. On systems with Output boards: 0, 0.2 to 8190 μs , in 0.1- μs steps. (Refer to the `acquire` statement in the manual *VNMR User Programming* for a description of these boards.)
- See also: *Getting Started*
- Related: `p1` Enter pulse width *p1* in degrees (C)
-
- p1pat** **Shape of excitation pulse (P)**
- Description: Specifies the shape of pulse `p1` when used in imaging experiments.
- Values: 'hard', 'sinc', 'gauss', 'sech', 'sine', or any shape resident in the system pulse shape library or libraries.
- See also: *User Guide: Imaging*
- Related: `p1` First pulse width (P)
`pwpat` Shape of refocusing pulse (P)
-
- p2** **180° refocus pulse width (P)**
- Applicability: Systems with imaging capabilities.
- Description: Sets the length of the 180° refocus rf pulse.
- Values: Number, in μs .
- See also: *User Guide: Imaging*
- Related: `p1` First pulse width (P)
`p2pat` RF pulse pattern of pulse `p2` (P)

- p2pat** **RF pulse pattern of 180° refocus pulse p2 (P)**
 Applicability: Systems with imaging capabilities.
 Description: Contains a string for the shape of the 180° refocus pulse **p2**.
 See also: *User Guide: Imaging*
 Related: **p2** 180° refocus pulse width (P)
- p2pul** **Set up sequence for PFG testing (M)**
 Applicability: Systems with the pulsed field gradient (PFG) module. *This sequence is not for NMR applications.*
 Syntax: **p2pul**
 Description: Sets up the PFG two-pulse sequence, a system checkout sequence for PFG installation. Several modes are controlled by the **cmd** parameter.
 - **cmd**= 'twinkle' sequentially addresses DACs 0 through 4. On the gradient channel interface, lights become a slow binary counter.
 - **cmd**= 'pulse' makes a pulse of value **gzlv11** for a time **gt1**.
 - **cmd**= 'bipulse' makes a pulse of value **gzlv11** for a time **gt1** followed by a pulse of value **-gzlv11** for a time **gzlv11**.
 For other modes, see the PFG installation manual.
 See also: *Pulsed Field Gradient Modules Installation*
- p31** **Automated phosphorus acquisition (M)**
 Syntax: **p31**<(solvent)>
 Description: Prepares parameters for automatically acquiring a standard ³¹P spectrum. The parameter **wexp** is set to 'procplot' for standard processing. If **p31** is used as the command for automation via the **enter** command, then the macro **au** is supplied automatically and should not be entered on the MACRO line of the **enter** program. However, it is possible to customize the standard **p31** macro on the MACRO line by following it with additional commands and parameters. For example, **p31 nt=1** will use the standard **p31** setup but with only one transient.
 Arguments: **solvent** is the name of the solvent. The default is CDC13. In automation mode, the solvent is supplied by the **enter** program.
 Examples: **p31**
p31('DMSO')
 See also: *Getting Started; User Guide: Liquids NMR*
 Related: **au** Submit experiment to acquisition and process data (M)
enter Enter sample information for automation run (C)
p31p Process 1D phosphorus spectra (M)
procl1d Processing macro for simple, non-arrayed 1D spectra (M)
procplot Automatically process FIDs (M)
wexp When experiment completes (P)
- p31p** **Process 1D phosphorus spectra (M)**
 Syntax: **p31p**
 Description: Processes non-arrayed 1D ³¹P spectra using a set of standard macros. **p31p** is called by the **procl1d** macro but can also be used directly. Fully automatic processing (up to a point where a spectrum could be plotted) is provided:

Fourier transformation (using preset weighting functions), automatic phasing (**aphx** macro), automatic integration (**integrate** macro, if required only), vertical scale adjustment (**vsadjc** macro), avoiding excessive noise (**noislm** macro), threshold adjustment (**thadj** macro), and referencing to the TMS signal, if present (**tmsref** macro).

See also: *Getting Started; User Guide: Liquids NMR*

Related:	aphx	Perform and check automatic phasing (M)
	integrate	Automatically integrate 1D spectrum (M)
	noislm	Avoids excessive noise (M)
	p31	Automated phosphorus acquisition (M)
	procl1d	Automatically process non-arrayed 1D fids (M)
	thadj	Adjust threshold (M)
	tmsref	Reference spectrum to TMS line (M)
	vsadjc	Adjust vertical scale for carbon spectra (M)

pa Set phase angle mode in directly detected dimension (C)

Syntax: **pa**

Description: Selects the phase angle mode by setting the parameter **dmg**= 'pa'. In the *phase angle display mode*, each real point in the displayed spectrum is calculated from the phase angle of the real and imaginary points comprising each respective complex data point. The phase angle also takes into account the phase parameters **rp** and **lp**.

For 2D data, if **pmode**= 'partial' or **pmode**= '' (two single quotes with no space in between), **pa** has an effect on the data prior to the second Fourier transform. If **pmode**= 'full', **pa** acts in concert with the commands **pa1**, **av1**, **pwr1**, or **ph1** to yield the resultant contour display for the 2D data.

See also: *User Guide: Liquids*

Related:	av	Set abs. value mode in directly detected dimension (C)
	dmg	Data display mode in directly detected dimension (P)
	ft	Fourier transform 1D data (C)
	ft1d	Fourier transform along f ₂ dimension (C)
	ft2d	Fourier transform 2D data (C)
	lp	First-order phase in directly detected dimension (P)
	pa1	Set phase angle mode in 1st indirectly detected dimension (C)
	ph	Set phased mode in directly detected dimension (C)
	pmode	Processing mode for 2D data (P)
	pwr	Set power mode in directly detected dimension (C)
	pwr1	Set power mode in 1st indirectly detected dimension (C)
	rp	Zero-order phase in directly detected dimension (P)
	wft	Weight and Fourier transform 1D data (C)
	wft1d	Weight and Fourier transform f ₂ of 2D data (M)
	wft2d	Weight and Fourier transform 2D data (M)

pa1 Set phase angle mode in 1st indirectly detected dimension (C)

Syntax: **pa1**

Description: Selects the phase angle spectra display mode along the first indirectly detected dimension by setting the parameter **dmg1** to the string value 'pa1'. If the parameter **dmg1** does not exist, **pa1** will create it and set it to 'pa1'.

In the phase angle mode, each real point in the displayed trace is calculated from the phase angle of the real and imaginary points comprising each respective complex data point. For hypercomplex data, the phase angle uses the real-real

and imaginary-real points from each respective hypercomplex data point. The phase angle also takes into account the phase parameters `rp1` and `lp1`.

The `pa1` command is only needed if mixed-mode display is desired. If the parameter `dmg1` does not exist or is set to the null string, the display mode along the first indirectly detected dimension defaults to the display mode of the directly detected dimension (characterized by the parameter `dmg`). For the contour display of multidimensional data, the result of `pa1` is the same as for traces provided that `pmode='partial'` or `pmode=''`.

See also: *User Guide: Liquids*

Related:	<code>av1</code>	Set abs. value mode in 1st indirectly detected dimension (C)
	<code>dmg1</code>	Data display mode in 1st indirectly detected dimension (P)
	<code>lp1</code>	First-order phase in 1st indirectly detected dimension (P)
	<code>pa</code>	Set phase angle mode in directly detected dimension (C)
	<code>ph1</code>	Set phased mode in 1st indirectly detected dimension (C)
	<code>pmode</code>	Processing mode for 2D data (P)
	<code>pwr1</code>	Set power mode in 1st indirectly detected dimension (C)
	<code>rp1</code>	Zero-order phase in 1st indirectly detected dimension (P)

pacosy **Plot automatic COSY analysis (C)**

Syntax: `pacosy`

Description: Automatically analyzes and plots a COSY data set with `fn=fn1` and `sw=sw1`. Symmetrization of the data with the command `foldt` is recommended, but not required. First, select a proper threshold and perform a 2D line listing with the command `l12d`. Next, plot the 2D data with the contour plot command `pcon`; leaving enough room at the left side of the plot for the connectivity table. Then, `pacosy` will analyze the data and plot the connectivities on the plotter. `pacosy` gets its input from the file `l12d.out` in the current experiment directory. The command `acosy` performs the same analysis and displays the connectivities on the screen.

See also: *Getting Started; User Guide: Liquids NMR*

Related:	<code>acosy</code>	Automatic analysis of COSY data (C)
	<code>fn</code>	Fourier number in directly detected dimension (P)
	<code>fn1</code>	Fourier number in 1st indirectly detected dimension (P)
	<code>foldt</code>	Fold COSY-like spectrum along diagonal axis (C)
	<code>hcosy</code>	Automated proton and COSY acquisition (M)
	<code>l12d</code>	Automatic and interactive 2D peak picking (C)
	<code>pcon</code>	Plot contours on plotter (C)
	<code>relayh</code>	Set up parameters for COSY pulse sequence (M)
	<code>sw</code>	Spectral width in directly detected dimension (P)
	<code>sw1</code>	Spectral width in 1st indirectly detected dimension (P)

pad **Preacquisition delay (P)**

Description: Each NMR experiment starts with a single delay time equal to `pad` over and above the delay `d1` that occurs before each transient. Normally, `pad` is set to a small, nominal time (0.5 seconds) to allow any hardware changes that may be required at the start of the acquisition to “settle in.” During experiments in which the temperature is changed, the acquisition starts `pad` seconds after the temperature regulation system comes to regulation. Since the sample temperature does not actually come to equilibrium for some time after that, it is generally desirable to increase `pad` to perhaps 300 seconds. This is especially true when running experiments involving arrays of temperatures. The `pad`

parameter is most useful for running kinetics experiments. For example, `pad=0,3600,3600,3600,3600` will run an experiment immediately when `go` is typed (`pad=0`), then wait an hour (3600 seconds), run the second experiment, etc.

Values: On *GEMINI 2000* systems: 0 to 4095, in seconds.

On systems other than *GEMINI 2000*: 0 to 8190, in seconds.

See also: *Getting Started; User Guide: Liquids NMR*

Related: `d1` First delay (P)
`go` Submit experiment to acquisition (C)

padept Perform adept analysis and plot resulting spectra (C)

Syntax: `padept<(<'noll'><,<'coef'><,<'theory'>>>`

Description: Performs the `adept` analysis and plots the resulting spectra with a scale and the assigned line listing. Leave enough space at the left end of the display for the line list.

Arguments: The following arguments can be supplied in any order:

'noll' is a keyword that specifies no line listing.

'coef' is a keyword that causes the combination coefficients to be printed.

'theory' is a keyword that causes the theoretical coefficients rather than optimized coefficients to be used.

Examples: `padept('noll','coef')`

See also: *User Guide: Liquids NMR*

Related: `adept` Automatic DEPT analysis and spectrum editing (C)
`autodept` Automated complete analysis of DEPT data (M)
`cdept` Automated carbon and DEPT acquisition (C)
`deptproc` Process DEPT data (M)
`hdept` Automated proton, carbon, and DEPT acquisition (C)
`pldept` Plot DEPT data, edited or unedited (M)

page Submit plot and change plotter page (C)

Syntax: `page<(number_pages<,<'clear' | file>>>`

Description: Submits the current plotter file, which has been created by all previous plotter commands, and changes the paper after the plot has been completed. Actual plotting is controlled by the `vnmrplot` script in the `bin` subdirectory of the VNMR system directory. The `page` command can also clear the current plotter file or save the data to a specified file name.

Arguments: `number_pages` is the number of pages to move the plotter forward. The default is 1. If `number_pages` is 0, `page` submits the plot but does not change the paper.

'clear' is a keyword to clear the plot made thus far; that is, clear the data in the current plotter file.

`file` is the name of a file to save the plot for import into a document. If the file already exists, it is overwritten.

Alternate: Page button in the 1D Plotting Menu.

Examples: `page`
`page(0)`

```
page('clear')
page('myplotfile')
```

See also: *Getting Started*

Related: [vnmrplot](#) Plot files (U)

pap Plot out “all” parameters (C)

Syntax: `pap(<template><, ><x><, y><, character_size>)>`

Description: Plots a parameter list containing “all” parameter names and values.

Arguments: `template` is the name of a template that controls the display. The default is the string parameter `ap`, which can be modified using `paramvi('ap')`. See the manual *VNMR User Programming* for rules on building a template.

`x` is the starting position in the x direction of the plot on the paper, in mm. The default is a preset value.

`y` is the starting position in the y direction of the plot on the paper, in mm. If `y` is specified, the `x` position must be also. The default is a preset value.

`character_size` is the character size of the list and is specified as a multiplier. The default is 0.70 (not available on all plotters or printers acting as plotters).

Alternate: All Params button in the 1D Plotting Menu, or All Params button in the 2D Plotting Menu.

Examples: `pap`
`pap(wcmax-40)`
`pap(10,wc2max*.9)`
`pap('newpap',wcmax-50,100,1.4)`

See also: *Getting Started, VNMR User Programming*

Related: [ap](#) Print out “all” parameters (C)
[ap](#) “All” parameters display control (P)
[hpa](#) Plot parameters on special preprinted chart paper (C)
[paramvi](#) Edit a variable and its attributes using `vi` text editor (M)
[ppa](#) Plot a parameter list in “English” (M)

par2d Create 2D acquisition, processing, and display parameters (M)

Syntax: `par2d`

Description: Creates the acquisition parameters `ni`, `sw1`, and `phase`, which can be used to acquire a 2D data set. `par2d` also creates any missing processing and display parameters for the `ni` (or second) dimension, including `flcoef`, `reffrq1`, `refpos1`, and `refsource1`. The `par2d` macro is functionally the same as `addpar('2d')`.

See also: *User Guide: Liquids NMR*

Related: [addpar](#) Add selected parameters to the current experiment (M)
[flcoef](#) Coefficient to construct F1 interferogram (P)
[ni](#) Number of increments in 1st indirectly detected dimension (P)
[phase](#) Phase selection (P)
[reffrq1](#) Reference frequency of reference line in 1st indirect dimension (P)
[refpos1](#) Position of reference line in 1st indirect dimension (P)
[refsource1](#) Center frequency in 1st indirect dimension (P)
[set2d](#) General setup for 2D experiments (M)
[sw1](#) Spectral width in 1st indirectly detected dimension (P)

par3d Create 3D acquisition, processing, and display parameters (M)

Syntax: `par3d`

Description: Creates the acquisition parameters `ni2`, `sw2`, `d3`, and `phase2` that can be used to acquire a 3D data set. `par3d` also creates any missing processing or display parameters for the `ni2` (or third) dimension, including `f2coef`, `fiddc3d`, `specdc3d`, and `ptspec3d`. The `par3d` macro is functionally the same as `addpar('3d')`.

See also: *User Guide: Liquids NMR*

Related:

<code>addpar</code>	Add selected parameters to the current experiment (M)
<code>d3</code>	Incremented delay in 2nd indirectly detected dimension (P)
<code>f2coef</code>	Coefficient to construct F2 interferogram (P)
<code>fiddc3d</code>	3D time-domain dc correction (P)
<code>ni2</code>	Number of increments in 2nd indirectly detected dimension (P)
<code>phase2</code>	Phase selection for 3D acquisition (P)
<code>ptspec3d</code>	Region-selective 3D processing (P)
<code>specdc3d</code>	3D spectral dc correction (P)
<code>sw2</code>	Spectral width in 2nd indirectly detected dimension (P)

par3rf Get display templates for 3rd rf channel parameters (M)

Applicability: Systems with a second decoupler.

Syntax: `par3rf`

Description: Retrieves the `dg2` and modified `ap` display templates from the parameter set `s2pul3rf` in the system `parlib` directory. These two templates support the display of second decoupler acquisition parameters and 3D acquisition and processing parameters.

See also: *VNMR User Programming*

Related:

<code>ap</code>	“All” parameters display control (P)
<code>dg2</code>	Control <code>dg2</code> parameter group display (P)

par4d Create 4D acquisition parameters (M)

Applicability: Systems with a third decoupler.

Syntax: `par4d`

Description: Creates the acquisition parameters `ni3`, `sw3`, `d4`, and `phase3` that can be used to acquire a 4D data set. The `par4d` macro is functionally the same as `addpar('4d')`.

See also: *User Guide: Liquids NMR*

Related:

<code>addpar</code>	Add selected parameters to the current experiment (M)
<code>d4</code>	Incremented delay for 3rd indirectly detected dimension (P)
<code>ni3</code>	Number of increments in 3rd indirectly detected dimension (P)
<code>phase3</code>	Phase selection for 4D acquisition (P)
<code>sw3</code>	Spectral width in 3rd indirectly detected dimension (P)

paramedit Edit a parameter and its attributes with user-selected editor (C)

Syntax: `paramedit(parameter<, tree>)`

Description: Opens a parameter file for editing with a user-selected text editor. The default editor is `vi`. If `vi` is used as the editor, `paramedit` is functionally the same as the `paramvi` command. To select another editor, set the UNIX environmental variable `vnmeditor` to the editor name (change `.login`

line `setenv vnmreditor old_editor` to become `setenv vnmreditor new_editor` (e.g., `setenv vnmreditor emacs`) and make sure a script with the prefix `vnmr_` followed by the name of the editor is placed in the `bin` subdirectory of the VNMR system directory (e.g., `vnmr_emacs`). The script file makes adjustments for the type of graphic interface in use.

Scripts in the software release include `vnmr_vi` and `vnmr_textedit`. To create other scripts, refer to the `vnmr_vi` script for non-window editor interfaces and to `vnmr_textedit` for window-based editor interfaces. The `vnmreditor` variable must be set before starting VNMR.

Arguments: `parameter` is the name of the parameter file to be edited.
`tree` is a keyword for one of the parameter trees 'current', 'global', or 'processed'. The default is 'current'.

Examples: `paramedit('ap')`
`paramedit('b', 'global')`

See also: *Getting Started; VNMR User Programming*

Related: `paramvi` Edit a parameter and its attributes with `vi` editor (M)
`vi` Edit text file with the `vi` text editor (C)

paramvi Edit a parameter and its attributes with vi editor (M)

Syntax: `paramvi (parameter<, tree>)`

Description: Opens a parameter file for editing using the UNIX `vi` text editor. The parameter file contains various attributes of the parameter in a format documented in the manual *VNMR User Programming*. Be sure you understand the format before modifying the parameter because if an error in the format is made, the parameter will not load. When the editor is exited, the modified parameter is reloaded into the system.

Arguments: `parameter` is the name of the parameter file to be edited.
`tree` is a keyword for one of the parameter trees 'current', 'global', or 'processed'. The default is 'current'.

Examples: `paramvi('ap')`
`paramvi('b', 'global')`

See also: *Getting Started; VNMR User Programming*

Related: `create` Create new parameter in a parameter tree (C)
`destroy` Destroy a parameter (C)
`destroygroup` Destroy parameters of a group in a tree (C)
`display` Display parameters and their attributes (C)
`fread` Read parameters from file and load them into a tree (C)
`fsave` Save parameters from a tree to a file (C)
`groupcopy` Copy parameters of group from one tree to another (C)
`paramedit` Edit a parameter and its attributes with user-selected editor (C)
`prune` Prune extra parameters from current tree (C)
`setgroup` Set group of a parameter in a tree (C)
`setlimit` Set limits of a parameter in a tree (C)
`setprotect` Set protection mode of a parameter (C)
`vi` Edit text file with the `vi` text editor (C)

pards Create additional parameters used by downsampling (M)

Syntax: `pards`

Description: Creates the parameters `downsamp`, `dscoef`, `dsfb`, `dslsfrq`, and `filtfile` necessary for digital filtering and downsampling. The `pardss` macro is functionally the same as `addpar` ('downsamp').

See also: *Getting Started*

Related: `addpar` Add selected parameters to current experiment (M)
`downsamp` Downsampling factor applied after digital filtering (P)
`dscoef` Digital filter coefficients for downsampling (P)
`dsfb` Digital filter bandwidth for downsampling (P)
`dslsfrq` Bandpass filter offset for downsampling (P)
`filtfile` File of FIR digital filter coefficients (P)
`movedssw` Set downsampling parameters for selected spectral region (M)

`parfidss` Create parameters for time-domain solvent subtraction (M)

Syntax: `parfidss`

Description: Creates solvent subtraction parameters `ssfilter`, `sslsfrq`, `ssntaps`, and `ssorder`. Entering `addpar` ('ss') is functionally equivalent to `parfidss`.

In a 1D transform, subtraction of the zero-frequency component from the time-domain data, usually in the context of solvent subtraction, is selected by setting `ssorder` and `ssfilter` to desired values and entering `wft`:

- The zfs (zero-frequency suppression) option is selected if both `ssfilter` and `ssorder` are set to a value other than “Not Used.”
- The lfs (low-frequency suppression) option is selected if `ssfilter` is set to a value other than “Not Used” and `ssorder` is set to “Not Used.”
- The zfs and lfs options are both turned off if `ssfilter` is set to “Not Used.”

The zfs option leads to the following series of processing events: (1) the raw FID is frequency-shifted by `sslsfrq` Hz, (2) the raw FID is subjected to a low-pass digital filter, (3) the filtered FID is fit to a polynomial of order `ssorder`, (4) the polynomial function is subtracted from the raw FID, and (5) the resulting FID is frequency-shifted by `-sslsfrq` Hz.

The lfs option does not include a polynomial fit (step 3 of the zfs option), which leads to the following series of processing events: (1) the raw FID is frequency-shifted by `sslsfrq` Hz, (2) the raw FID is subjected to a low-pass digital filter, (3) the filtered FID is directly subtracted from the raw FID, (4) the resulting FID is frequency-shifted by `-sslsfrq` Hz.

The quality of filtering with zfs diminishes rapidly as the solvent peak moves off the exact center of the digital filter. It may be necessary to adjust `lsfrq` or `sslsfrq` to move the solvent peak to within ± 0.2 Hz of the center of the filter to obtain optimal solvent suppression. The lfs option is less sensitive to small offsets, but typically removes or distorts peaks near to the solvent peak.

In a 2D transform, solvent correction to the t_2 FIDs is invoked in the same manner with the `ft1d`, `ft2d`, `wft1d`, and `wft2d` commands and with the `ft2da`, `ft1da`, `wft2da`, and `wft1da` macros.

In a 3D transform, solvent suppression works on t_3 FIDs of 3D spectra just like in the 1D and 2D cases.

See also: *Getting Started; User Guide: Liquids NMR*

Related: `addpar` Add selected parameters to the current experiment (M)
`ft` Fourier transform 1D data (C)
`ft1d` Fourier transform along f_2 dimension (C)

<code>ft2d</code>	Fourier transform 2D data (C)
<code>ft3d</code>	Perform a 3D Fourier transform on a 3D FID data set (M,U)
<code>lsfrq</code>	Frequency shift of the <code>fn</code> spectrum in Hz (P)
<code>ntype3d</code>	N-type peak selection in f_1 or f_2 (P)
<code>ssfilter</code>	Full bandwidth of digital filter to yield a filtered FID (P)
<code>sslsfrq</code>	Center of solvent-suppressed region of spectrum (P)
<code>ssorder</code>	Order of polynomial to fit digitally filtered FID (P)
<code>ssntaps</code>	Number of coefficients to be used in the digital filter (P)
<code>wft</code>	Weight and Fourier transform 1D data (C)

parfix **Update parameter sets (M)**

Syntax: `parfix`

Description: Corrects upper limits, lower limits, and step sizes of a number of parameters in the current experiment. In addition, the template parameter `dgs` is updated. This is automatically done via the macro `fixpar` if the parameter `parversion` is less than 4.3. `parfix` is used by the macro `updatepars` to correct saved data. This macro has been applied to all parameters as of VNMR version 4.3 and should be run on older parameter sets (e.g., `rtpl('pars')` `svp('pars')` update a parameter set named `pars`).

See also: *Getting Started*

Related:	<code>ap</code>	“All” parameters display control (P)
	<code>dgs</code>	Control <code>dgs</code> parameter group display (P)
	<code>fixpar</code>	Correct parameter characteristics in experiment (M)
	<code>parversion</code>	Version of parameter set (P)
	<code>updatepars</code>	Update all parameter sets saved in a directory (M)

parlc **Create parameters for LC-NMR experiments (M)**

Applicability: Systems with LC-NMR accessory.

Syntax: `parlc`

Description: Creates the following parameters used for a variety of LC-NMR experiments: `curscan`, `dtrig`, `inject`, `nscans`, `ntrig`, and `savefile`. The `parlc` macro also creates `ni` and `swl` (if they don't exist) for use in isocratic runs. Finally, it creates a display parameter `dglc`, so that the `dg('dglc')` command (or the equivalent macro `dglc`) can be used to display all the LC-related parameters.

Note that `parlc` can be used without worrying about losing existing values or attributes; if the parameters already exist, they are left untouched.

See also: *User Guide: Liquids NMR*

Related:	<code>curscan</code>	Scan currently in progress (P)
	<code>dglc</code>	Control LC-NMR parameter display (P)
	<code>dtrig</code>	Delay to wait for another trigger or acquire a spectrum (P)
	<code>inject</code>	Trigger the injection of a sample (P)
	<code>nscans</code>	Number of scout/real scan repetitions (P)
	<code>ntrig</code>	Number of trigger signals to wait before acquisition (P)
	<code>savefile</code>	Base file name for saving FIDs or data sets (P)

par112d **Create parameters for 2D peak picking (M)**

Syntax: `par112d`

Description: Creates additional parameters `th2d` and `xdiag` for use with `l12d` 2D peak picking program. `parl12d` is functionally the same as `addpar('l12d')`.

See also: *User Guide: Liquids NMR*

Related: `addpar` Add selected parameters to the current experiment (M)
`l12d` Automatic and interactive 2D peak picking (C)
`th2d` Threshold for integrating peaks in 2D spectra (P)
`xdiag` Threshold for excluding diagonal peaks when peak picking (P)

parlp Create parameters for linear prediction (M)

Syntax: `parlp<(dimension)>`

Description: Creates parametrized options for linear prediction (LP) in the current experiment. The display template for the `dglp` macro is also created if necessary. `parlp` is functionally the same as `addpar('lp')`.

Arguments: `dimension` is the dimension of a multidimensional data set. The default is to create the LP parameters `lpalg`, `lpopt`, `lpfilt`, `lpnupts`, `strtlp`, `lpext`, `strtext`, `lptrace`, and `lpprint`.

`parlp(1)` creates LP parameters `lpalg1`, `lpopt1`, `lpfilt1`, `lpnupts1`, `strtlp1`, `lpext1`, `strtext1`, `lptrace1`, and `lpprint1`. `addpar('lp', 1)` is functionally equivalent to `parlp(1)`.

`parlp(2)` creates LP parameters `lpalg2`, `lpopt2`, `lpfilt2`, `lpnupts2`, `strtlp2`, `lpext2`, `strtext2`, `lptrace2`, and `lpprint2`. `addpar('lp', 2)` is functionally equivalent to `parlp(2)`.

Examples: `parlp`
`parlp(1)`

See also: *Getting Started; User Guide: Liquids NMR*

Related: `dglp` Display group of linear prediction parameters (M)
`lpalg` LP algorithm for `np` dimension (P)
`lpext` LP data extension for `np` dimension (P)
`lpfilt` LP coefficients to calculate for `np` dimension (P)
`lpnupts` LP number of data points for `np` dimension (P)
`lpopt` LP algorithm data extension for `np` dimension (P)
`lpprint` LP print output for `np` dimension (P)
`lptrace` LP output spectrum for `np` dimension (P)
`proc` Type of processing on `np` FID (P)
`proc1` Type of processing on `ni` interferogram (P)
`proc2` Type of processing on `ni2` interferogram (P)
`strtext` Starting point for LP data extension for `np` dimension (P)
`strtlp` Starting point for LP calculation for `np` dimension (P)

parmax Parameter maximum values (P)

Description: An array that holds the maximum values of other parameters. The maximum value of a parameter is an index into the array, and more than one parameter can have the same index into `parmax`. Several global parameters set in the CONFIG window (opened from `config`) are part of `parmax`. To display all `parmax` values, enter `display('parmax', 'systemglobal')`.

See also: *VNMR and Solaris Software Installation; VNMR User Programming*

Related: `config` Display current configuration and possibly change it (M)
`display` Display parameters and their attributes (C)
`paramedit` Edit a parameter and its attributes with user-selected editor (C)
`paramvi` Edit a parameter and its attributes using `vi` text editor (M)

`parmin` Parameter minimum values (P)
`parstep` Parameter step size values (P)

parmin Parameter minimum values (P)

Description: An array that holds the minimum values for other parameters. The minimum value of a parameter is the index into the `parmin` array. More than one parameter may have the same index into the array. To display all the values in `parmin`, enter `display('parmin','systemglobal')`.

See also: *VNMR User Programming*

Related: `paramvi` Edit a parameter and its attributes using vi text editor (M)
`display` Display parameters and their attributes (C)
`paramedit` Edit a parameter and its attributes with user-selected editor (C)
`parmax` Parameter maximum values (P)
`parstep` Parameter step size values (P)

paros Create additional parameters used by oversampling (M)

Syntax: `paros`

Description: Creates the parameters `def_osfilt`, `filtfile`, `oscoef`, `osfb`, `osfilt`, `oslsfrq`, and `oversamp` for oversampling and digital filtering. `paros` is functionally the same as `addpar('oversamp')`.

See also: *Getting Started*

Related: `addpar` Add selected parameters to current experiment (M)
`def_osfilt` Default value of `osfilt` parameter (P)
`filtfile` File of FIR digital filter coefficients (P)
`oscoef` Digital filter coefficients for oversampling (P)
`osfb` Digital filter bandwidth for oversampling (P)
`osfilt` Oversampling filter for real-time DSP (P)
`oslsfrq` Bandpass filter offset for oversampling (P)
`oversamp` Oversampling factor for acquisition (P)

parstep Parameter step size values (P)

Description: An array that holds the step size values for other parameters. The step size value of a parameter is the index into the array. More than one parameter can have the same index into `parstep`. Several configuration parameters set in the CONFIG window (from `config`) are part of `parstep`. To display all `parstep` values, enter `display('parstep','systemglobal')`.

See also: *VNMR and Solaris Software Installation; VNMR User Programming*

Related: `config` Display current configuration and possibly change it (M)
`display` Display parameters and their attributes (C)
`paramedit` Edit a parameter and its attributes with user-selected editor (C)
`paramvi` Edit a parameter and its attributes using vi text editor (M)
`parmax` Parameter maximum values (P)
`parmin` Parameter minimum values (P)

parstyle Parameter style for plotting (P)

Description: Stores a string command to plot parameters. `parstyle` is a string parameter set by the *GLIDE* interactive interface, stored in `global`, and then executed within macros. For example, setting `parstyle='box'` results in boxed parameters below the spectrum, setting `parstyle='pap'` results in

parameters listed to the left of the spectrum, and setting `parstyle= ''` results in no parameters being plotted.

See also: *Getting Started*

Related: `pkpick` Peak pick (P)

parversion Version of parameter set (P)

Description: Stores the version of a parameter set. When a parameter set is updated with `updatepars` or `parfix`, `parversion` is set to 4.3 to indicate that fact. When a parameter set is retrieved into an experiment, `fixpar` checks `parversion` to determine if other parameters need to be updated using `parfix`.

See also: *Getting Started*

Related: `fixpar` Correct parameter characteristics in experiment (M)
`parfix` Update parameter sets (M)
`updatepars` Update all parameter sets saved in a directory (M)

path3d Path to currently displayed 2D planes from a 3D data set (P)

Applicability: All systems; however, although available on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*, such systems can only process 3D data and cannot acquire such data.

Description: Stores the absolute path to the current 3D data directory tree. If `path3d` does not exist, it is created by the macro `par3d`. The command `select`, as well as the many macros that make use of `select`, require `path3d` in order to know where the 2D planes extracted from a 3D data set can be found.

`path3d` is set automatically by the macros `ft3d` and `getplane`:

- `ft3d` sets `path3d` to `curexp/datadir3d` if `ft3d` is not supplied with a directory path for the transformed 3D data. If `ft3d` is supplied with such a directory path (e.g., `/home/data/test3d`), `path3d` is set equal to that directory path. In this case, the 3D spectral data would reside in the directory `/home/data/test3d/data`.
- `getplane` sets `path3d` to `curexp/datadir3d` if `getplane` is not supplied with a directory path to the transformed 3D data. If `getplane` is supplied with such a directory path (e.g., `/home/data/test3d`), `path3d` is set equal to that directory path. In this case, the extracted 3D planes would reside in the directory `/home/data/test3d/extr`.

See also: *User Guide: Liquids NMR*

Related: `dplane` Display a 3D plane (M)
`dproj` Display a 3D plane projection (M)
`dsplanes` Display a series of 3D planes (M)
`ft3d` Perform a 3D Fourier transform on a 3D FID data set (M)
`getplane` Extract planes from a 3D spectral set (M)
`nextpl` Display the next 3D plane (M)
`par3d` Create 3D acquisition, processing, display parameters (C)
`plane` Currently displayed 3D plane type (P)
`plplanes` Plot a series of 3D planes (M)
`prevpl` Display the previous 3D plane (M)
`select` Select a spectrum or 2D plane without displaying it (C)

patlist Active pulse template parameter list (P)

Applicability: Systems with imaging capabilities.

Description: Contains an array of strings, whose values define the rf pattern parameters used in conjunction with the length parameters defined in `plist`, for example, `patlist= 'p1pat ', 'p2pat ', 'p3pat '`. The `nD`, `seqcon`, `plist`, `patlist`, `pwrlist`, `fliplist` and `sslist` parameters configure a particular parameter set for an application sequence defined by the value of the `seqfil` parameter. The `plist`, `patlist`, `pwrlist`, `fliplist` and `sslist` parameters provide information concerning the rf pulse and conjugate gradients used by the sequence.

See also: *User Guide: Imaging*

Related:	<code>fliplist</code>	Standard flip angle list (P)
	<code>nD</code>	Application dimension (P)
	<code>plist</code>	Active pulse length parameter list (P)
	<code>pwrlist</code>	Active pulse power level parameter list (P)
	<code>seqcon</code>	Acquisition loop control (P)
	<code>seqfil</code>	Application object code name (P)
	<code>sslist</code>	Conjugate gradient list (P)

paxis Plot horizontal LC axis (M)

Applicability: Systems with the LC-NMR accessory.

Syntax: `paxis(time, major_tic, minor_tic)`

Description: Plots a horizontal LC axis. Horizontal axes are assumed to be used with “LC plots” of an entire LC run are labeled accordingly. It is assumed that relevant parameters (e.g., `sc`, `wc`, `vo`, `vp`) have not been changed after plotting the data.

Arguments: `time` is the time scale, in minutes (decimal values are fine), of the axis.
`major_tic` is spacing, in minutes (decimal values are fine), of major tics.
`minor_tic` is spacing, in minutes (decimal values are fine), of minor tics.

See also: *User Guide: Liquids NMR*

Pbox Pulse shaping software (U)

Syntax: `Pbox file options`

Description: Main Pbox (Pandora’s Box) program for the generation of shape files for RF and gradients. (See *User Guide: Liquids* manual for description of interactive Pbox usage).

Arguments: `file` is the name of a shape file.

`options` is any of the Pbox parameters initialized by the ‘-’ sign and followed by the parameter value. The following options can be in any order and combinations:

<code>-b time</code>	Activates Bloch simulator, sets <code>simtime</code> , in sec.
<code>-c</code>	Calibrate only, do not create a shape file.
<code>-f file</code>	Set name of the output file.
<code>-h wave</code>	Print wave file header.
<code>-i wave</code>	Print wave file parameters.
<code>-l ref_pw90</code>	Length, in μ s, of reference pw90 pulse.
<code>-o</code>	List options.

-p ref_pwr	Reference power level, in dB.
-r file	Reshape Pbox pulse.
-s stepsize	Define length, in μ s, of a single step in waveform.
-t wave	Print wave title.
-w wavestr	Set wave data string.
-v	Run in verbose mode. Also print Pbox version.
-value	Sets reps to value.

Examples: `Pbox -i eburp2`
`Pbox newshape -wc 'eburp1 450 -1280.0' -1`
`Pbox sel.RF -w 'eburp1 420 -800' 'eburp1 420 1200'`
`Pbox -w 'eburp1 200 -1200' -attn e -pl 45 54.2 -b`
`Pbox tst -w 'esnob 20p 170p' -sfrq 150.02 -refofs 55p`
`-ref_pwr 45 -ref_pw90 54.2`

See also: *User Guide: Liquids NMR*

Related:	<code>cpx</code>	Create Pbox shape file (M)
	<code>dprofile</code>	Display pulse excitation profile from Pbox software (M)
	<code>dshape</code>	Display pulse shape (M)
	<code>dshapef</code>	Display last generated pulse shape (M)
	<code>dshapei</code>	Display pulse shape interactively (M)
	<code>opx</code>	Open shape definition file for Pbox (M)
	<code>pbox_bw</code>	Define excitation band (M)
	<code>pbox_bws</code>	Define excitation band for solvent suppression (notch) pulses (M)
	<code>pbox_dmf</code>	Extract dmf value from Pbox shape file (M)
	<code>pbox_dres</code>	Extract dres value from Pbox shape file (M)
	<code>pbox_name</code>	Extract name of last shape file generated by Pbox (M)
	<code>pbox_pw</code>	Extract pulse length from Pbox shape file (M)
	<code>pbox_pwr</code>	Extract pulse power from Pbox shape file (M)
	<code>pbox_pwrf</code>	Extract pulse fine power from Pbox (M)
	<code>pboxget</code>	Extract all calibration data from a Pbox shape file (M)
	<code>pboxpar</code>	Add parameter definition to the pbox.inp file (M)
	<code>pboxrst</code>	Reset temporary Pbox/VNMR variables (M)
	<code>pboxunits</code>	Converts to Pbox default units (M)
	<code>pph</code>	Print pulse header (M)
	<code>pprofile</code>	Plot pulse excitation profile from Pbox software (M)
	<code>pshape</code>	Plot pulse shape (M)
	<code>pshapef</code>	Display pulse shape or modulation pattern interactively (M)
	<code>putwave</code>	Write a wave into Pbox.inp file (M)
	<code>pxset</code>	Assign Pbox calibration data to experimental parameters (M)
	<code>pxshape</code>	Generates a single-band shape file (M)
	<code>Pxsim</code>	Simulate Bloch profile for a shaped pulse (M)
	<code>Pxspy</code>	Create shape definition using Fourier coefficients (U)
	<code>selex</code>	Defines excitation band (M)
	<code>setwave</code>	Sets a single excitation band in Pbox.inp file (M)
	<code>shdec</code>	Shaped observe excitation sequence (M)

`pbox_bw` **Define excitation band (M)**

Syntax: `pbox_bw< (shapename)>`

Description: Defines the excitation band from the position of cursors in the graphics window and reports them to user. It also sets `r1` to excitation bandwidth and `r2` to offset. This macro is used mainly in Pbox menus and macros.

Arguments: `shapename` is the name of a shape as in `wavelib`; mainly for use with menus.

See also: *User Guide: Liquids NMR*

Related: **Pbox** Pulse shaping software (U)

pbox_bws Define excitation band for solvent suppression (notch) pulses (M)

Syntax: `pbox_bws<(shapename)>`

Description: Defines the excitation band from the position of cursors in the graphics window and reports them to user. It also sets `r1` to excitation bandwidth and `r2` to offset. Note, the left cursor should be placed on the left side of the excitation band and the right cursor on resonance of the solvent signal. This macro is mainly used in **Pbox** menus and macros.

Arguments: `shapename` is the name of a shape file as in `wavelib`, mainly for use with menus.

See also: *User Guide: Liquids NMR*

Related: **Pbox** Pulse shaping software (U)

pbox_dmf Extract dmf value from pbox.cal or Pbox shape file (M)

Syntax: `pbox_dmf<(shapefile.DEC)>:exp_param`

Description: Extracts the **dmf** value from the file `shapefile.DEC` created by **Pbox** or, if file name is not provided, from the `pbox.cal` file containing parameters of the last created **Pbox** shape file.

Arguments: `shapefile.DEC` is the name of a shape file.

`exp_param` is a **dmf** type experiment parameter.

Examples: `pbox_dmf('myfile.DEC'):mydmf`
`pbox_dmf:dmf2`

See also: *User Guide: Liquids NMR*

Related: **dmf** Decoupler modulation frequency for first decoupler (P)
Pbox Pulse shaping software (U)

pbox_dres Extract dres value from pbox.cal or Pbox shape file (M)

Syntax: `pbox_dres<(shapefile.DEC)>:exp_param`

Description: Extracts the **dres** value from the file `shapefile.DEC` created by **Pbox** or, if file name is not provided, from the `pbox.cal` file containing parameters of the last created **Pbox** shape file.

Arguments: `shapefile.DEC` is the name of a shape file.

`exp_param` is a **dres** type experiment parameter.

Examples: `pbox_dres('myfile.DEC'):mydres`
`pbox_dres:dres2`

See also: *User Guide: Liquids NMR*

Related: **dres** Tip-angle resolution for first decoupler (P)
Pbox Pulse shaping software (U)

pbox_name Extract name of last shape generated by Pbox from pbox.cal (M)

Syntax: `pbox_name:exp_name`

Description: Extracts name of the last shape file generated by **Pbox** and stored in the `Pbox.cal` file. Note, that the file name extension is not stored explicitly and is not provided by this macro.

Arguments: `exp_name` returns the name of last shape file.

Examples: `pbox_pw:shname`
`pbox_pw:pwpat`

See also: *User Guide: Liquids NMR*

Related: **Pbox** Pulse shaping software (U)

pbox_pw Extract pulse length from pbox.cal or Pbox shape file (M)

Syntax: `pbox_pw<(shapefile.RF)>:exp_param`

Description: Extracts pulse length from the file `shapefile.RF` generated by **Pbox** or, if file name is not provided, from `pbox.cal` file containing parameters of the last created Pbox shape file. Returns the pulse length, in μs .

Arguments: `shapefile.RF` is the shape file name, including the extension.
`exp_param` is a pw type experiment parameter.

Examples: `pbox_pw('myfile.RF'):softpw`
`pbox_pw:selpw`

See also: *User Guide: Liquids NMR*

Related: **Pbox** Pulse shaping software (U)

pbox_pwr Extract power level from Pbox.cal or Pbox shape file (M)

Syntax: `pbox_pwr<(shapefile.ext)>:exp_param`

Description: Extracts the power lever from the file `shapefile.ext` generated by **Pbox** or, if file name is not provided, from the `pbox.cal` file containing parameters of the last created Pbox shape file. Returns the power level, in dB. The `exp_param` parameter will not be changed by this macro if the parameter is previously set to 'n' (not used).

Arguments: `shapefile.ext` is the name of the shape file.
`exp_param` is a power type experiment parameter.

Examples: `pbox_pwr('myfile.DEC'):mypwr`
`pbox_pwr:dpwr2`

See also: *User Guide: Liquids NMR*

Related: **Pbox** Pulse shaping software (U)

pbox_pwrf Extract fine power level from pbox.cal or Pbox shape file (M)

Syntax: `pbox_pwrf<(shapefile.ext)>:exp_param`

Description: Extracts the fine power lever from the file `shapefile.ext` generated by **Pbox** or, if file name is not provided, from the `pbox.cal` file containing parameters of the last created Pbox shape file. Returns the value of fine power, in dB. Note that the parameter will not be changed by this macro if it was previously set to 'n' (not used).

Arguments: `shapefile.ext` is the name of the shape file.
`exp_param` is a fine power type experiment parameter.

Examples: `pbox_pwrf('myfile.DEC'):mypwrf`
`pbox_pwrf:dpwrf`

See also: *User Guide: Liquids NMR*

Related: [Pbox](#) Pulse shaping software (U)

pboxget Extract Pbox calibration data (M)

Syntax: `pboxget<(shfile.ext)>:$name,$pw,$pwr,$pwrF,$dres,$dmf`

Description: Extracts calibration data from the file `shfile.ext` generated by [Pbox](#) or, if a file name is not provided, from the `pbox.cal` file containing parameters of the last created Pbox shape file. Returns shape name and the values of total pulse length (in μs), power (dB), fine power, `dres`, and `dmf`. The parameter will not be changed by this macro if the parameter was previously set to 'n' (not used).

Arguments: `shfile.ext` is the name of the shape file, including the extension.

`name` is the experiment parameter receiving the shape name (without the extension).

`pw` is the experiment parameter receiving the total pulse length, in μs .

`pwr` is the experiment parameter receiving the power level, in dB.

`pwrF` is the experiment parameter receiving the fine power level.

`dres` is the experiment parameter receiving the decoupler resolution.

`dmf` is the experiment parameter receiving the decoupler modulation frequency.

Examples: `pboxget('myfile.DEC'):dseq,r1,dpwr,dpwrF,dres,dmf`

`pboxget('selshape.RF'):pwpat,selpw,selpwr`

`pboxget:dseq2,r1,dpwr2,dpwrF2,dres2,dmf2`

See also: *User Guide: Liquids NMR*

Related: [Pbox](#) Pulse shaping software (U)

pboxpar Add parameter definition to the Pbox.inp file (M)

Syntax: `pboxpar(param,value)`

Description: Adds a parameter definition to the `Pbox.inp` file.

Arguments: `param` is the parameter name

`value` is the value of the parameter.

Examples: `pboxpar('name','myfile.DEC')`

`pboxpar('bsim','y')`

`pboxpar('T1',0.24)`

See also: *User Guide: Liquids NMR*

Related: [Pbox](#) Pulse shaping software (U)

pboxrst Reset temporary Pbox VNMR variables (M)

Syntax: `pbox_rst`

Description: Resets `r1=0`, `r2=0`, `r3=0`, `r4=0`, `n2='n'`, `n3=''`, and adds some standard comment lines to the `Pbox.inp` file. This macro is used in menus and other Pbox macros.

See also: *User Guide: Liquids NMR*

Related: [Pbox](#) Pulse shaping software (U)

pboxunits **Converts to Pbox default units (M)**

Syntax: `pboxunits`

Description: Used by Pbox menus to scale parameters related to time or frequency down to Pbox default units (Hz or seconds) before the parameter is stored in the `Pbox.inp` file.

See also: *User Guide: Liquids NMR*

Related: [Pbox](#) Pulse shaping software (U)

pcmapapply **Apply phase correction map to data in EPI experiments (C)**

Applicability: Systems with echo planar imaging (EPI) capabilities.

Syntax: `pcmapapply(<file,>index)`

Description: Applies a pixel-by-pixel phase shift to the current data file using the complex phase correction values from the phase correction map file, which must exist in `$vnmruser/expN/datdir`, where N is the current experiment number. `pcmapapply` opens and closes a phase map file unless it has been explicitly opened with `pcmapopen`.

Arguments: `file` specifies a phase correction map file name that must reside in the directory `$vnmruser/expN/datdir`. The default file is `$vnmruser/expN/datdir/pcmap`.

`index` specifies which phase correction map to use in the file. The value is usually 1, but can range up to the number of map blocks in the file.

Examples: `pcmapapply(2)`
`pcmapapply('mypcmap',1)`

See also: *User Guide: Imaging*

Related: [pcmapclose](#) Apply phase correction map to data in EPI experiments (C)
[pcmapgen](#) Generate phase correction map in EPI experiments (C)
[pcmapopen](#) Open phase correction map file in EPI experiments (C)

pcmapclose **Close phase correction map in EPI experiments (C)**

Applicability: Systems with echo planar imaging (EPI) capabilities.

Syntax: `pcmapclose`

Description: Closes a phase correction map file that was explicitly opened with the `pcmapopen` command.

See also: *User Guide: Imaging*

Related: [pcmapapply](#) Apply phase correction map to data in EPI experiments (C)
[pcmapgen](#) Generate phase correction map in EPI experiments (C)
[pcmapopen](#) Open phase correction map file in EPI experiments (C)

pcmapgen **Generate phase correction map in EPI experiments (C)**

Applicability: Systems with echo planar imaging (EPI) capabilities.

Syntax: `pcmapgen(<file,>index)`

Description: Generates pixel-by-pixel complex phase correction values from the current data file and stores them into the selected block in the phase correction map file. One or more phase correction maps can be generated. For multislice echo planar imaging experiments, there can be one phase correction map for each slice.

`pcmapgen` creates, opens, and closes a phase map file unless the file has been explicitly opened with the `pcmapopen` command.

Arguments: `file` specifies a phase correction map file name, which must reside in the directory `$vnmruser/expN/datdir`, where `N` is the current experiment number. The default file is `$vnmruser/expN/datdir/pcmap`.

`index` specifies which phase correction map to use in the file. The value is usually 1, but can range up to the number of map blocks in the file.

Examples: `pcmapgen(2)`
`pcmapgen(myPCmap,1)`

See also: *User Guide: Imaging*

Related: `pcmapapply` Apply phase correction map to data in EPI experiments (C)
`pcmapclose` Close phase correction map file in EPI experiments (C)
`pcmapopen` Open phase correction map file in EPI experiments (C)

pcmapopen Open phase correction map in EPI experiments (C)

Applicability: Systems with echo planar imaging (EPI) capabilities.

Syntax: `pcmapopen(<file>, >max_index)`

Description: Explicitly opens a phase correction map file, which can significantly speed up data processing. After the map file is open, use `pcmapgen` and `pcmapapply` to generate maps and correct data. Use `pcmapclose` to close the file when you are finished with it.

Arguments: `file` specifies the phase correction map file name residing in the directory `$vnmruser/expN/datdir`, where `N` is the current experiment number. The default is the file `pcmap`.

`max_index` specifies the maximum number of phase correction maps in the file, which ensures that memory mapping extends to or past the end of the file. `max_index` must be greater than or equal to the maximum number of phase maps stored in the file.

Examples: `pcmapopen(2)`
`pcmapopen('myPCmap',1)`

See also: *User Guide: Imaging*

Related: `pcmapapply` Apply phase correction map to data in EPI experiments (C)
`pcmapclose` Close phase correction map file in EPI experiments (C)
`pcmapgen` Generate phase correction map in EPI experiments (C)

pcon Plot contours on a plotter (C)

Syntax: `pcon(<('pos'|'neg')><,'noaxis'><,<levels><,<spacing>>>`

Description: Plots positive and negative peaks of a contour plot display using different colors. Specifically, if `maxpen` is set for `n` pens, positive peaks are plotted using colors 1 through $(n+1)/2$, and negative peaks are plotted using colors $((n+1)/2)+1$ through `n` (i.e., half the colors for each, plus one extra for positive if an odd number of pens is specified). Pen 1 is always used for the axes, and the lowest contour of the positive peaks is also plotted with pen 1. In all cases, the pen colors are cycled if more contours are to be plotted than there are pens available.

To plot both negative and positive contours of a phase-sensitive spectrum on a monochrome device such as a LaserJet or a plotter with a single pen, different numbers of contours may be plotted for the different sign. For example, `pcon('pos',10,1.4)` `pcon('neg',1)` will plot ten closely spaced positive contours and one negative contour.

Arguments: 'pos' is a keyword specifying that phase-sensitive spectra plot positive peaks only. The default is to plot both positive and negative peaks.

'neg' is a keyword specifying that phase-sensitive spectra plot negative peaks only. The default is to plot both positive and negative peaks.

'noaxis' is a keyword to omit outlining the plot and omit plotting the horizontal and vertical axes.

levels is maximum number of contour levels to plot. The default is 4.

spacing is relative intensity of successive contour levels. The default is 2.

Examples: `pcon`
`pcon(4,1.4)`
`pcon('pos','noaxis')`
`pcon('neg',3)`

See also: *User Guide: Liquids NMR*

Related: `dpcon` Display plotted contours (C)
`maxpen` Maximum number of pens to use (P)

pcss Calculate and show proton chemical shifts spectrum (M)

Syntax: `pcss(<threshold><,max_cc><,max_width>)`

Description: Calculates and shows the proton chemical shifts spectrum. The `dsp` command is used to display the results. The list of chemical shifts is saved in the file `pcss.outpar`. The original spectrum can be calculated by the `wft` command.

Arguments: `threshold` sets the level whether a point belongs to a peak or is noise. The default is that `pcss` automatically calculates the threshold.

`max_cc` is the maximum allowable coupling constant in the spectrum. The default is 20 Hz.

`max_width` is the maximum width of a spin multiplet in the spectrum. The default is 60 Hz.

Examples: `pcss`
`pcss(10)`
`pcss(9,20,80)`

See also: *User Guide: Liquids*

Related: `do_pcss` Calculate proton chemical shifts spectrum (C)
`dsp` Display pulse sequence (C)
`wft` Weight and Fourier transform 1D data (C)

peak Find tallest peak in specified region (C)

Syntax: `peak<(min_freq,max_freq)><:height,freq>`

Description: Returns the height and frequency of the tallest peak in the selected region, including any referencing (i.e., the same frequency that you would measure by placing a cursor on the peak). A spectrum need not actually be displayed for `peak` to work.

Arguments: With no return arguments, `peak` displays on the screen information about peak height and frequency. If two cursors are displayed, `peak` without arguments finds the tallest peak between the cursors.

`min_freq` is minimum frequency limit of the region to be searched. The default value is `sp`.

`max_freq` is maximum frequency limit, in Hz, of the region to be searched.
The default value is `sp + wp`.

`height` returns the height, in mm, of the tallest peak in the selected region.

`freq` returns the frequency, in Hz, of the tallest peak in the selected region.

Examples: `peak:$ht,$freq`
`peak(0,2000):r3`
`peak:$ht,cr`

See also: *VNMR User Programming*

Related: `sp` Start of plot (P)
`wp` Width of plot (P)

peak2d Return information about maximum in 2D data (C)

Syntax: `peak2d:$maximum_intensity<,$trace,$point>`

Description: Searches the area defined by `sp`, `wp`, `sp1`, and `wp1` in a 2D data set for a maximum intensity.

Arguments: `$maximum_intensity` returns the maximum intensity value found.

`$trace` returns the trace number of the maximum. The parameter `trace` defines whether `f1` or `f2` traces are counted.

`$point` returns the data point number of the maximum on that trace.

See also: *User Guide: Liquids NMR*

Related: `sp` Start of plot (P)
`sp1` Start of plot in 1st indirectly detected dimension (P)
`trace` Mode for *n*-dimensional data display (P)
`wp` Width of plot (P)
`wp1` Width of plot in 1st indirectly detected dimension (P)

peccfile Programmable eddy current compensation file (obsolete)

Description: An obsolete SIS parameter that used to specify the file that contained the current programmable eddy current compensation values for a gradient coil. The parameter may still exist in old SIS parameter sets and gradient tables but it is not used.

Related: `gcoil` Read data from gradient calibration tables (P)

pen Select a pen or color for drawing (C)

Syntax: `pen(<'graphics' | 'plotter', ><'xor' | 'normal', >
pen|color)`

Description: Selects the pen number for a plotter or the color for the graphics screen. This command is part of a line drawing capability that includes the `move` and `draw` commands. `move` sets the coordinates from which the line starts. `draw` draws a line from that point to the new coordinates specified by `draw`. Refer to the description of `draw` for examples of using the line drawing capability.

Arguments: `'graphics'` and `'plotter'` are keywords selecting the output device. The default is `'plotter'`. The output selected is passed to subsequent `pen`, `move`, or `draw` commands and remains active until a different output is specified.

`'xor'` and `'normal'` are keywords selecting the drawing mode for the `'graphics'` output device. In the `'xor'` mode, if a line is drawn such that

one or more points of the line are in common with a previously drawn line, the common points are erased. In the 'normal' mode, the common points remain. The mode selected is passed to subsequent pen, **draw**, or **move** commands and remains active until a different mode is specified. The default mode is 'normal'.

pen is the plotter pen number: 'pen1', 'pen2', 'pen3', etc.

color is the active color for the graphics screen: 'red', 'green', 'blue', 'cyan', 'magenta', 'yellow', 'black', or 'white'.

Examples: `pen('pen2')`
`pen('graphics','red')`

See also: *Getting Started*

Related: **draw** Draw line from current location to another location (C)
move Move to an absolute location (C)

pexpl Plot exponential or polynomial curves (C)

Syntax: `pexpl(<options>,<line1,line2, ...>`

Description: Plots exponential curves resulting from T_1 , T_2 , or kinetics analysis. Also plots polynomial curves from diffusion or other types of analysis. The `analyze.out` file is the data input file used to make the plot. Refer to the **expl** entry for the format of this file. The parameters **sc**, **wc**, **sc2**, and **wc2** control the size of the plot.

Arguments: `options` are any of the following keywords:

- 'linear', 'square', and 'log' provide for plotting of the data points against the square or log of the data. 'linear' controls x-axis scale, 'square' controls the y-axis. The default is 'linear'.
- 'link' causes the data points to be connected rather than a plot of the theoretical curve.
- 'nocurve' produces a plot of data points only.
- 'oldbox' plots an additional curve on an existing plot. Only the first data set in `analyze.out` is plotted. It causes the program to get box and scale description from `expfit.out` in the current experiment.
- 'file' followed by a file name replaces `analyze.out` as the input.

`line1`, `line2`, ... specify curves to be plotted. The default is to plot the first six curves (if that many exist) along with the data points.

Examples: `pexpl`
`pexpl(1,3,6)`

See also: *User Guide: Liquids NMR, VNMR User Programming*

Related: **expl** Display exponential or polynomial curves (C)
sc Start of chart (P)
sc2 Start of chart in second direction (P)
wc Width of chart (P)
wc2 Width of chart in second direction (P)

pexpladd Add another diffusion analysis to current plot (M)

Applicability: Systems with the diffusion option.

Syntax: `pexpladd(integral_region)`

Description: Adds results of another diffusion analysis to the currently plotted results.

Arguments: `integral_region` specifies the number of the region whose results are to be added to the existing plot.

Examples: `pexpladd(1)`

See also: *User Guide: Liquids NMR*

Related: `expl` Display exponential or polynomial curves (C)
`pexpl` Plot exponential or polynomial curves (C)
`expladd` Add another diffusion analysis to current display (M)

pfgon Pulsed field gradient amplifiers on/off control (P)

Applicability: Systems with pulsed field gradient (PFG) modules.

Description: A global string parameter controlling the X, Y, and Z gradients for the PFG current amplifiers. Entering `su` or `go` sets the amplifiers at the current value of `pfgon`. For `pfgon` to take effect, `gradtype` must equal `p`, `q`, `l`, `t`, or `u` for the corresponding X, Y, or Z gradient, and a `su` or a `go` must be issued.

Values: A three-character string, with the first character controlling the X gradient, the second the Y gradient, and the third the Z gradient. For each gradient, setting the value to `y` turns on an amplifier and setting the value to `n` turns it off. For example, `pfgon='nny'` turns on only the PFG amplifier on the Z channel, and `pfgon='nnn'` turns off the PFG amplifiers on all channels.

See also: *Getting Started; Performa I Pulsed Field Gradient Module Installation; Pulsed Field Gradient Modules Installation; VNMR User Guide: Liquids NMR*

Related: `go` Submit experiment to acquisition (M)
`gradtype` Gradients for X, Y, and Z axes (P)
`setup` Set up parameters for basic experiments (M)
`su` Submit a setup experiment to acquisition (M)

pfiltr Programmable filters (P)

Applicability: *GEMINI 2000* systems.

Description: Specifies presence or absence of programmable filters on observe receiver. If this parameter is not present, it can be created by `vnmr1` with the command `create('pfiltr','string','systemglobal')`.

Values: `'y'` if the filters are present, `'n'` if the filters are not present.

See also: *VNMR and Solaris Software Installation.*

Related: `attens` Fast attenuators present (P)
`create` Create new parameter in parameter tree (C)
`rcvr` Receiver version in system (P)

pfwf Plot FIDs in whitewash mode (C)

Syntax: `pfwf(<start><,finish><,step><,'all'|'imag'>>`

Description: Plots FIDs in whitewash mode (after the first FID, each FID is blanked out in regions in which it is behind an earlier FID). The position of the first FID is governed by parameters `wc`, `sc`, and `vpf`.

Arguments: `start` is the index of a particular FID for arrayed 1D or 2D data sets. For multiple FIDs, `start` is the index of the first FID.

`finish` is the index of the last FID for multiple FIDs.

`step` specifies the increment for the FID index. The default is 1.

`'all'` is a keyword to plot all of the FIDs. This is the default.

'imag' is a keyword to plot only the imaginary FID channel. The default is 'all'.

Examples: `pfww`
`pfww(4,10,2,'imag')`

See also: *Getting Started*

Related: `dfs` Display stacked FIDs (C)
`dfww` Display FIDs in whitewash mode (C)
`plfid` Plot FIDs (C)
`sc` Start of chart (P)
`vpf` Current vertical position of FID (P)
`wc` Width of chart (P)

pge Convert parameter set to PGE pulse sequence (M)

Applicability: Systems with the diffusion option.

Syntax: `pge`

Description: Adds all necessary parameters to perform the PGE (Pulse Gradient Experiment) pulse sequence, taking those parameters from the file `/vnmr/parlib/pge`.

See also: *User Guide: Liquids NMR*

Related: `pge_calib` Calibrate gradient strengths for PGE pulse sequence (M)
`pge_data` Extract data from single element of PGE pulse sequence (M)
`pge_output` Output results from PGE pulse sequence (M)
`pge_process` Automated processing of data from PGE pulse sequence (M)
`pge_results` Calculate diffusion constant for integral region (M)
`pge_setup` Set up gradient control parameters for PGE pulse sequence (M)

pge_calib Calibrate gradient strengths for PGE pulse sequence (M)

Applicability: Systems with the diffusion option.

Syntax: `pge_calib`

Description: Calibrates the parameters `grad_cw_coef` and `grad_p_coef`, which relate the DAC values (in DAC units) to the gradient strengths (in gauss/cm). Given a diffusion constant measurement (made with `pge_results`) for a known diffusion constant, `pge_calib` then adjusts the calibration parameters to produce the correct diffusion constant.

See also: *User Guide: Liquids NMR*

Related: `pge` Calibrate gradient strengths for PGE pulse sequence (M)
`pge_results` Calculate diffusion constant for integral region (M)

pge_data Extract data from single element of PGE pulse sequence (M)

Applicability: Systems with the diffusion option.

Syntax: `pge_data(array_index)`

Description: Extracts integral information from a currently displayed element of a PGE (Pulse Gradient Experiment) and writes the results in the current experiment directory as the file `info_#`, where # is the value of the `array_index` argument (e.g., if `array_index` is 5, the file is `info_5`)

Arguments: `array_index` is the number of the array element from which the data is extracted.

Examples: `pge_data(5)`

See also: *User Guide: Liquids NMR*

Related: [pge](#) Calibrate gradient strengths for PGE pulse sequence (M)

pge_output Output results from PGE pulse sequence (M)

Applicability: Systems with the diffusion option.

Syntax: `pge_output`

Description: Prints the calculated results from the PGE (Pulse Gradient Experiment) pulse sequence on a printer and plots the graphs of calculated decay curves.

See also: *User Guide: Liquids NMR*

Related: [pge](#) Calibrate gradient strengths for PGE pulse sequence (M)

pge_process Automated processing of data from PGE pulse sequence (M)

Applicability: Systems with the diffusion option.

Syntax: `pge_process`

Description: Performs full automated processing of data from a PGE (Pulse Gradient Experiment) pulse sequence.

See also: *User Guide: Liquids NMR*

Related: [pge](#) Calibrate gradient strengths for PGE pulse sequence (M)

pge_results Calculate diffusion constant for integral region (M)

Applicability: Systems with the diffusion option.

Syntax: `pge_results(integral_region<,reference_region>)`

Description: Calculates a diffusion coefficient based on a single integral region in the spectrum (if one input argument) or calculates diffusion coefficient of an integral region consisting of two components (if two input arguments).

Arguments: `integral_region` is the number of the integral region on which to perform the analysis

`reference_region` is the number of the integral region used to get the value of the diffusion coefficient.

Examples: `pge_results(2)`
`pge_results(1,3)`

See also: *User Guide: Liquids NMR*

Related: [pge](#) Calibrate gradient strengths for PGE pulse sequence (M)

pge_setup Set up gradient control parameters for PGE pulse sequence (M)

Applicability: Systems with the diffusion option.

Syntax: `pge_setup('no')`

Description: Prompts the user for the values of the `g_max`, `g_min`, `g_steps`, `g_array`, `nt_first`, `nt_array`, and other parameters for the PGE (Pulse Gradient Experiment) pulse sequence. These parameters are then used to calculate the `grad_p1` and `nt` arrays.

Arguments: `'no'` is a keyword to turn off prompting the user and instead use the current values of the parameters to calculate the `grad_p1` and `nt` arrays.

Examples: `pge_setup`
`pge_setup('no')`

See also: *User Guide: Liquids NMR*

Related: `pge` Calibrate gradient strengths for PGE pulse sequence (M)

ph Set phased mode in directly detected dimension (C)

Syntax: `ph`

Description: Selects the phased mode by setting the parameter `dmg` = 'ph'. In the *phased spectra display mode*, each real point in the displayed spectrum is calculated from a linear combination of the real and imaginary points comprising each respective complex data point. The coefficients for this linear combination are derived from the phase parameters `rp` and `lp`.

For 2D data, if `pmode` = 'partial' or `pmode` = '' (two single quotes with no space in between), `ph` has an effect on the data prior to the second Fourier transform. If `pmode` = 'full', `ph` acts in concert with the commands `ph1`, `av1`, or `pwr1` to yield the resultant contour display for the 2D data.

See also: *Getting Started; User Guide: Liquids NMR*

Related:

<code>av</code>	Set abs. value mode in directly detected dimension (C)
<code>av1</code>	Set abs. value mode in 1st indirectly detected dimension (C)
<code>dmg</code>	Data display mode in directly detected dimension (P)
<code>ft</code>	Fourier transform 1D data (C)
<code>ft1d</code>	Fourier transform along f_2 dimension (C)
<code>ft2d</code>	Fourier transform 2D data (C)
<code>lp</code>	First-order phase in directly detected dimension (P)
<code>pa</code>	Set phase angle mode in directly detected dimension (C)
<code>pa1</code>	Set phase angle mode in 1st indirectly detected dimension (C)
<code>ph1</code>	Set phased mode in 1st indirectly detected dimension (C)
<code>ph2</code>	Set phased mode in 2nd indirectly detected dimension (C)
<code>pmode</code>	Processing mode for 2D data (P)
<code>pwr</code>	Set power mode in directly detected dimension (C)
<code>pwr1</code>	Set power mode in 1st indirectly detected dimension (C)
<code>rp</code>	Zero-order phase in directly detected dimension (P)
<code>wft</code>	Weight and Fourier transform 1D data (C)
<code>wft1d</code>	Weight and Fourier transform f_2 of 2D data (M)
<code>wft2d</code>	Weight and Fourier transform 2D data (M)

ph1 Set phased mode in 1st indirectly detected dimension (C)

Syntax: `ph1`

Description: Selects the phased spectra display mode along the first indirectly detected dimension by setting the parameter `dmg1` to the string value 'ph1'. If the parameter `dmg1` does not exist, `ph1` will create it and set it to 'ph1'.

In the phased mode, each real point in the displayed trace is calculated from a linear combination of the real and imaginary points comprising each respective complex data point. For hypercomplex data, the linear combination uses the real-real and imaginary-real points from each respective hypercomplex data point. The coefficients for this linear combination are derived from the phase parameters `rp1` and `lp1`.

The `ph1` command is only needed if mixed-mode display is desired. If the parameter `dmg1` does not exist or is set to the null string, the display mode along the first indirectly detected dimension defaults to the display mode of the directly detected dimension (characterized by the parameter `dmg`). For the contour display of multidimensional data, the result of `ph1` is the same as for traces provided that `pmode` = 'partial' or `pmode` = ''.

See also: *User Guide: Liquids NMR*

Related:	av1	Set abs. value mode in 1st indirectly detected dimension (C)
	dmg1	Data display mode in 1st indirectly detected dimension (P)
	lp1	First-order phase in 1st indirectly detected dimension (P)
	pa	Set phase angle mode in directly detected dimension (C)
	pa1	Set phase angle mode in 1st indirectly detected dimension (C)
	ph	Set phased mode in directly detected dimension (C)
	pmode	Processing mode for 2D data (P)
	pwr1	Set power mode in 1st indirectly detected dimension (C)
	rp1	Zero-order phase in 1st indirectly detected dimension (P)

ph2 **Set phased mode in 2nd indirectly detected dimension (C)**

Syntax: `ph2`

Description: Selects phased spectrum display mode processing along the second indirectly detected dimension by setting the parameter `dmg2='ph2'`. If `dmg2` does not exist or is set to the null string, `ph2` creates `dmg2` and sets it to `'ph2'`.

In the phased mode, each real point in the displayed trace is calculated from a linear combination of the real and imaginary points comprising each respective complex data point. For hypercomplex data, the linear combination uses the real-real and imaginary-real points from each respective hypercomplex data point. The coefficients for this linear combination are derived from the phase parameters `rp2` and `lp2`.

The `ph2` command is only needed if mixed-mode display is desired. If the parameter `dmg2` does not exist or is set to the null string, the display mode along the second indirectly detected dimension defaults to the display mode of the directly detected dimension (characterized by the parameter `dmg`). For the contour display of multidimensional data, the result of `ph2` is the same as for traces provided that `pmode='partial'` or `pmode=''`.

See also: *User Guide: Liquids NMR*

Related:	av2	Set abs. value mode in 2nd indirectly detected dimension (C)
	dmg2	Data display mode in 2nd indirectly detected dimension (P)
	ft1d	Fourier transform along f_2 dimension (C)
	ft2d	Fourier transform 2D data (C)
	lp2	First-order phase in 2nd indirectly detected dimension (P)
	ph	Set phased mode in directly detected dimension (C)
	pmode	Processing mode for 2D data (P)
	pwr2	Set power mode in 2nd indirectly detected dimension (C)
	rp2	Zero-order phase in 2nd indirectly detected dimension (P)

phase **Change frequency-independent phase rp (M)**

Syntax: `phase(phase_change)`

Description: Changes the phase of all peaks in the spectrum by adding a value to the current `rp` value. Any excess over 360° is removed.

Arguments: `phase_change` is the value to be added to the current `rp` value (i.e., $new\ rp = old\ rp + phase_change$).

Examples: `phase(45)`

See also: *Getting Started*

Related:	rp	Zero-order phase in directly detected dimension (P)
----------	-----------	---

phase **Phase selection (P)**

Description: Selects the phase cycling that determines the experiment type. To create the parameters `phase`, `ni`, and `sw1` for acquisition of a 2D data set in the current experiment, enter `addpar(' 2d ')`.

Values: The following values are generally used in experiments with phase cycling. For more details, see the specific pulse sequence.

`phase=0` selects an absolute-value 2D experiment.

`phase=1, 2` selects the required two components of a hypercomplex (States-Haberkorn) experiment.

`phase=3` selects TPPI (Time Proportional Phase Incrementation).

See also: *User Guide: Liquids NMR*

Related:	<code>addpar</code>	Add selected parameters to the current experiment (M)
	<code>cosyps</code>	Set up parameters for phase-sensitive COSY (M)
	<code>dqcosy</code>	Set up parameters for double quantum filtered COSY (M)
	<code>hmqc</code>	Set up parameters for HMQC pulse sequence (M)
	<code>hmqcr</code>	Set up parameters for HMQCR pulse sequence (M)
	<code>inadqt</code>	Set up parameters for INADEQUATE pulse sequence (M)
	<code>mqcosy</code>	Set up parameters for MQCOSY pulse sequence (M)
	<code>noesy</code>	Set up parameters for NOESY pulse sequence (M)
	<code>roesy</code>	Set up parameters for ROESY pulse sequence (M)
	<code>tocsy</code>	Set up parameters for TOCSY pulse sequence (M)

phase1 **Phase of first pulse (P)**

Applicability: Systems with a solids module.

Description: Controls the first pulse phase in the cycle, in multipulse experiments.

See also: *User Guide: Solid-State NMR*

Related:	<code>br24</code>	Set up BR24 multiple pulse experiment (M)
	<code>flipflop</code>	Set up sequences for multipulse (M)

phase2 **Phase selection for 3D acquisition (P)**

Description: Selects phase cycling type for 3D data acquisitions. Also selects the phase of the second pulse in the sequence set up by `flipflop`. To create the parameters `phase2`, `d3`, `ni2`, and `sw2` for acquisition of a 3D data set in the current experiment, enter `addpar(' 3d ')`.

See also: *User Guide: Liquids NMR*; *User Guide: Solid-State NMR*

Related:	<code>addpar</code>	Add selected parameters to the current experiment (M)
	<code>d3</code>	Incremented delay for 2nd indirectly detected dimension (P)
	<code>flipflop</code>	Set up sequences for multipulse (M)
	<code>ni2</code>	Number of increments in 2nd indirectly detected dimension (P)
	<code>par3d</code>	Create 3D acquisition, processing, display parameters (C)
	<code>sw2</code>	Spectral width in 2nd indirectly detected dimension (P)

phase3 **Phase selection for 4D acquisition (P)**

Description: Selects phase cycling type for 4D data acquisitions. To create the parameters `phase3`, `d4`, `ni3`, and `sw3` for acquisition of a 4D data set in the current experiment, enter `addpar(' 4d ')`.

See also: *User Guide: Liquids NMR*

Related:	<code>addpar</code>	Add selected parameters to the current experiment (M)
	<code>d4</code>	Incremented delay for 3rd indirectly detected dimension (P)
	<code>ni3</code>	Number of increments in 3rd indirectly detected dimension (P)
	<code>par4d</code>	Create 4D acquisition parameters (C)
	<code>sw3</code>	Spectral width in 3rd indirectly detected dimension (P)

phasing **Control update region during interactive phasing (P)**

Description: Controls the percentage of the spectrum updated during interactive phasing using the `ds` command.

Values: 10 to 100, in percent, where 100 causes the entire spectrum to be updated, and 20 causes the area between the two vertical cursors to be updated.

See also: *Getting Started*

Related:	<code>ds</code>	Display a spectrum (C)
----------	-----------------	------------------------

phfid **Zero-order phasing constant for the np FID (P)**

Description: Specifies the angle of zero-order rotation. This zero-order rotation is executed as a part of retrieving the time-domain data into the active region of the VNMR memory and can be used instead of the parameter `rp` applied to the frequency-domain data. `phfid` is used only in a complex phase rotation.

`phfid` (and related parameters `lsfid` and `lsfrq`) operate on complex `np` FID data, referred to as the t_2 dimension in a 2D experiment or as the t_3 dimension in a 3D experiment. `phfid` is in the processing group and is properly handled through the `wti` display.

Values: -360.0 to +360.0, in degrees; 'n'

See also: *Getting Started; User Guide: Liquids NMR*

Related:	<code>dfid</code>	Display a single FID (C)
	<code>ds</code>	Display a spectrum FID (C)
	<code>ft</code>	Fourier transform 1D data (C)
	<code>ft1d</code>	Fourier transform along f_2 dimension (C)
	<code>ft2d</code>	Fourier transform 2D data (C)
	<code>lsfid</code>	Number of complex points to left-shift the <code>np</code> FID (P)
	<code>lsfrq</code>	Frequency shift of the <code>fn</code> spectrum in Hz (P)
	<code>np</code>	Number of data points (P)
	<code>phfid1</code>	Zero-order phasing constant for <code>ni</code> interferogram (P)
	<code>phfid2</code>	Zero-order phasing constant for <code>ni2</code> interferogram (P)
	<code>rp</code>	Zero-order phase in directly detected dimension (P)
	<code>wft</code>	Weight and Fourier transform 1D data (C)
	<code>wft1d</code>	Weight and Fourier transform f_2 of 2D data (M)
	<code>wft2d</code>	Weight and Fourier transform 2D data (M)
	<code>wti</code>	Interactive weighting (C)

phfid1 **Zero-order phasing constant for ni interferogram (P)**

Description: Specifies the angle of zero-order rotation. This zero-order rotation is executed as a part of retrieving the time-domain data into the active region of the VNMR memory and can be used instead of the parameter `rp1` applied to the frequency-domain data. `phfid1` is used in a complex phase rotation for complex t_1/t_2 interferograms and in a hypercomplex phase rotation for hypercomplex t_1/t_2 interferograms.

`phfid1` (and related parameters `lsfid1` and `lsfrq1`) operate on `ni` interferogram data, both hypercomplex and complex. `ni` interferogram data are referred to as the t_1 dimension in both a 2D and a 3D experiment. `phfid1` is in the processing group and is properly handled through the `wti` display; that is, a `wti` operation on an `ni` interferogram applies the parameters `phfid1`, `lsfid1`, and `lsfrq1`, if selected, to the time-domain data prior to the Fourier transformation.

Values: -360.0 to $+360.0$, in degrees; 'n'.

See also: *User Guide: Liquids NMR*

Related:	<code>lsfid1</code>	Number of complex points to left-shift the <code>ni</code> interferogram (P)
	<code>lsfrq1</code>	Frequency shift of the <code>fn1</code> spectrum in Hz (P)
	<code>ni</code>	Number of increments in 1st indirectly detected dimension (P)
	<code>phfid</code>	Zero-order phasing constant for <code>np</code> FID (P)
	<code>phfid2</code>	Zero-order phasing constant for <code>ni2</code> interferogram (P)
	<code>rp1</code>	Zero-order phase in 1st indirectly detected dimension (P)
	<code>wti</code>	Interactive weighting (C)

`phfid2` Zero-order phasing constant for `ni2` interferogram (P)

Description: Specifies the angle of zero-order rotation. This zero-order rotation is executed as a part of retrieving the time-domain data into the active region of the VNMR memory and can be used instead of the parameter `rp2` applied to the frequency-domain data. `phfid2` is used in a complex phase rotation for complex t_1/t_2 interferograms and in a hypercomplex phase rotation for hypercomplex t_1/t_2 interferograms.

`phfid2` (and related parameters `lsfid2` and `lsfrq2`) operate on `ni2` interferogram data, both hypercomplex and complex. `ni2` interferogram data are referred to as the t_2 dimension in a 3D experiment. `phfid2` is in the processing group and is properly handled through the `wti` display.

Values: -360.0 to $+360.0$, in degrees; 'n'.

See also: *User Guide: Liquids NMR*

Related:	<code>lsfid2</code>	Number of complex points to left-shift <code>ni2</code> interferogram (P)
	<code>lsfrq2</code>	Frequency shift of the <code>fn2</code> spectrum in Hz (P)
	<code>ni2</code>	Number of increments in 2nd indirectly detected dimension (P)
	<code>phfid</code>	Zero-order phasing constant for <code>np</code> FID (P)
	<code>phfid1</code>	Zero-order phasing constant for <code>ni</code> interferogram (P)
	<code>rp2</code>	Zero-order phase in 2nd indirectly detected dimension (P)
	<code>wti</code>	Interactive weighting (C)

`phi` Euler angle phi from magnet frame (P)

Applicability: Systems with imaging capabilities.

Description: Euler angle phi from magnet frame.

Values: -180 to $+180$, in degrees.

See also: *User Guide: Imaging*

Related:	<code>psi</code>	Euler angle psi from magnet frame (P)
	<code>theta</code>	Euler angle theta from magnet frame (P)

PHOSPHORUS Set up parameters for phosphorus spectrum (M)

Applicability: *GLIDE* only

Syntax: PHOSPHORUS

Description: Internal macro that sets up a phosphorus spectrum in *GLIDE*. This macro is not used if phosphorus is the first experiment in the chain.

pi **Inversion pulse length (P)**

Applicability: Systems with imaging capabilities.

Description: Pulse length for an inversion pulse, often used as an optional first pulse preceding the main sequence to provide contrast based on T_1 relaxation.

A `pi` pulse will often be programmed so that it may be toggled on or off by the operator with the inversion-recovery flag `ir`.

See also: *User Guide: Imaging*

Related: `ir` Inversion recovery mode (P)
`pipat` Shape of an inversion pulse (P)
`ti` Second delay in an inversion recovery sequence (P)
`tpwri` Intensity of an inversion pulse in dB (P)

pi3ssbsq **Set up pi/3 shifted sinebell-squared window function (M)**

Syntax: `pi3ssbsq(<t1_inc><,t2_inc>)>`

Description: Sets up a pi/3 unshifted sinebell-squared window function in 1, 2, or 3 dimensions. The macro checks whether the data is 1D, 2D, and 3D.

Arguments: `t1_inc` is the number of t1 increments. The default is `ni`.
`t2_inc` is the number of t2 increments. The default is `ni2`.

See also: *Getting Started; User Guide: Liquids NMR*

Related: `gaussian` Set up unshifted Gaussian window function (M)
`ni` Number of increments in 1st indirectly detected dimension (P)
`ni2` Number of increments in 2nd indirectly detected dimension (P)
`pi4ssbsq` Set up pi/4 shifted sinebell-squared window function (M)
`sqcosine` Set up unshifted cosine-squared window function (M)
`sq sinebell` Set up unshifted sinebell-squared window function (M)

pi4ssbsq **Set up pi/4 shifted sinebell-squared window function (M)**

Syntax: `pi4ssbsq(<t1_inc><,t2_inc>)>`

Description: Sets up a pi/4 unshifted sinebell-squared window function in 1, 2, or 3 dimensions. The macro checks whether the data is 1D, 2D, and 3D.

Arguments: `t1_inc` is the number of t1 increments. The default is `ni`.
`t2_inc` is the number of t2 increments. The default is `ni2`.

See also: *Getting Started; User Guide: Liquids NMR*

Related: `gaussian` Set up unshifted Gaussian window function (M)
`ni` Number of increments in 1st indirectly detected dimension (P)
`ni2` Number of increments in 2nd indirectly detected dimension (P)
`pi3ssbsq` Set up pi/3 shifted sinebell-squared window function (M)
`sqcosine` Set up unshifted cosine-squared window function (M)
`sq sinebell` Set up unshifted sinebell-squared window function (M)

pilot **Automatic sequence setup (P)**

Applicability: Systems with imaging capabilities.

Description: Provides a degree of automatic setup of a sequence, where this capability is available. If `pilot='y'`, access is provided to automatic setting for the gradients `gssr` and `gror`. These gradient levels are then adjusted to compensate for gradient slew rate. The adjustments are made at the time of `go`; however, the values used are not returned to the parameter set.

Values: 'y' means the automatic mode is on.
'n' means the manual mode is set.

See also: *User Guide: Imaging*

Related: `go` Submit experiment to acquisition (C)
`gror` Readout compensation gradient (P)
`gssr` Slice selection refocusing gradient (P)

pintvast Plots of integral regions (M)

Applicability: Systems with VAST accessory.

Syntax: `pintvast (last)`

Description: `pintvast` plots the integrals of the partial regions of each spectra from wells 0 to `last`.

Arguments: `last` is the number last sample well. The default is 96.

See also: *User Guide: Liquids NMR*

Related: `intvast` Builds text file the integral regions (M)

pipat Shape of an inversion pulse (P)

Applicability: Systems with imaging capabilities.

Description: Specifies the shape of inversion pulse `pi`.

Values: 'hard', 'sinc', 'gauss', 'sech', 'sine', or any shape resident in the system pulse shape library or libraries.

See also: *User Guide: Imaging*

Related: `ir` Inversion recovery mode (P)
`pi` Width of an inversion pulse (P)

pir Plot integral amplitudes below spectrum (C)

Syntax: `pir`

Description: Plots integral amplitudes below the appropriate spectral regions.

See also: *Getting Started*

Related: `dpf` Display peak frequencies over spectrum (C)
`dpir` Display integral amplitudes below spectrum (C)
`dpirn` Display normalized integral amplitudes below spectrum (M)
`pirn` Plot normalized integral amplitudes below spectrum (M)
`ppf` Plot peak frequencies over spectrum (M)

pirn Plot normalized integral amplitudes below spectrum (M)

Syntax: `pirn`

Description: Equivalent to the command `pir` except that the sum of the integrals is normalized to the value of the parameter `ins`.

See also: *Getting Started*

Related: `dpirn` Display normalized integral amplitudes below spectrum (M)
`ins` Integral normalization scale (P)
`pir` Plot integral amplitudes below spectrum (C)

pkpick **Peak pick (P)**

Description: Stores the string command to do peak picking. `pkpick` is a string parameter set by the *GLIDE* interactive window tool, stored in `global`, and then executed within macros.

See also: *Getting Started*

Related: `parstyle` Parameter style (P)

p1 **Plot spectra (C)**

Syntax: `p1<(<start,finish<,step>><,<'int'><,<'all'><
<,options>>>`

Description: Plots one or more spectra. When a single spectrum is plotted, integral plotting is controlled by the parameter `intmod` as follows: `intmod='off'` turns off the integral plot, `intmod='full'` plots the entire integral, and `intmod='partial'` plots every other integral region.

For arrayed 1D spectra or for 2D spectra, a particular trace can be plotted by supplying the index number as an argument. For 2D data sets, spectra can be plotted from either the `f1` or `f2` domain by setting the parameter `trace` to `'f1'` or `'f2'`, respectively. After the command `ft1d`, interferograms can be plotted by setting `trace='f1'` and then typing `p1`. Multiple spectra can be plotted by supplying the indexes of the first and last spectra.

The position of the first spectrum is governed by the parameters `wc`, `sc`, and `vp`. For 1D data, subsequent spectra are positioned relative to the preceding spectrum by the vertical and horizontal offset parameters `vo` and `ho`. For 2D data, `ho` defines the total horizontal offset between the first and last spectrum. Also for 2D data, `vo` is inactive while the parameter `wc2` defines the total vertical offset between the first and last spectrum.

The parameter `cutoff`, if it exists and is active, defines the distance above and below the current vertical position `vp` at which peaks are truncated. By arraying `cutoff` to have two different values, truncation limits above and below the current vertical position can be controlled. For example, `cutoff=50` truncates peaks at `vp+50` mm and `vp-50` mm. `cutoff=50,10` truncates peaks at `vp+50` mm and `vp-10` mm.

Arguments: `start` is the index of a particular trace for arrayed 1D or 2D spectra. For multiple spectra, `start` is the index of the first spectrum.

`finish` is the index of the last spectrum for multiple spectra.

`step` specifies the increment for the spectral index. The default is 1.

`'int'` is a keyword that specifies displaying only the integral, independently of the value of `intmod`.

`'all'` is a keyword to plot all of the spectra. This value is the default.

`options` can be any of the following keywords:

- `'top'` or `'side'` cause the spectrum to be plotted either above or at the left edge of a contour plot. This assumes that the parameters `sc`, `wc`, `sc2`, and `wc2` are those used to position the contour plot.

- 'dodc' causes all spectra to be drift corrected independently.
- 'pen1', 'pen2', 'pen3', etc. specify a pen number on a plotter.

Examples: `p1`
`p1(1,6,2)`

Alternate: Plot button in the 1D Plotting Menu,

See also: *Getting Started; User Guide: Liquids NMR*

Related:	<code>cutoff</code>	Data truncation limit (P)
	<code>dssa</code>	Display stacked spectra automatically (C)
	<code>dsww</code>	Display spectra in whitewash mode (C)
	<code>ft1d</code>	Fourier transform along f_2 dimension (C)
	<code>ho</code>	Horizontal offset (P)
	<code>intmod</code>	Integral display mode (P)
	<code>plww</code>	Plot spectra in whitewash mode (C)
	<code>sc</code>	Start of chart (P)
	<code>sc2</code>	Start of chart in second direction (P)
	<code>trace</code>	Mode for 2D data display (P)
	<code>vo</code>	Vertical offset (P)
	<code>vp</code>	Vertical position of spectrum (P)
	<code>wc</code>	Width of chart (P)
	<code>wc2</code>	Width of chart in second direction (P)

p12d Plot 2D spectra in whitewash mode (C)

Syntax: `p12d('nobase' | 'fill' | 'fillnb')>`

Description: Plots a stacked plot of 2D spectra in whitewash mode (after the first spectra, each spectra is blanked out in regions in which it is behind an earlier spectra). Color does not represent intensity (unlike `dcon`), since intensity can be seen visually, but instead successive traces are displayed in different colors so that color represents frequency. The horizontal offset parameter `ho` is not active for this command.

Arguments: 'nobase' is a keyword to activate `th` to suppress intensity below `th`.
'fill' is a keyword to fill in the peaks. Note that if 'fill' (or 'fillnb') is used, `th` operates linearly and not logarithmically (with factors of 2) as it does in contour or color intensity displays.
'fillnb' is a keyword to combine base suppression and peak filling.

Examples: `p12d`
`p12d('nobase')`

See also: *User Guide: Liquids NMR*

Related:	<code>dcon</code>	Display noninteractive color intensity map (C)
	<code>ds2d</code>	Display 2D spectra in whitewash mode (C)
	<code>dsww</code>	Display spectra in whitewash mode (C)
	<code>ho</code>	Horizontal offset (P)
	<code>plww</code>	Plot spectra in whitewash mode (C)
	<code>th</code>	Threshold (P)

plan Display menu for planning a target scan (M)

Applicability: Systems with imaging capabilities.

Syntax: `plan`

Description: Brings up a menu that provides access to the target scan planning utilities. The `plan` menu has three buttons: Slice, Voxel, and Exit.

The Slice button provides access to the slice planning menu. The user first clears the current experiment of any `mark2d.out` files using the Clear Marks button. The image display may then be made interactive using the Interactive View button. This activates the `dconi` program. The user should select and mark two points that lie on the edge of the desired target slice plane using the Mark button of the `dconi` menu. To write the mark data into the `mark2d.out` file, the user should exit `dconi` using the Return button. This exits to the slice planner menu.

The target slice selection can be shown graphically on the image display using the Show Target button of the slice planner menu. This button uses the `drawslice` macro. The slice parameters (`pss`, `psi`, `phi`, and `theta`) are calculated and set using the Calculate Target button of the slice planner menu. This button uses the `ssplan` macro. This program creates the string parameter `planlock` and assigns it the value `'ssplan'`. This prevents a user inadvertently performing a second planning operation without applying the `reset` command to restore the original parameters for the scout data.

At this point, the current parameters of the scout experiment contain the data needed to acquire the desired slice. The user can use these directly or use the `mp` or transfer commands to move the information to another experiment.

The Voxel button of the `plan` menu provides access to the voxel planning menu. The user may enter the interactive mode using the Interactive View button. This activates the `dconi` program. The user should clear any previous unwanted planning information before starting.

The size and position of the voxel face parallel to the image plane can be selected by positioning the 2D box cursor. Once this is done, the user leaves the interactive mode using the Return button of the `dconi` menu. This returns the user to the voxel planning menu. The user can plan for more than one voxel. These target voxel selections can be shown graphically on the image display using the Show Target button of the planner menu. This button uses the `drawvox` macro. The parameter for the voxel can be calculated and set using the Calculate Target button, which uses the `voxplan` macro.

The `voxplan` macro requests the user to enter the voxel size in the direction parallel to the scout image slice select axis. Voxel parameters are computed from the 2D box cursor data and user entry. The voxel center is taken to lie in the scout image plane at the center of the 2D box. `voxplan` also creates the string parameter `planlock` and assigns it the value `'voxplan'`. This provides an interlock against further planning operations. The `reset` command restores the original scout parameters and removes the `planlock` parameter.

The current parameters of the scout experiment contain the data needed to acquire the voxel. The user *must* use the transfer program to copy this data to the parameter set of a suitable voxel selective sequence.

See also: *User Guide: Imaging*

Related:	<code>drawslice</code>	Display target slices (M)
	<code>drawvox</code>	Display target voxels (M)
	<code>mp</code>	Move parameters between experiments (C)
	<code>phi</code>	Euler angle phi from magnet frame (P)
	<code>planlock</code>	Planner lockout (P)
	<code>psi</code>	Euler angle psi from magnet frame (p)
	<code>pss</code>	Slice position (P)
	<code>ssplan</code>	Set slice parameters for target slice (M)
	<code>theta</code>	Euler angle theta from magnet frame (P)
	<code>voxplan</code>	Set voxel parameters for voxel defined by 2D box cursor (M)

plane **Currently displayed 3D plane type (P)**

Applicability: All systems; however, although `plane` is available on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*, such systems can only process 3D data and cannot acquire 3D data.

Description: Stores the type of 3D plane currently displayed within VNMR. If `plane` does not exist, it is created by the macro `par3d`. The command `select`, as well as the many macros that make use of `select`, requires the parameter `plane` to exist for 3D data sets and to contain an appropriate value.

`plane` is set automatically by the macro `getplane`; it can also be set by the macro `ft3d` if automatic plane extraction is requested at the end of the 3D FT. The order of priority for the plane types is 'f1f3', 'f2f3', and then 'f1f2'. In other words, if `getplane` is requested to extract the f_1f_3 and the f_2f_3 planes, `plane` will be set to 'f1f3'. `plane` can also be set manually.

Values: 'f1f3', 'f3f1', 'f2f3', 'f3f2', 'f1f2', or 'f2f1'

See also: *User Guide: Liquids NMR*

Related:	<code>dplane</code>	Display a 3D plane (M)
	<code>dproj</code>	Display a 3D plane projection (M)
	<code>dsplanes</code>	Display a series of 3D planes (M)
	<code>ft3d</code>	Perform a 3D Fourier transform on a 3D FID data set (M,U)
	<code>getplane</code>	Extract planes from a 3D spectral set (M)
	<code>nextpl</code>	Display the next 3D plane (M)
	<code>par3d</code>	Create 3D acquisition, processing, display parameters (C)
	<code>path3d</code>	Number of complex points to left-shift np FID (P)
	<code>plplanes</code>	Plot a series of 3D planes (M)
	<code>prevpl</code>	Display the previous 3D plane (M)
	<code>select</code>	Select a spectrum or 2D plane without displaying it (C)

planlock **Planner lock (P)**

Applicability: Systems with imaging capabilities.

Description: Created by `voxplan` and assigned the value 'voxplan' to provide an interlock against further planning operations. This parameter is also created by the `ssplan` macro and assigned the value 'ssplan' to prevents a user inadvertently performing a second planning operation. In both cases, the `reset` command removes the value assigned to `planlock`.

See also: *User Guide: Imaging*

Related:	<code>plan</code>	Display menu for planning a target scan (M)
	<code>ssplan</code>	Set slice parameters for target slice (M)
	<code>voxplan</code>	Set voxel parameters for voxel defined by 2D box cursor (M)

plapt **Plot APT-type spectra automatically (M)**

Syntax: `plapt<(13Cexp_number)>`

Description: Automatically plots APT spectra. The APT spectrum is plotted on top of a standard carbon spectrum if either an experiment with such data is specified or if a file C13 is found in `curexp+ /subexp'`. If neither such a subfile is found nor an experiment with standard carbon data is specified, the APT spectrum is plotted alone.

Arguments: `13Cexp_number` specifies the number, from 1 to 9, of an experiment with a standard ^{13}C spectrum.

Examples: `plapt`
`plapt(2)`

See also: *Getting Started*

Related: `curexp` Current experiment directory (P)

plarray Plotting macro for arrayed 1D spectra (M)

Syntax: `plarray`

Description: A generic macro for plotting arrayed 1D spectra. `plarray` is called by the `plot` macro, but can also be used directly. For the plot layout, `procarray` distinguishes between arrays with few elements (6 or less), which will be stacked vertically (no horizontal offset), and spectra with many (greater than 6) elements. Those are stacked horizontally by default, unless there are too many lines, in which case a diagonally stacked display is chosen. Horizontal stacking is mostly adequate for pulse and power calibrations, where there are usually few lines only; diagonally stacked displays/plots are frequently chosen for T_1 and T_2 experiments on entire spectra, often with many lines.

The automatic stacking mode can be overridden by creating and setting a string parameter `stackmode` in the startup macro or before calling `procplot` or `procarray`. Possible values for `stackmode` are 'horizontal', 'vertical', or 'diagonal'. DEPT-type spectra can, in principle, also be processed with `procarray`, but no DEPT editing occurs, of course.

See also: *Getting Started*

Related: `aexmpl` Automatic expansion plot (M)
`plc` Plot carbon spectrum (M)
`plh` Plot proton spectrum (M)
`plot` Automatically plot spectra (M)
`procarray` Process arrayed 1D spectra (M)
`stackmode` Stack control for processing arrayed 1D spectra (P)

plate_glue Define a glue order for plotting and display (U)

Applicability: Systems with VAST accessory

Syntax: `plate_glue`

Description: In a Unix terminal or shell window type `plate_glue`. The glue order is determined by clicking on the wells to be displayed. Save the glue order file in the user's `vnmsys/templates/glue` directory.

See also: *User Guide: Liquids NMR*

Related: `dsvast2d` Display VAST data in a pseudo-2D format (M)
`plvast` Plot VAST data in a stacked 1D-NMR matrix (M)
`plvast2d` Plot VAST data in a pseudo-2D format (M)

plc Plot a carbon spectrum (M)

Syntax: `plc<(pltmod)>`

Description: Plots a carbon spectrum based on the parameters `pltmod` (the options 'off', 'full', and 'fixed' are implemented) and `intmod` ('off', 'full', and 'partial' are implemented). Peak frequency labels, in ppm, are usually plotted.

Arguments: `pltmod` is an alternate value of `pltmod` for this macro only. The value of the `pltmod` parameter is not changed.

Examples: `plc`
`plc('full')`

See also: *Getting Started*

Related: `intmod` Integral display mode (P)
`pltmod` Plotter display mode (P)

plcosy Plot COSY- and NOESY-type spectra automatically (M)

Syntax: `plcosy(<'pos' | 'neg'><, ><levels<, spacing<, exp1D>>>)`

Description: Automatically plots 2D COSY- and NOESY-type spectra (homonuclear correlated spectra). Features include the following:

- Keeps the orientation (f_1 , f_2) of the spectrum on the screen.
- Plot area is optimized.
- Number of contour levels and their spacing can be selected.
- Negative or positive contours can be suppressed.
- 1D traces can be plotted along both axes; such 1D traces are taken from a full (or reduced) 1D spectrum in an other experiment, or from a subfile from within the current experiment.
- Works correctly for expansions.
- 1D traces can be suppressed, allowing a larger area for the 2D spectrum.
- 1D spectrum can be in any experiment.
- With phase-sensitive spectra using a plotter with one pen or a printer such as a LaserJet, if 'pos' or 'neg' are not selected, seven positive levels (or the specified number of positive contours) and one negative level are plotted, to distinguish positive and negative signals.

In multiexperiment mode, for the first plot, the experiment with the 1D spectrum should be specified (at least if it is not in exp1). From then on, the 1D spectrum will be stored *within* the experiment with the 2D spectrum, which allows much faster switching between spectra and also frees the other (1D) experiment for other tasks. Because of this internal storage, the `exp1D` argument is not required for subsequent plots.

Arguments: 'pos' is a keyword to plot only positive contours.

'neg' is a keyword to plot only negative contours.

`levels` is the number of contour levels. The default is 7.

`spacing` is the spacing between the contours. The default is 2.

`exp1D` is the experiment in which the proton 1D spectrum resides. This can be a full 1D spectrum, but the referencing must be the same as for the 2D. A negative number suppresses the proton trace. The default is from a subfile.

Examples: `plcosy`
`plcosy(12, 1.5)`
`plcosy('pos', 7, 2, 3)`
`plcosy(7, 2, -1)`
`plcosy('neg')`

See also: *Getting Started*

pldept Plot DEPT data, edited or unedited (M)

Syntax: `pldept`

Description: Plots out DEPT data, either edited or not edited.

Alternate: Plot button in the Automatic DEPT Analysis Menu.

See also: *User Guide: Liquids NMR*

Related: **adept** Automatic DEPT analysis and spectrum editing (C)
autodept Automated complete analysis of DEPT data (M)
deptproc Process DEPT data (M)
padept Perform adept analysis and plot resulting spectra (C)

plfid **Plot FIDs (C)**

Syntax: `plfid(<start><,finish><,step><,'all'|'imag'>
<,pen>>`

Description: Plots one or more FIDs. The position of the first FID is governed by the parameters **wc**, **sc**, and **vpf**. A subsequent FID is positioned relative to the preceding FID by the vertical and horizontal offset parameters **vo** and **ho**.

Arguments: **start** is the index of a particular FID for arrayed 1D or 2D data sets. For multiple FIDs, **start** is the index of the first FID.
finish is the index of the last FID for multiple FIDs. To include all FIDs, set **start** to 1 and **finish** to the parameter **arraydim** (see example).
step specifies the increment for the FID index. The default is 1.
'all' is a keyword to plot all of the FIDs. This is the default.
'imag' is a keyword to plot the imaginary FID channel only. The default is **'all'**.

pen is a keyword with the plotter pen number: **'pen1'**, **'pen2'**, **'pen3'**, etc. The default is **'pen1'**.

Examples: `plfid(1,arraydim,3)`

See also: *Getting Started*

Related: **arraydim** Dimension of experiment (P)
dfs Display stacked FIDs (C)
dfww Display FIDs in whitewash mode (C)
ho Horizontal offset (P)
sc Start of chart (P)
vo Vertical offset (P)
vpf Current vertical position of FID (P)
wc Width of chart (P)

plfit **Plot deconvolution analysis (M)**

Syntax: `plfit`

Description: Produces a complete output plot of a deconvolution analysis, plotting the observed spectrum, the full calculated spectrum, each individual component, as well as the numerical results of the analysis.

Alternate: Plot button in the Deconvolution Menu.

See also: *User Guide: Liquids NMR*

Related: **fitspec** Perform spectrum deconvolution (C)
showfit Display numerical results of deconvolution (M)
usemark Use "mark" output as deconvolution starting point (M)

plgrid **Plot a grid on a 2D plot (M)**

Syntax: (1) `plgrid(<spacing><,><pen>)>`
 (2) `plgrid(<start_f2,incr_f2,start_f1,incr_f1<,pen>)>`

Description: Plots grid lines over a 2D plot.

Arguments: `spacing` specifies the approximate spacing of the grid lines, in cm. The default is intervals of approximately 1 cm, rounded so that the intervals fall at a multiple of 1, 2, or 5 (in Hz) or 1p, 2p, or 5p (in ppm).

`pen` is a keyword with the plotter pen number: 'pen1', 'pen2', 'pen3', etc. The default is 'pen1'.

`start_f2`, `incr_f2`, `start_f1`, `incr_f1` define the starting and increment frequencies in both f_2 and f_1 for a grid. Add the p suffix to a value to enter it in ppm (see last example below).

Examples: `plgrid`
`plgrid(2)`
`plgrid('pen5')`
`plgrid(1.5,'pen2')`
`plgrid(1p,0.5p,3p,0.5p)`

See also: *User Guide: Liquids NMR*

Related: [grid](#) Draw a grid on a 2D display (C)

plh **Plot proton spectrum (M)**

Syntax: `plh<(pltmod)>`

Description: Plots a proton spectrum based on the parameters `pltmod` (the options 'off', 'fixed', 'full', and 'variable' are implemented) and `intmod` ('off', 'full', and 'partial' are implemented).

Arguments: `pltmod` is an alternate value of the parameter `pltmod` for this macro only. The value of the `pltmod` parameter is not changed.

Examples: `plh`
`plh('full')`

See also: *Getting Started*

Related: [intmod](#) Integral display mode (P)
[pltmod](#) Plotter display mode (P)
[sp](#) Start of plot (P)
[wp](#) Width of plot (P)

plhet2dj **Plot heteronuclear J-resolved 2D spectra automatically (M)**

Syntax: `plhet2dj(<'pos' | 'neg'<,>levels<,>spacing<,>exp1D>>>)>`

Description: Automatically plots 2D spectra of type HET2DJ (heteronuclear J-resolved 2D spectra) with the following features:

- Displayed portion of the spectrum is plotted in f2-mode
- Plot area is optimized
- Number of contour levels and their spacing can be selected
- Negative or positive contours can be suppressed
- A 1D trace can be plotted along the f_2 axis; such a 1D trace is taken from a full (or reduced) 1D spectrum in an other experiment, or from a file from within the current experiment.
- Expansions are handled correctly

- The 1D trace can be suppressed, which allows using a larger area for the 2D spectrum
- The 1D spectrum can be in any experiment
- With phase-sensitive spectra, if 'pos' or 'neg' are not selected and the plotter has only one pen (also for printers like the LaserJet), the specified number of positive contours are plotted (default is 7), but only one negative level, to distinguish positive and negative signals.

In multiexperiment mode, for the first plot the experiment with the 1D spectrum should be specified (at least if it is not in `exp1`). From then on, the 1D spectrum is stored *within* the experiment with the 2D spectrum, which allows much faster switching between the spectra and also frees the other 1D experiment for other tasks. Because of this internal storage, the `exp1D` argument is not required for subsequent plots.

Arguments: 'pos' is a keyword to only plot positive contours

'neg' is a keyword to only plot negative contours

levels is the number of contour levels. The default is 7.

spacing is the spacing between the contours. The default is 2.

exp1D is the number from 1 to 9 of the experiment in which the 1D spectrum resides. This can be a full 1D spectrum, but the referencing must be the same as for the 2D. A negative number will suppress the 1D trace. The default is 1 (for `exp1`).

Examples: `plhet2dj`
`plhet2dj(12,1.5)`
`plhet2dj('pos',7,2,3)`
`plhet2dj(7,2,-1)`

See also: *Getting Started*

plhom2dj Plot homonuclear J-resolved 2D spectra automatically (M)

Syntax: (1) `plhom2dj<(levels<, spacing<, exp1D>>>>`
 (2) `plhom2dj('pos' | 'neg' <, levels<, spacing<, exp1D>>>>`

Description: Automatically plots 2D spectra of type HOM2DJ (homonuclear J-resolved 2D spectra). Features include the following:

- The displayed portion of the spectrum is plotted in f_2 -mode
- The plot area is optimized
- Number of contour levels and their spacing can be selected
- Negative or positive contours can be suppressed
- A 1D trace can be plotted along the f_2 axis; such a 1D trace is taken from a full (or reduced) 1D spectrum in an other experiment, or from a file from within the current experiment.
- It also works correctly for expansions
- The 1D trace can be suppressed, which allows using a larger area for the 2D spectrum
- The 1D spectrum can be in any experiment
- With phase-sensitive spectra, if 'pos' or 'neg' are not selected and the plotter has only 1 pen (also for printers like the LaserJet) 7 or the specified number of positive contours are plotted, but only one negative level, to distinguish positive and negative signals.

In multiexperiment mode, for the first plot the experiment with the 1D spectrum should be specified (at least if it is not in `exp1`). From then on, the 1D spectrum will be stored *within* the experiment with the 2D spectrum, which allows much faster switching between the spectra and also frees the other (1D) experiment for other tasks. Because of this internal storage, the `exp1D` argument is not required for subsequent plots.

Arguments: `levels` is the number of contour levels. The default is 7.
`spacing` is the spacing between the contours. The default is 2.
`exp1D` is a number from 1 to 9 for the experiment in which the 1D spectrum resides. The spectrum can be a full 1D spectrum but the referencing must be the same as for the 2D. A negative number will suppress the 1D trace. The default is 1 (for `exp1`).
`'pos'` specifies only plot positive contours.
`'neg'` specifies only plot negative contours.

Examples: `plhom2dj`
`plhom2dj(25,1.2)`
`plhom2dj('pos',7,2,3)`
`plhom2dj(7,2,-1)`

See also: *Getting Started*

plhxcor **Plot X,H-correlation 2D spectrum (M)**

Syntax: `plhxcor(<'pos' | 'neg'><, ><levels<, spacing
<, exp1D_H<, exp1D_X>>>>)`

Description: Automatically plots 2D spectra of type HETCOR, COLOC, HMQC, HMBC (direct and indirect detection). Features include the following:

- Keeps the orientation (f_1 , f_2) of the spectrum on the screen.
- Plot area is optimized.
- Number of contour levels and their spacing can be selected.
- Negative or positive contours can be suppressed.
- 1D proton and X traces can be plotted along both axes; such 1D traces are taken from full (or reduced) 1D spectra in other experiments or subfile within the current experiment.
- Works correctly for expansions.
- 1D traces can be suppressed, allowing a larger area for the 2D spectrum.
- 1D spectra can be in any experiment.

Arguments: `'pos'` is a keyword to plot only positive contours.
`'neg'` is a keyword to plot only negative contours.
`levels` is the number of contour levels. The default is 7.
`spacing` is the spacing between the contours. The default is 2.
`exp1D_H` is a number from 1 to 9 of the experiment in which the proton 1D spectrum resides; this can be a full 1D spectrum, but the referencing must be the same as for the 2D. A negative number will suppress the proton trace. The default is a subfile in the current experiment.
`exp1D_X` is a number from 1 to 9 of the experiment in which the X 1D spectrum resides. A negative number suppresses the X trace. the default is a subfile in the current experiment.

Examples: `plhxcor(12,1.5)`
`plhxcor(7,2,3)`
`plhxcor(7,2,1,3)`
`plhxcor('pos',7,2,-1,3)`
`plhxcor(7,2,-1,-1)`
`plhxcor('neg')`

See also: *Getting Started; User Guide: Liquids NMR*

Related: `hetcor` Set up parameters for HETCOR pulse sequence (M)

p1list Active pulse length parameter list (P)

Applicability: Systems with imaging capabilities.

Description: Contains an array of strings, whose values are the names of the rf pulse length parameters used by the sequence (e.g., `p1list='p1','p2','p3'`). The `nD`, `seqcon`, `p1list`, `patlist`, `pwrlist`, `fliplist` and `sslist` parameters configure a particular parameter set for an application sequence defined by the value of the `seqfil` parameter. The `p1list`, `patlist`, `pwrlist`, `fliplist` and `sslist` parameters provide information concerning the rf pulse and conjugate gradients used by the sequence.

See also: *User Guide: Imaging*

Related: `fliplist` Standard flip angle list (P)
`gcoil` Read data from gradient calibration tables (P)
`nD` Application dimension (P)
`patlist` Active pulse template parameter list (P)
`pwrlist` Active pulse power level parameter list (P)
`rfcoil` RF pulse calibration identity (P)
`seqcon` Acquisition loop control (P)
`seqfil` Application object code name (P)
`sslist` Conjugate gradient list (P)

p11 Plot a line list (M)

Syntax: `p11<(x,y,minimum_y)>`

Description: Produces a columnar line list on a plotter, similar to what would appear on a printer. `p11` is quite different from the alternative method of plotting peak frequencies using `ppf`. The output of `p11` is automatically formatted into multiple columns, depending on the number of lines.

Arguments: `x` is the x position of the upper left of the line list.

`y` is the y position of the upper left of the line list.

`minimum_y` is the minimum y at which to reset back to top.

Examples: `p11`
`p11(20,150)`
`p11(5,wc2max*.8,wc2max*.5)`

See also: *Getting Started*

Related: `ppf` Plot peak frequencies over spectrum (M)

p112d Plot results of 2D peak picking (C)

Syntax: `p112d<(options)>`

Description: Plots the results of applying the `112d` command to pick 2D peaks in a 2D spectrum or a 2D plane of a 3D spectrum. Refer to the description of `112d` for a description of the process and the options available.

See also: *User Guide: Liquids NMR*

Related: `112d` Automatic and interactive 2D peak picking (C)

`plot` Automatically plot spectra (M)

Syntax: `plot`

Description: A universal plotting macro normally called through the `procplot` macro (which by itself serves as processing and plotting facility for automatic experiments). `plot` can also be used directly by the user who then doesn't have to remember specific plotting macros. Of course, the specialized macros can still be called directly if the user know their names.

The main purpose of `plot` is to automatically call the correct specialized plotting macro, depending on the user definition or otherwise on the type of data in the experiment. A plotting macro is selected automatically as follows:

APT spectra:	<code>plapt</code>
other, non-arrayed 1D data:	<code>plot1d</code>
DEPT type arrayed spectra:	<code>pldept</code>
other arrayed 1D spectra:	<code>plarray</code>
J-resolved 2D spectra:	<code>pl2dj</code>
homonuclear correlation 2D spectra:	<code>plcosy</code>
heteronuclear correlation 2D spectra:	<code>plhxcor</code>

Other types of 2D spectra (mostly multiple-quantum 2D spectra such as 2D-INADEQUATE) are not plotted automatically at this time. For phase-sensitive 2D spectra, automatic plotting is only provided if they were acquired using the method described by States, Haberkorn, and others; TPPI spectra are not covered.

Note that plot macros in general should not adjust the phase, the vertical scale, or change the integral size and reset points; these are assumed to be adjusted either by hand or by a suitable processing macro like `procplot` and the macros called therein. The plotting macros only make adjustments in order to make spectrum and parameters fit onto the page the desired way.

See also: *Getting Started*

Related:	<code>plapt</code>	Plot APT spectra (M)
	<code>plarray</code>	Plot arrays (M)
	<code>plcosy</code>	Plot homonuclear 2D correlation spectra (M)
	<code>pldept</code>	Plot DEPT type spectra (M)
	<code>plhxcor</code>	Plot heteronuclear correlation spectra (M)
	<code>plot1d</code>	Plot 1D spectra (M)
	<code>procplot</code>	Automatically process FIDs (M)

`plot1d` Plotting macro for simple (non-arrayed) 1D spectra (M)

Syntax: `plot1d`

Description: A generic macro for plotting non-arrayed 1D spectra using a set of standard macros. `plot1d` is called by the `plot` macro, but can also be used directly. `plot1d` first tries to find a specific macro (e.g., `plh`, `plc`, `plp`) for the current

observe nucleus. If such a macro exists, it is called. If a nucleus-specific macro is not found in the command path, a “minimal” 1D plot is produced.

See also: *Getting Started*

Related: `plc` Plot carbon spectrum (M)
`plh` Plot proton spectrum (M)
`plp` Plot phosphorus spectrum (M)
`plot` Automatically plot spectra (M)

plot2D Plot 2D spectra (M)

Syntax: `plot2D('pos' | 'neg' | 'both', levels, spacing, \`
`'top' | 'notop' | 'proj', 'side' | 'noside' | 'proj')`

Description: Checks for the presence of appropriate proton or carbon high-resolution spectra in the directory `userdir+' /data/' +sample` and decides to plot high resolution spectra or a projection depending on whether or not the proton or carbon spectrum exists.

Arguments: `'pos'` is a keyword to plot positive contours.
`'neg'` is a keyword to plot negative contours.
`'both'` is a keyword to plot both positive and negative contours.
`levels` is the number of levels to be plotted.
`spacing` is the spacing between contour levels.
`'top'` is a keyword to plot a high-resolution spectrum on the top.
`'notop'` is a keyword to plot a non-high-resolution spectrum or projection.
`'proj'` is a keyword to plot a projection on top.
`'side'` is a keyword to plot a high-resolution spectrum on the side.
`'noside'` is a keyword to plot a non-high-resolution spectrum or projection.
`'proj'` is a keyword that plots a projection on the side.

Examples: `plot2D('pos', 2, 5, 'top', 'side')`

Related: `Autoplot2D` Check for *GLIDE*-selected plot options (M)
`plot` Automatically plot spectra (M)
`plotside` Plot spectrum on side (M)
`plottop` Plot spectrum on top (M)
`plottopside` Plot spectrum on top and side (M)

plotside Plot spectrum on side (M)

Syntax: `plotside`

Description: Plots projection or high-resolution spectrum on the side of a 2D spectrum. `plotside` is used with `plot2D` and is not useful by itself.

Related: `plot2D` Plot 2D spectra (M)

plotter Plotter device (P)

Description: Sets the plotter in use on the system.

Values: A string with entries such as `'DraftPro'`, `'ThinkJet_96'`, `'LaserJet_300'`, `'jim'`, `'varian1'`, and `'Laser1'`.

See also: *Getting Started*

Related: `setplotdev` Return characteristics of a named plotter (C)
`showplotter` Show list of currently defined plotters and printers (M)

plottop **Plot spectrum on top (M)**

Syntax: `plottop`

Description: Plots projection or high resolution spectra on the top of a 2D spectrum. `plottop` is used with `plot2D` and is not useful by itself.

Related: `plot2D` Plot 2D spectra (M)

plottopside **Plot spectrum on top and side (M)**

Syntax: `plottopside`

Description: Plots projection or high-resolution spectrum on the top and side of a 2D spectrum. `plottopside` is used with `plot2D` and is not useful by itself.

Related: `plot2D` Plot 2D spectra (M)

plp **Plot phosphorus spectrum (M)**

Syntax: `plp<(pltmod)>`

Description: Plots a phosphorus spectrum based on the parameters `pltmod` (the options 'off', 'full', and 'fixed' are implemented) and `intmod` ('off', 'full', and 'partial' are implemented). Peak frequency labels, in ppm, are usually plotted.

Arguments: `pltmod` is an alternate value of `pltmod` for this macro only. The value of the `pltmod` parameter is not changed.

Examples: `plp`
`plp('full')`

See also: *Getting Started*

Related: `intmod` Integral display mode (P)
`plh` Plot proton spectrum (M)
`pltmod` Plotter display mode (P)

plplanes **Plot a series of 3D planes (M)**

Applicability: All systems; however, although `plplanes` is available on *GEMINI 2000* systems, such systems can only process 3D data and cannot acquire 3D data.

Syntax: `plplanes(start_plot,stop_plot<,'pos'|'neg'>
<,number_levels><,spacing>)`

Description: Creates the 2D contour plots for a subset of the 3D planes specified by the parameter `plane`.

Arguments: `start_plot` specifies the number, greater than 0, of the 3D plane with which plotting is to begin.

`stop_plot` specifies the number of the 3D plane with which plotting is to end. If `start_plot` is greater than `stop_plot`, only the first plane, whose number is `start_plot`, is plotted. The range of `stop_plot` depends on the value of the parameter `plane`:

- if `plane='f1f3'`, `stop_plot` is between 0 and `fn2/2`
- if `plane='f2f3'`, `stop_plot` is between 0 and `fn1/2`

- if `plane='f1f2'`, `stop_plot` is between 0 and `fn/2`

'`pos`' is a keyword specifying that phase-sensitive spectra plot positive peaks only. The default is to plot both positive and negative peaks.

'`neg`' is a keyword specifying that phase-sensitive spectra plot negative peaks only. The default is to plot both positive and negative peaks.

`levels` is maximum number of contour levels to plot. The default is 4.

`spacing` is relative intensity of successive contour levels. The default is 2.

Note that the optional arguments '`pos`' | '`neg`', `number_levels`, and `spacing` are for the VNMR plotting command `pcon`.

Examples: `plplanes(1,3)`
`plplanes(2,3,'pos',4)`

See also: *User Guide: Liquids NMR*

Related:	<code>dplane</code>	Display a 3D plane (M)
	<code>dproj</code>	Display a 3D plane projection (M)
	<code>dsplanes</code>	Display a series of 3D planes (M)
	<code>getplane</code>	Extract planes from 3D spectral data set (M)
	<code>nextpl</code>	Display the next 3D plane (M)
	<code>path3d</code>	Path to currently displayed 2D planes from a 3D data set (P)
	<code>pcon</code>	Plot contours on a plotter (C)
	<code>plane</code>	Currently displayed 3D plane type (P)
	<code>prevpl</code>	Display the previous 3D plane (M)

`plttext` **Plot text file (M)**

Syntax: `plttext(<file><,x<,y<,width>>>>>`
`<:$x_next,$y_next,$y_increment>`

Description: Plots a text file.

Arguments: `file` is the name of a text file. The default is the current experiment text file.

`x` and `y` are coordinates, in mm, of the first line of text. This positions the location of the output. The default is the upper left-hand corner of the page.

`width` is the maximum column text width, in characters. `plttext` uses a word wrap to make the text fit into the width specified.

`$x_next` and `$y_next` are the coordinates where the start of the next line would have been plotting. This is useful for subsequent character plotting.

`$y_increment` is the vertical increment between lines.

Examples: `plttext`
`plttext(wcmax-70)`
`plttext(userdir+'/exp3/text')`
`plttext(100,100)`
`plttext(userdir+'/exp4/text',200,200,24)`
`plttext:$x,$y,$dy`

See also: *Getting Started*

Related:	<code>dtext</code>	Display a text file in the graphics window (C)
	<code>ptext</code>	Print out a text file (M)
	<code>text</code>	Display text or set new text for current experiment (C)
	<code>userdir</code>	VNMR user directory (P)

`pltmod` **Plotter display mode (P)**

Description: Controls plotting of a proton, carbon, or phosphorus spectrum.

Values: 'off' sets no plotting.
 'fixed' takes `sp` and `wp` as is.
 'full' adjusts `sp` and `wp` to plot the full spectrum.
 'variable' adjusts `sp` and `wp` to plot only the region of interest.

See also: *Getting Started*

Related: `plc` Plot carbon spectrum (M)
`plh` Plot proton spectrum (M)
`plp` Plot phosphorus spectrum (M)
`sp` Start plot (P)
`wp` Width of plot (P)

plvast Plot VAST data in a stacked 1D-NMR matrix format (M)

Applicability: Systems with the VAST accessory.

Syntax: `plvast<(display order, number of columns plotted)>`

Description: `plvast` arranges and plots the traces from a reconstructed 2D data set (see `vastglue`) as an array of 1D spectra in a convenient format (as a matrix of 1D spectra). If no arguments are provided, the number of rows and columns are determined by the periodicity of the display order. For example, if a block of 96 spectra, as is typical for a microtiter-plate, have been acquired using VAST automation, the spectra is plotted in a matrix 8 rows and 12 columns.

The default is to plot the spectra from 1 through `arraydim` (the number of spectra in the 2D data set). An optional argument (`plvast(##)`) allows one to specify that only spectra from 1 through `##` should be plotted.

Arguments: `display order` is optional and its default value is the glue order as listed in `glueorderarray`.

`number of columns plotted`. The default value of is deduced by examining the periodicity of the requested display order. The `number of columns plotted` can entered as the second argument or as the first argument if the default `display order` is used.

Examples: `plvast`
`plvast(12)`
`plvast('glue_file', 4)`

See also: *User Guide: Liquids NMR*

Related: `dsast2d` Display VAST data in a pseudo-2D format (M)
`dsvast` Display VAST data in a stacked 1D-NMR matrix (M)
`plvast2d` Plot VAST data in a pseudo-2D format (M)
`plate_glue` define a display order (U)

plvast2d Plot VAST data in a stacked pseudo-2D format (M)

Applicability: Systems with the VAST accessory.

Syntax: `plvast2d<(number)>`

Description: If an array of 1D spectra have been acquired (in particular if a block of 96 spectra has been acquired using VAST automation, especially in a microtiter-plate format) and if these spectra have been glued into a reconstructed 2D dataset (see `vastglue`), `plvast2d` will arrange and plot them (on the plotter) in a convenient pseudo-2D format (almost like an LC-NMR chromatogram). Well labels are not attached to the spectra and spectra are plotted with 12 spectra per row.

Arguments: `number` specifies that only spectra from 1 through `number` should be plotted. The default is to plot all the spectra (from 1 through `arraydim`).

See also: *User Guide: Liquids NMR*

Related: `dsast2d` Display VAST data in a pseudo-2D format (M)
`dsvast` Display VAST data in a stacked 1D-NMR matrix (M)
`plvast` Plot VAST data in a stacked 1D-NMR matrix (M)

plww **Plot spectra in whitewash mode (C)**

Syntax: `plww<(start, finish, step)><, 'all'>>`

Description: Plots one or more spectra in whitewash mode (after the first spectra, each spectra is blanked out in regions in which it is behind an earlier spectra).

Arguments: `start` is the index of the first spectra when plotting multiple spectra. It is also the index number of a particular trace to be plotted when plotting arrayed 1D spectra or 2D spectra. The default is to plot all spectra.

`finish` is the index of the last spectra when plotting multiple spectra.

`step` is the increment for the spectral index when plotting multiple spectra. The default is 1.

'all' is a keyword to plot all spectra in the array. This is the default.

See also: *User Guide: Liquids NMR*

Related: `dss` Display stacked spectra (C)
`dsww` Display spectra in whitewash mode (C)
`pl` Plot spectra (C)

pmode **Processing mode for 2D data (P)**

Description: Specifies the type of 2D spectral data that the 2D Fourier transform (FT) will yield. `pmode` is in the processing group.

Values: ' ' (null string, shown by two single quotes with no space in between) specifies a processing mode in which it is not possible to change either the f_2 or f_1 display mode after the 2D FT. If the f_2 display mode has been set to phased (`dmg= 'ph'`), each f_2 spectrum is phase rotated using the phase constants `rp` and `lp` prior to the FT along the second dimension. If the f_2 display mode has been set to power (`dmg= 'pwr'`) or absolute-value (`dmg= 'av'`), however, the f_2 spectrum is not processed any further after the first FT. The complex t_1 interferograms are handled in a similar manner. If the f_1 display mode has been set to phased (`dmg1= 'ph1'`), each f_1 spectrum is phased using the phase constants `rp1` and `lp1`. If the display mode has been set to power (`dmg1= 'pwr1'`) or to absolute value (`dmg1= 'av1'`), the appropriate magnitude calculation is performed, with the result being placed in the real part of the appropriate complex datum and a 0 being placed in the imaginary part. At the end of the 2D transform, the spectral data file `datdir/data` is reduced from complex data to real data ("VNMR REDUCE" display message).

'partial' specifies a processing mode in which it is not possible to change the f_2 display mode after the 2D FT. It is possible, however, to select between the three f_1 display modes without having to reprocess the 2D data. If the f_2 display mode has been set to phased (`dmg= 'ph'`), each f_2 spectrum is phase rotated using the phase constants `rp` and `lp` prior to FT along the second dimension. If the f_2 display mode is set to power (`dmg= 'pwr'`) or absolute value (`dmg= 'av'`), the f_2 spectrum is not processed any further after the first FT. Regardless of the requested f_1 display mode, no further processing is performed by `ft2d` on the f_1 spectra after the second FT. The calculations on

2D spectral data necessary to achieve the requested f_1 display mode are performed by `dcon` or `dconi`. If `pmode` does not exist, it is assigned a value of 'partial' internal to VNMR.

'full' specifies a processing mode in which it is possible to select between the three display modes for each dimension without having to reprocess the 2D data. Regardless of any requested display mode, no display mode processing is performed by `ft2d` on the f_2 spectra after the first or second FT. Display mode processing is performed exclusively by `dcon` or `dconi`.

The hypercomplex data structure for the 2D time domain data is

$$\{\text{Re}(t_1)\text{Re}(t_2), \text{Re}(t_1)\text{Im}(t_2), \text{Im}(t_1)\text{Re}(t_2), \text{Im}(t_1)\text{Im}(t_2)\}$$

and is experimentally composed by the pulse sequence generation arraying mechanism. The hypercomplex data structure for the t_1 interferograms is

$$\{\text{Re}(t_1)\text{Re}(F_2), \text{Re}(t_1)\text{Im}(F_2), \text{Im}(t_1)\text{Re}(F_2), \text{Im}(t_1)\text{Im}(F_2)\}$$

where Re represents the real part and Im represents the imaginary part. A hypercomplex FT along t_1 yields a hypercomplex 2D spectrum with the following data structure per hypercomplex point:

$$\{\text{Re}(F_1)\text{Re}(F_2), \text{Re}(F_1)\text{Im}(F_2), \text{Im}(F_1)\text{Re}(F_2), \text{Im}(F_1)\text{Im}(F_2)\}$$

Note that if `pmode` = 'full', the `ft2d` program will require an array index or coefficients for the construction of the t_1 interferograms.

See also: *User Guide: Liquids NMR*

Related:	<code>av</code>	Set abs. value mode in directly detected dimension (C)
	<code>av1</code>	Set abs. value mode in 1st indirectly detected dimension (C)
	<code>dcon</code>	Display noninteractive color intensity map (C)
	<code>dconi</code>	Interactive 2D data display (C)
	<code>dmg</code>	Data display mode in directly detected dimension (P)
	<code>dmg1</code>	Data display mode in 1st indirectly detected dimension (P)
	<code>ft1d</code>	Fourier transform along f_2 dimension (C)
	<code>ft2d</code>	Fourier transform 2D data (C)
	<code>ph</code>	Set phased mode in directly detected dimension (C)
	<code>ph1</code>	Set phased mode in indirectly detected dimension (C)
	<code>pwr</code>	Set power mode in directly detected dimension (C)
	<code>pwr1</code>	Set power mode in 1st indirectly detected dimension (C)
	<code>wft1d</code>	Weight and Fourier transform 2D data (C)
	<code>wft2d</code>	Weight and Fourier transform 2D data (C)

`poly0` **Display mean of the data in regression.inp file (M)**

Syntax: `poly0`

Description: Calculates and displays the mean of data in the file `regression.inp`.

See also: *VNMR User Programming*

Related:	<code>averag</code>	Calculate average and standard deviation of input (C)
	<code>expl</code>	Display exponential or polynomial curves (C)

`pos1, pos2, pos3` **Position of voxel center (P)**

Applicability: Systems with imaging capabilities.

Description: Define the center position, in cm, of the desired voxel for localized spectroscopy experiments.

See also: *User Guide: Imaging*

Related: **transfer** Move parameters to target experiment (M)
vox1, vox2, vox3 Voxel dimensions (P)

pp Decoupler pulse length (P)

Description: Sets the decoupler pulse length for use by pulse sequences such as DEPT, HET2DJ, and HETCOR.

See also: *Getting Started; User Guide: Liquids NMR*

Related: **AC1-AC9** Automatic calibration (M)
dept Set up parameters for DEPT pulse sequence (M)
dhp Decoupler high-power control with class C amplifier (P)
dpwr Power level for first decoupler with linear amplifier (P)
hetcor Set up parameters for HETCOR pulse sequence (M)
p1 First pulse width (P)
pplvl Pulse power level (P)
pw Pulse width (P)

ppa Plot a parameter list in plain English (M)

Syntax: `ppa< (x< , y>) >`

Description: Plots parameters in plain English (instead of in a table with parameter names and their values as plotted by the parameter **pap**).

Arguments: *x* controls the *x* offset, in mm, from the lower left of the plot to the starting position (upper left) of the parameter list. The default is a preset position on the page (upper left corner).

y controls the *y* offset, in mm, from the lower left of the plot to the starting position (upper left) of the parameter list. Default is a preset position on the page (upper left corner).

Examples: `ppa`
`ppa (10)`
`ppa (wcmx-80 , wc2max* .9)`

Alternate: Params button in the 1D Plotting Menu, or
 Params button in the 2D Plotting Menu.

See also: *Getting Started*

Related: **bpa** Plot boxed parameters (M)
hpa Plot parameters on special preprinted chart paper (C)
pap Plot out "all" parameters (C)
pltext Plot a text file (M)

ppcal Proton decoupler pulse calibration (M)

Syntax: `ppcal`

Description: Proton decoupler pulse calibration for DEPT, HETCOR, INEPT, etc.

See also: *Getting Started*

Related: **AC1-AC9** Automatic calibration (M)
d2pul Set up parameters for D2PUL pulse sequence (M)
dept Set up parameters for DEPT pulse sequence (M)
hetcor Set up parameters for HETCOR pulse sequence (M)
inept Set up parameters for INEPT pulse sequence (M)

ppe **Position of image center on 2D phase encode axis (P)**

Applicability: Systems with imaging capabilities.

Description: Position of image center on 2D phase encode axis, in cm.

See also: *User Guide: Imaging*

Related: **pro** Position of image center on the readout axis (P)

ppf **Plot peak frequencies over spectrum (C)**

Syntax: (1) `ppf(<'noll'><,'pos'><,noise_mult><,'top'>)>`
 (2) `ppf(<'noll'><,'pos'><,noise_mult><,'leader'><,length>)>`

Description: Plots peak frequencies, in units specified by the **axis** parameter, in the plotter device. Only those peaks greater than **th** high are selected. Two basic modes of label positioning are available: labels placed at the top, with long “leaders” extending down to the tops of the lines (syntax 1 using the 'top' keyword), or labels positioned just above each peak, with short leaders (syntax 2 using the 'leader' keyword). The default is short leaders.

Arguments: 'noll' is a keyword to plot frequencies using the last previous line listing.

'pos' is a keyword to plot positive peaks only ('noneg' is the same as 'pos').

noise_mult is a numerical value that determines the number of noise peaks plotted for broad, noisy peaks. The default is 3. A smaller value results in more peaks, a larger value results in fewer peaks, and a value of 0.0 results in a line listing containing all peaks above the threshold **th**. Negative values of noise_mult default to 3. The noise_mult argument is inactive when the 'noll' keyword is specified.

'top' is a keyword to plot labels at the top with long leaders. In this mode, the height of labels is varied by changing the parameter **wc2**.

'leader' is a keyword to plot labels positioned just above each peak with short leaders.

length specifies the leader length, in mm, if labels are positioned just above each peak. The default length is 20 mm.

Examples: `ppf('pos')`
`ppf('leader',30)`
`ppf('top','noll')`
`ppf('pos',0.0,'leader',30)`

Alternate: Peaks button in the 1D Plotting Menu.

See also: *Getting Started*

Related: **axis** Axis label for displays and plots (P)
dprf Display peak frequencies over spectrum (C)
dpir Display integral amplitudes below spectrum (C)
dpirn Display normalized integral amplitudes below spectrum (M)
pir Plot integral amplitudes below spectrum (C)
pirn Plot normalized integral amplitudes below spectrum (M)
th Threshold (P)

pph **Print pulse header (M)**

Syntax: `pph(file)`

Description: Prints out the shape file header (i.e., all lines starting with #).

Arguments: `file` is the name of the shape file, including the extension.

Examples: `pph('shgrad.GRD')`

See also: *User Guide: Liquids NMR*

Related: [Pbox](#) Pulse shaping software (U)

pp1v1 Proton pulse power level (P)

Applicability: *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* broadband systems with the diode switching version of RF Control board (refer to the description of the [attens](#) parameter to identify the types of RF Control boards) and systems with `amptype='a'`.

Description: Sets the pulse power level. `pp1v1` is only a relevant parameter in sequences that use decoupler pulses, such as DEPT, HET2DJ, and HETCOR.

Values: On *MERCURY-VX* and *MERCURY*, 0 to 63, in dB, steps of 1 dB. On *GEMINI 2000*: 0 to 63, in dB, steps of 0.5 dB.

When used with a 5-mm Gen. III switchable probe, typical value is 54 or 56.

See also: *Getting Started; User Guide: Liquids NMR*

Related: [amptype](#) Amplifier type (P)
[attens](#) Fast attenuators present (P)
[d2pul](#) Set up parameters for D2PUL pulse sequence (M)
[dept](#) Set up parameters for DEPT pulse sequence (M)
[het2dj](#) Set up parameters for HET2DJ pulse sequence (M)
[hetcor](#) Set up parameters for HETCOR pulse sequence (M)

ppmm Resolution on printers and plotters (P)

Description: An internal software parameter, selected automatically based on the plotter configuration, that contains the resolution in dots/mm on raster graphics printers. On pen plotters, `ppmm` contains the resolution of points drawn. On PostScript printers, `ppmm` adjusts linewidths.

pprofile Plot pulse excitation profile (M)

Syntax: `pprofile<(axisflag<,profile<,shapefile>>>>`

Description: Plots the X, Y and Z excitation (inversion) profile for a pulse shape that has been generated with the Pbox software. If shape names is not provided, the last simulation data stored in the `shapelib/pbox.sim` file are plotted.

Arguments: The `axisflag` and `profile` arguments can be given in any order.

`axisflag` is 'y' to display the full spectrum and a frequency scale, or 'n' to suppress the scale and spectrum. The default is 'n'.

`profile` is a character string identifying the desired profile. 'xyz' selects X, Y, and Z (inversion) profiles; 'xy' selects only the excitation (transverse) profiles; 'x' selects only the X transverse excitation profile; and 'z' selects only the inversion profile. The default is 'xyz'.

`shapefile` is the name of a *.RF or *.DEC file, including the extension.

Examples: `pprofile`
`pprofile('y','x')`
`pprofile('xy','n','softpls.RF')`

See also: *User Guide: Liquids NMR*

Related: **dprofile** Display pulse excitation profile (M)
Pbox Pulse shaping software (U)

pps Plot pulse sequence (C)

Syntax: `pps<(file<,x,y,width,height)>>`

Description: Plots pulse sequences. The plotted picture consists of three to five parts. At the top is the transmitter pulse sequence. Below that is the decoupler pulse sequence. Next is the second decoupler pulse sequence or gradients, depending on the program. At the bottom is the status.

The parameter of each pulse is plotted if its length is less than 30 letters. The value of each pulse is also plotted. If its value is less than zero, a question mark “?” is plotted. The time units are displayed as letters (s, m, or u). The height of pulses are plotted according to their power level.

Arguments: `file` specifies the pulse sequence to be plotted. The default is `seqfil`.

`x,y` specifies the start of the plotting position with respect to the lower-left corner of the plotter.

`width,height` are in proportion to `wcmax` and `wc2max`.

Examples: `pps`
`pps('s2pul')`
`pps(3,50)`

See also: *Getting Started*

Related: **dps** Display pulse sequence (C)
seqfil Pulse sequence name (P)
wcmax Maximum width of chart (P)
wc2max Maximum width of chart in second direction (P)

presat Set up parameters for PRESAT pulse sequence (M)

Applicability: This sequence is not supplied with *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Syntax: `presat`

Description: Sets up a 1D water suppression experiment.

See also: *User Guide: Liquids NMR*

presig Preamplicifier signal level selection (P)

Applicability: *UNITYINOVA* and *UNITYplus* imaging systems, or *UNITYINOVA* and *UNITYplus* spectrometers with selectable large-signal mode preamplifiers.

Description: Allows the user to select either high or low signal handling on preamplifiers that support this capability:

- *UNITYINOVA* and *UNITYplus* imaging systems support this capability by using attenuation and a current increase. This allows larger signals and results in a lower overall signal level.
- *UNITYINOVA* and *UNITYplus* spectrometers with selectable large-signal mode preamplifiers support this capability by allowing a current increase in the preamplifier. This allows larger signals so that the overall signal level is slightly higher.

Using `presig` to control the hardware depends on the Magnet Leg Driver Board Configuration ID being set to 16 for imaging systems, or to 1 for

UNITY *INOVA* and UNITY *plus* spectrometers with the selectable large-signal mode preamplifier.

Values: 'h' signifies high-signal mode at the preamplifier.

'l' signifies low-signal mode at the preamplifier. The default is this mode at the preamplifier if the hardware is present

'n' signifies not used.

See also: *User Guide: Imaging*

Related: [gain](#) Receiver gain (P)

prevpl Display the previous 3D plane (M)

Applicability: All systems; however, although `prevpl` is available on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*, such systems can only process 3D data and cannot acquire 3D data.

Syntax: `prevpl`

Description: Displays 2D color map of the previous 3D plane in the set of planes defined by the parameters `plane` and `path3d`. For example, if `dplane(40)` has just been executed, `prevpl` results in the display of 3D plane 39 of that set. (If `prevpl` immediately follows the command `dproj`, an error results because there is no 3D plane whose number is -1.) `prevpl` is more efficient than `dplane` or `dproj` because the 3D parameter set (`procp3d`) is not loaded into VNMR. It is assumed to have already been loaded by, for example, `dplane` or `dproj`.

See also: *User Guide: Liquids NMR*

Related: [dplane](#) Display a 3D plane (M)
[dproj](#) Display a 3D plane projection (M)
[dsplanes](#) Display a series of 3D planes (M)
[getplane](#) Extract planes from a 3D spectral data set (M)
[nextpl](#) Display the next 3D plane (M)
[path3d](#) Path to currently displayed 2D planes from a 3D data set (P)
[plane](#) Currently displayed 3D plane type (P)
[plplanes](#) Plot a series of 3D planes (M)

printer Printer device (P)

Description: Selects the printer in use on the system.

Values: A string with entries such as 'ThinkJet_96', 'LaserJet_300', 'jim', 'varian1', and 'Laser1'.

See also: *Getting Started*

Related: [showplotter](#) Show list of currently defined plotters and printers (M)

printoff Stop sending text to printer and start print operation (C)

Syntax: `printoff<('clear' | file)>`

Description: Stops redirection of output to printer caused by the `printon` command and starts the print operation. **The command `printoff` must be entered to obtain output on the printer.** Actual printing is controlled by the `vnmrprint` script in the `bin` subdirectory of the VNMR system directory. `printoff` can also clear the data in the current print file or save data to a specified file name (i.e., print or plot to a file).

Arguments: 'clear' is a keyword to clear the print file made so far.

file specifies the name of a file to save the printout. If the file already exists, it is overwritten.

Examples: `printoff`
`printoff('clear')`
`printoff('vnmrsys/papers/peaks.list')`

See also: *Getting Started*

Related: `printon` Direct text output to printer (C)
`vnmrprint` Print text files (U)

printon Direct text output to printer (C)

Syntax: `printon`

Description: Sends information to the printer that is normally displayed in the text window. After using `printon`, output from commands that use the text window, such as `dg` and `cat`, is sent to the printer and does not appear on the VNMR screen. The value of the parameter `printer` is used to select which printer is used.

See also: *Getting Started*

Related: `cat` Output one or more files to output text window (C)
`dg` Display group of acquisition/processing parameters (C)
`printer` Printer device (P)
`printoff` Stop sending text to printer and start print operation (C)

pro Position of image center on the readout axis (P)

Applicability: Systems with imaging capabilities.

Description: Position of image center on readout axis, in cm.

See also: *User Guide: Imaging*

Related: `ppe` Position of image center on 2D phase encode axis (P)

probe Probe type (P)

Description: Contains a string with the name of the probe currently in the magnet. This parameter is set automatically when the `addprobe` macro is entered. The `getparam` and `setparams` macros use `probe` to retrieve and write parameters into the current probe file.

See also: *Getting Started*

Related: `addnucleus` Add new nucleus to existing probe file (M)
`addprobe` Create new probe directory and probe file (M)
`getparam` Receive parameter from probe file (M)
`setparams` Write parameter to current probe file (M)

Probe_edit Edit probe for specific nucleus (U)

Syntax: (UNIX) `Probe_edit probe nucleus`

Description: Opens a dialog box showing all the parameters related to a specific nucleus from the probe table.

Arguments: `probe` is the name of the probe.
`nucleus` is the specified nucleus from the probe table.

Examples: `Probe_edit 5mmSW H1`

Related: `probe_edit` Edit probe for specific nucleus (M)

probe_edit **Edit probe for specific nucleus (M)**

Syntax: `probe_edit (probe, nucleus)`

Description: Opens a dialog box showing all the parameters related to a specific nucleus from the probe table.

Arguments: `probe` is the name of the probe.
`nucleus` is the specified nucleus from the probe table.

Examples: `probe_edit ('5mmSW', 'H1')`
`probe_edit (probe, tn)`

Related: [Probe_edit](#) Edit probe for a specific nucleus (U)

probe_protection **Probe protection control (P)**

Description: Controls the power check for probe protection.

See also: *Getting Started*

proc **Type of processing on np FID (P)**

Description: Specifies the type of data processing to be performed upon the `np` (t_2) FID. Similarly, parameters `proc1` and `proc2` specify the type of data processing on the `ni` (t_1) and `ni2` interferograms, respectively.

All Varian data must be processed along `np` with a complex Fourier transform (FT). Sequentially sampled Bruker data (the usual case) must be processed along this dimension with a real FT, while simultaneously sampled Bruker data must be processed with a complex FT.

Pure absorptive 2D data collected by the States-Haberhorn (hypercomplex) method must be processed along `ni` or `ni2` with a complex FT.

Pure absorptive 2D data collected by the TPPI method on a Varian spectrometer can be processed in one of two ways, depending upon how the data was collected:

<code>phase=3</code>	Complex FT, i.e., <code>proc1='ft'</code> (standard way)
<code>phase=1,4</code>	Real FT, i.e., <code>proc1='rft'</code> (new way)
<code>phase2=3</code>	Complex FT, i.e., <code>proc2='ft'</code>
<code>phase2=1,4</code>	Real FT, i.e., <code>proc2='rft'</code>

Pure absorptive 2D data collected by TPPI method on a Bruker spectrometer must be processed along `ni` with a real FT (i.e., `proc1='rft'`).

Values: `'ft'` specifies complex FT data processing.

`'rft'` specifies real FT data processing.

`'lp'` specifies linear prediction processing on complex data. If `'lp'` is selected, additional parameters must be set to fully define how the time-domain data is to be processed; see the description of the [addpar](#) command.

See also: *Getting Started*

Related: [addpar](#) Add selected parameters to the current experiment (M)
[ni](#) Number of increments in 1st indirectly detected dimension (P)
[np](#) Number of data points (P)
[parlp](#) Create parameters for linear prediction (C)
[phase](#) Phase selection (P)
[phase2](#) Phase selection for 3D acquisition (P)

`proc1` Type of processing on `ni` interferogram (P)
`proc2` Type of processing on `ni2` interferogram (P)

`proc1` Type of processing on `ni` interferogram (P)

Description: Specifies the type of data processing to be performed upon the `ni` (`t1`) interferogram (2D). Refer to the description of `proc` for further information.

Values: `'ft'` specifies complex Fourier transform (FT) data processing.
`'rft'` specifies real FT data processing.
`'lp'` specifies linear prediction processing on complex data. If `'lp'` is selected, additional parameters must be set to fully define how the time-domain data is to be processed; see the description of the `addpar` command.

See also: *User Guide: Liquids NMR*

Related: `addpar` Add selected parameters to the current experiment (M)
`ni` Number of increments in 1st indirectly detected dimension (P)
`proc` Type of processing on `np` FID (P)

`proclD` Processing macro for simple (non-arrayed) 1D spectra (M)

Syntax: `proclD`

Description: A generic macro for processing non-arrayed 1D spectra using a set of standard macros. `proclD` is called by the `procplot` macro, but can also be used directly. `proclD` first tries to find a macro of the form `{tn}p` with the name of the observe nucleus in lower case (e.g., `h1p`, `c13p`). If such a macro exists, it is called. If such a nucleus-specific macro is not found in the command path, minimal 1D processing is performed (the intent is to provide a well-processed spectrum in most cases): Fourier transformation (using pre-set weighting functions), automatic phasing (`aphx` macro), automatic integration (`integrate` macro), vertical scale adjustment (`vsadj` macro), avoiding excessive noise (`noislm` macro), and threshold adjustment (`thadj` macro). `proclD` does not work with arrayed 1D spectra: use `deptproc` (for DEPT-type spectra) or `procarray` (for all other arrayed 1D data).

See also: *Getting Started*

Related: `aphx` Perform optimized automatic phasing (M)
`c13p` Process 1D carbon spectra (M)
`deptproc` Process arrayed dept type spectra (M)
`h1p` Process 1D proton spectra (M)
`integrate` Automatically integrate 1D spectrum (M)
`noislm` Avoids excessive noise (M)
`procarray` Process arrayed 1D spectra (M)
`procplot` Automatically process FIDs (M)
`thadj` Adjust threshold (M)
`vsadj` Adjust vertical scale (M)

`proc2` Type of processing on `ni2` interferogram (P)

Description: Specifies the type of data processing to be performed upon the `ni2` interferogram (3D). Refer to the description of `proc` for further information.

Values: `'ft'` specifies complex Fourier transform (FT) data processing.
`'rft'` specifies real FT data processing.

'lp' specifies linear prediction processing on complex data. If 'lp' is selected, additional parameters must be set to fully define how the time-domain data is to be processed; see the description of the `addpar` command.

See also: *User Guide: Liquids NMR*

Related: `addpar` Add selected parameters to the current experiment (M)
`ni2` Number of increments in 2nd indirectly detected dimension (P)
`proc` Type of processing on np FID (P)

`proc2d` Process 2D spectra (M)

Syntax: `proc2d`

Description: A general 2D processing macro that tries to do the appropriate processing for as many types of 2D experiments as possible. It uses `wft2da` for phase-sensitive spectra, `wft2d` for absolute-value 2D spectra, `wft2d('ptype')` for HOM2DJ and COSYPS (absolute value). Symmetric homonuclear correlation spectra (`fn=fn1, sw=sw1`) in absolute-value mode is symmetrized using `foldt`. The resulting spectrum is then normalized (adjustment of `vs` and `th`) using `nm2d` and displayed (if not in background mode). `proc2d` is called as part of the `procplot` macro, but can also be used directly by the user.

See also: *Getting Started*

Related: `fn` Fourier number in the directly detected dimension (P)
`fn1` Fourier number in 1st indirectly detected dimension (P)
`foldt` Fold COSY-like spectrum along diagonal axis (C)
`nm2d` Normalize intensity of 2D spectrum (M)
`procplot` Automatically process FIDs (M)
`sw` Spectral width in the directly detected dimension (P)
`sw1` Spectral width in the 1st indirectly detected dimension (P)
`th` Threshold (P)
`vs` Vertical scale (P)
`wft2d` Weight and Fourier transform 2D data (C)
`wft2da` Weight and Fourier transform for pure absorption 2D data (M)

`proccarray` Process arrayed 1D spectra (M)

Syntax: `proccarray`

Description: A generic macro for processing arrayed 1D data. It is called within the `procplot` macro, but can also be called directly. It transforms all traces, phase the trace with the largest signal, scale the traces appropriately, and set up the display parameters such that the data can be plotted directly. The plotting is done in a separate macro `plarray` that is also called in the `procplot` macro.

For the display setup, `proccarray` distinguishes between arrays with 6 or less elements, which are stacked vertically (no horizontal offset), and spectra with greater than 6 elements, which are stacked horizontally by default, unless there are too many lines, in which case a diagonally stacked display is chosen.

Horizontal stacking is mostly adequate for pulse and power calibrations, where there are usually only a few lines. Diagonally stacked displays and plots are frequently chosen for T_1 and T_2 experiments on entire spectra, often with many lines. The automatic stacking mode can be overridden by creating and setting a string parameter `stackmode` in the startup macro, or before calling `procplot` or `proccarray`. Possible values for `stackmode` are 'horizontal', 'vertical', and 'diagonal'. DEPT-type spectra can, in principle, be also processed with `proccarray` but, of course, no DEPT editing occurs.

See also: *Getting Started*

Related:	<code>deptproc</code>	Process arrayed dept type spectra (M)
	<code>plarray</code>	Plot arrayed 1D spectra (M)
	<code>procl1d</code>	Processing macro for simple (non-arrayed) 1D spectra (M)
	<code>procplot</code>	Automatically process FIDs (M)
	<code>stack</code>	Set stacking control parameter (M)
	<code>stackmode</code>	Stack control for processing arrayed 1D spectra (P)

process **Generic automatic processing (M)**

Syntax: `process`

Description: Processes a wide range of data types. It selects a macro depending on the type of data. For simple 1D spectra, `process` looks for a macro of form `{tn}p` with the observe nucleus in lower case (e.g., `h1p`, `c13p`, `f19p`). If no such macro is found, `process` calls `procl1d`, a generic processing macro for 1D spectra. For DEPT type data, `deptproc` is called. For other arrays of 1D spectra, `procarray` is called. For 2D spectra, `proc2d` is called. `process` by itself is called within the `procplot` macro.

See also: *Getting Started*

Related:	<code>c13p</code>	Processing of 1D carbon spectra (M)
	<code>deptproc</code>	Process array of DEPT spectra (M)
	<code>f19p</code>	Processing of 1D fluorine spectra (M)
	<code>h1p</code>	Processing of 1D proton spectra (M)
	<code>procl1d</code>	Automatically process non-arrayed 1D fids (M)
	<code>proc2d</code>	Process 2D spectra (M)
	<code>procarray</code>	Process arrayed 1D spectra (M)
	<code>procplot</code>	Automatically process FIDs (M)
	<code>tn</code>	Nucleus for observe transmitter (P)

procplot **Automatically process FIDs (M)**

Syntax: `procplot<(pltmod_value)>`

Description: Universal FID processing macro called usually with `wexp='procplot'` by automatic acquisition macros such as `h1`, `c13`, `hcapt`, and `hcosy`. The purpose of `procplot` is not the data processing itself, but rather the selection of the appropriate processing macro for a given data set.

First, `procplot` calls a macro `process` that calculates spectra; that macro by itself then selects an appropriate processing macro, like `procl1d` for non-arrayed 1D spectra. Depending whether the parameter `pltmod` is set to 'none' or not, `procplot` then calls `plot`, a universal plotting macro. The setting of the parameter `pltmod` can be temporarily overridden by specifying an alternative value as argument to `procplot`.

One of the concepts behind `procplot` is that the user should never have to modify any processing macro for customizing the processing or the output of automatic experiments or processing; this outcome can happen by selecting a parameter in the calling macro or before calling `procplot`.

Arguments: `pltmod_value` is an alternate value for the parameter `pltmod` that is only used for the current call. The values 'none' and 'off' suppress plotting. The range of possible (active) values for `pltmod_value` depends on the plotting macros. Often, the parameter `pltmod` has no effect other than turning on or off plotting. Note that if only the calculation of a spectrum is desired, it is usually easier to call the `process` macro.

Examples: `procplot`
`procplot('none')`

See also: *Getting Started; User Guide: Liquids NMR*

Related: `deptproc` Process arrayed dept type spectra (M)
`plot` Automatically plot spectra (M)
`pltmod` Determine plot mode (P)
`procl1d` Processing macro for simple (non-arrayed) 1D spectra (M)
`proc2d` Process 2D spectra (M)
`procarray` Process arrayed 1D spectra (M)
`process` Automatically calculate spectra (M)

profile Set up pulse sequence for gradient calibration (M)

Applicability: Systems with the pulsed field gradients (PFG) module.

Syntax: `profile`

Description: Performs an rf and gradient echo sequence that gives a high quality profile of the sample. This sequence is used with the macro `setgcal` to provide gradient strength calibration. The `gradaxis` parameter is used by `profile` to select the x, y, or z gradient axis.

See also: *Perform a Pulsed Field Gradient Module Installation; Pulsed Field Gradient Modules Installation; VNMR User Programming*

Related: `gcal` Gradient calibration constant (P)
`gradaxis` Gradient axis (P)
`setgcal` Calibrate gradient strength from measured data (M)

proj Project 2D data (C)

Syntax: `proj(exp_number<, 'sum'><, start<, width>>)`

Description: Projects 2D data onto the axis parallel to the screen x-axis, which can be f_1 or f_2 , depending upon the parameter `trace`. Two projections are available:

- *Summing projection.* The data at each frequency are summed and the result becomes the projection.
- *Skyline projection.* The data are searched and the maximum intensity at any given frequency becomes the intensity in the projection (similar to looking at the skyline of a city where only the largest building along any given line of sight is visible).

Phase-sensitive data can be projected, but the resulting projection can only be displayed in an absolute-value mode

Arguments: `exp_number` is the number of the experiment, from 1 through 9, in which the resulting spectrum is stored.

`'sum'` is a keyword to use the summing projection. The default is skyline.

`start` defines the starting trace, in Hz. The default is to project all data.

`width` defines the width of the traces, in Hz, to be projected. The default is to project all data. If `width` is supplied as zero, a single trace corresponding to the `start` frequency will be stored.

Examples: `proj(3)`
`proj(5, 'sum')`
`proj(4, 3*sfrq, 6*sfrq)`

See also: *User Guide: Liquids NMR*

Related: `trace` Select mode for 2D data display (P)

PROTON Set up parameters for proton spectrum (M)Applicability: *GLIDE*

Syntax: PROTON

Description: Internal macro that sets up parameters for a proton spectrum in *GLIDE*. PROTON is used only when proton is selected in a carbon experiment.**prune Prune extra parameters from current tree (C)**Syntax: `prune(file)`Description: Destroys parameters in the current parameter tree that are not also defined in the supplied parameter file. `prune` is used to remove leftover parameters from previous experimental setups. Recalling a new parameter set into an experiment has a similar effect and, in general, `prune` is not required.Arguments: `file` is the path of a parameter file.Examples: `prune(systemdir+' /parlib/cosyps.par/procpar ')`
`prune(' /vnmr/par400/stdpar/H1.par/procpar ')`
`prune(userdir+' /exp3/curpar ')`See also: *VNMR User Programming*

Related:	<code>create</code>	Create new parameter in a parameter tree (C)
	<code>destroy</code>	Destroy a parameter (C)
	<code>display</code>	Display parameters and their attributes (C)
	<code>fread</code>	Read parameters from file and load them into a tree (C)
	<code>fsave</code>	Save parameters from a tree to a file (C)

pscale Plot scale below spectrum or FID (C)Syntax: `pscale(<'fid'><,axis><,vert_start><,plot_start><,pen>>`

Description: Plots a scale under a spectrum or FID.

Arguments: `'fid'` is a keyword to plot a FID scale; if used, it must be the first argument. `axis` is a letter to be used to label the axis. For a spectrum scale, if `'p'`, `'h'`, `'k'`, `'c'`, `'m'`, `'u'`, etc. is supplied, the letter within the single quotes is used instead of the current value of the `axis`. For an FID scale, if `'s'`, `'m'`, or `'u'` is supplied, it is used instead of the current value of the `axisf`.`vert_start` is a real number that defines the vertical position where the scale is plotted. The default is 5 mm below the current value of the parameter `vp`.`plot_start` is a real number that modifies the start of a plot. For example, if the plot is from 347 to 447 Hz, but a scale of 0 to 100 Hz is desired, `plot_start` would be 0. `pen` is a pen number: `'pen1'`, `'pen2'`, `'pen3'`, etc. The default is `'pen1'`.Examples: `pscale`
`pscale(20)`
`pscale('h',0,'pen2')`
`pscale('fid','m')`
`pscale('h',vp-10,0)`

Alternate: Scale button in the 1D Plotting Menu.

See also: *Getting Started*

Related:	<code>axis</code>	Axis label for displays and plots (P)
	<code>axisf</code>	Axis label for FID displays and plots (P)

`dscale` Display scale below spectrum or FID (C)
`vp` Vertical position of spectrum (P)

pseudo Set default parameters for pseudo-echo weighting (M)

Syntax: `pseudo<(C1,C2,C3,C4)>`

Description: Generates an initial guess at good weighting parameters for absolute-value 2D experiments. To generate modified guesses, four coefficients are allowed to set the values of the weighting functions.

Arguments: C1 sets `lb=-0.318/(C1*at)`. The default value of C1 is 0.0625.

C2 sets `gf=C2*at`. The default value of C2 is 0.25.

C3 sets `lb1=-0.318/(C3*(ni/sw1))` but is used with 2D experiments only. The default value of C3 is 0.0625.

C4 sets `gf1=C4*(ni/sw1)` but is used with 2D experiments only. The default value of C4 is 0.25.

Examples: `pseudo`
`pseudo(.1,.4,.2,.5)`

Alternate: Pseudo button in the 2D Processing Parameter Setup Menu.

See also: *User Guide: Liquids NMR*

Related: `sinebell` Select default parameters for sinebell weighting (M)

psg Display pulse sequence generation errors (M)

Syntax: `psg`

Description: Helps identify the problem if, after entering `go` or `su`, etc., the message is returned that pulse sequence generation (PSG) aborted abnormally. Any parameters that are not found are listed. This information is stored in the user's directory (`vnmr/sys`) in a text file named `psg.error`. If the message "Maximum communication retries exceeded, Experiment unable to be sent" is displayed, a program communications problem is indicated. Consult the system operator for assistance.

See also: *VNMR User Programming*

Related: `go` Submit experiment to acquisition (C)
`su` Submit a setup experiment to acquisition (M)

psggen Compile a user PSG object library (M,U)

Syntax: `psggen`

Description: A user PSG (pulse sequence generation) kit is supplied that allows editing low-level pulse sequence code. `psggen` compiles these edits so that subsequent pulse sequence generation with the `seqgen` command uses the customized pulse sequence source.

See also: *VNMR User Programming*

Related: `seqgen` Initiate compilation of user's pulse sequence (M,U)

psgset Set up parameters for various pulse sequences (M)

Syntax: `psgset(file,par1,par2,...,parN)`

Description: Sets up parameters for various pulse sequences using information in a `parlib` file. Rather than returning the entire parameter file, `psgset` returns the

parameters listed. `psgset`, in general, is never entered from the keyboard but is used as part of experiment setup macros.

Arguments: `file` is the file from the user or system `parlib` that provides information on setting up the parameters listed. The parameters `seqfil` and `pslabel` are set to the supplied file name.

`par1, par2, . . . , pN` are 1 to 11 parameters to be returned from `parlib`.

Examples: `psgset('cosy', 'dg', 'ap', 'ss', 'dl', 'axis', 'phase')`

See also: *VNMR User Programming*

Related: `pslabel` Pulse sequence label (P)
`seqfil` Pulse sequence name (P)

psgupdateon Enable update of acquisition parameters (C)

Syntax: `psgupdateon`

Description: Permits the interactive updating of acquisition parameters.

See also: *SpinCAD*

Related: `psgupdateoff` Prevent update of acquisition parameters (C)
`updtparam` Update specified acquisition parameters (C)

psgupdateoff Prevent update of acquisition parameters (C)

Syntax: `psgupdateoff`

Description: Prevents the interactive updating of acquisition parameters.

See also: *SpinCAD*

Related: `psgupdateon` Enable update of acquisition parameters (C)
`updtparam` Update specified acquisition parameters (C)

pshape Plot pulse shape or modulation pattern (M)

Syntax: `pshape<(pattern.ext)>`

Description: Plots the real (X) and imaginary (Y) components of a shaped pulse. Any type of waveform (.RF, .DEC or .GRD) can be plotted.

Arguments: `pattern` is the name of a shape or pattern file specified by an absolute file name, relative file name, or a simple pattern file name. `ext` is a file name extension that specifies the file type. In the case of a simple file name, `dshape` searches for the file in the local directory, then in the user's `shapelib`, and finally in the directory `/vnmr/shapelib`. If `pattern.ext` is not given, `pshape` displays the last created waveform stored in the `pbox.fid` file.

Examples: `pshape`
`pshape('my_shape.DEC')`

See also: *User Guide: Liquids NMR*

Related: `dshape` Display the last created pulse shape (M)
`Pbox` Pulse shaping software (U)

pshapenf Plot the last created pulse shape (M)

Syntax: `pshapenf`

Description: Plots real (X) and imaginary (Y) components of the last created shaped pulse.

See also: *User Guide: Liquids NMR*

Related: `dshape` Display the last created pulse shape (M)
: `pbox` Pulse shaping software (U)

psi Euler angle psi from magnet frame (P)

Applicability: Systems with imaging capabilities.

Description: Euler angle psi from magnet frame.

Values: -90 to +90, in degrees

See also: *User Guide: Imaging*

Related: `phi` Euler angle phi from magnet frame (P)
`theta` Euler angle theta from magnet frame (P)

pslabel Pulse sequence label (P)

Description: Contains the text to be displayed in the `Seq:` field on the top line of the screen. This string may be different from the pulse sequence name selected with `seqfil`. However, the string in `seqfil` is the name of the pulse sequence searched for when an experiment is started. Generally `seqfil=pslabel`, and when `seqfil` is set, the system sets `pslabel` to the same string.

See also: *Getting Started*

Related: `seqfil` Pulse sequence name (P)

pss Slice position (P)

Applicability: Systems with imaging capabilities.

Description: Position of slice, in cm.

See also: *User Guide: Imaging*

Related: `plan` Display menu for planning a target scan (M)

ptext Print out a text file (M)

Syntax: `ptext(file)`

Description: Prints out a text file.

Arguments: `file` is the name of the text file.

Examples: `ptext('/vnmr/maclib/ptext')`
`ptext(curexp+' /dept.out')`

See also: *Getting Started*

Related: `curexp` Current experiment directory (P)
`dtext` Display a text file in the graphics window (C)
`lookup` Look up words and lines from a text file (C)
`pltext` Plot a text file (C)
`text` Display text or set new text for current experiment (C)
`textvi` Edit text file of current experiment (M)
`vi` Edit text file with `vi` text editor (C)

ptspec3d **Region-selective 3D processing (P)**

Applicability: All systems; however, although `ptspec3d` is available on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*, such systems can only process 3D data and cannot acquire 3D data.

Description: Sets whether region-selective 3D processing occurs. If `ptspec3d` does not exist, it is created by the macro `par3d`. `ptspec3d` is functional at this time only for the f_3 dimension. If `ptspec3d='ynn'`, only the currently displayed region of f_3 is retained as non-zero values after the f_3 transform in the 3D FT. A larger f_3 region may be kept to ensure that the number of hypercomplex f_3 points is a power of 2; but that portion of the f_3 spectrum that is retained outside of the currently displayed region contains only zeroes. This 3D utility can reduce the fully transformed 3D data size by factors of 2 to 4, especially in some of the triple resonance experiments.

Values: A three-character string such as 'nnn', 'nny', 'nyn', etc. The default is 'nnn'. The first character refers to the f_3 dimension (`sw`, `np`, `fn`); the second character, to the f_1 dimension (`sw1`, `ni`, `fn1`); and the third character, to the f_2 dimension (`sw2`, `ni2`, `fn2`). Each character may take one of two values: 'n' for no region-selective processing in the relevant dimension, or 'y' for region-selective processing in the relevant dimension.

See also: *User Guide: Liquids NMR*

Related:	<code>fiddc3d</code>	3D time-domain dc correction (P)
	<code>fn</code>	Fourier number in directly detected dimension (P)
	<code>fn1</code>	Fourier number in 1st indirectly detected dimension (P)
	<code>fn2</code>	Fourier number in 2nd indirectly detected dimension (P)
	<code>ft3d</code>	Perform a 3D Fourier transform (M)
	<code>ni</code>	Number of increments in 1st indirectly detected dimension (P)
	<code>ni2</code>	Number of increments in 2nd indirectly detected dimension (P)
	<code>np</code>	Number of data points (P)
	<code>ntype3d</code>	N-type peak selection in f_1 or f_2 (P)
	<code>par3d</code>	Create 3D acquisition, processing, display parameters (C)
	<code>specdc3d</code>	3D spectral dc correction (P)
	<code>sw</code>	Spectral width in directly detected dimension (P)
	<code>sw1</code>	Spectral width in 1st indirectly detected dimension (P)
	<code>sw2</code>	Spectral width in 2nd indirectly detected dimension (P)

ptsva1 **PTS frequency synthesizer value (P)**

Description: Configuration parameter for the frequency of the PTS synthesizer on each channel. Every broadband system is equipped with a PTS frequency synthesizer as part of broadband frequency generation. The frequency of the unit is marked on its front panel. The value is set for each channel using the Synthesizer label in the CONFIG window (opened from `config`).

Values: On *MERCURY-VX* and *MERCURY*, `ptsva1` has no meaning. On *GEMINI 2000* broadband, the value is implicitly set (using `config`) to 160 or 250. On systems other than *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*, 0 (Not Present choice in CONFIG window); 160, 200, 250, 320, 500, 620, 1000 (PTS 160, PTS 200, PTS 250, PTS 320, PTS 500, PTS 620, PTS 1000 choices in CONFIG window, respectively).

See also: *VNMR and Solaris Software Installation*.

Related:	<code>config</code>	Display current configuration and possibly change it (M)
	<code>latch</code>	Frequency synthesizer latching (P)
	<code>overrange</code>	Frequency synthesizer overrange (P)

pulsecal Update and display pulse calibration data file (M)

Applicability: Systems with the imaging capabilities.

Syntax: (1) `pulsecal<(name, pattern, length, flip, power)>`
 (2) `pulsecal(name, 'remove')`

Description: Creates and maintains a database file of rf coil calibration data. This database is accessed by the SEQD command `setflip` in order to automatically enter power level settings for various types of rf pulses.

If entered without arguments, `pulsecal` displays the current contents of the database file. Using `pulsecal` with syntax 1 creates an entry in the file `userdir+ '/pulsecal'`. Using syntax 2 removes the entire line associated with the calibration name.

Arguments: `name` is the name of the rf coil or calibration.

`pattern` is the rf pattern used in the calibration experiment.

`length` is the length of the rf pulse, in μs , used for calibration.

`flip` is the flip angle calibrated, in degrees.

`power` is the calibrated power level, in attenuator units.

'remove' is a keyword to remove the line associated with the calibration name.

Examples: `pulsecal`
`pulsecal('small_coil', 'sinc', 5000, 180, 88)`
`pulsecal('small_coil', 'remove')`

See also: *User Guide: Imaging*

Related: `setflip` Set rf power levels for desired flip angle (M)
`userdir` VNMR user directory (P)

pulseinfo Shaped pulse information for calibration (M)

Syntax: `pulseinfo<(shape, pulse_width<, reference_power>)>`
`:width, power`

Description: Returns or prints a table with the bandwidth and predicted pulse power settings for a given pulse shape. No parameter settings are changed. The necessary data is contained in the file `shapeinfo` in the VNMR system `shapelib` subdirectory.

Arguments: `shape` is the name of the pulse shape. The default is the system interactively prompts the operator for the name of the shape and the duration of the pulse and then prints a table containing the bandwidth of that pulse and the predicted pulse power settings.

`pulse_width` is the duration of the pulse, in μs .

`reference_power` is a value, in dB, for power calculations. The default is 55. This value replaces the assumption used for power calculation that `pw90` is set for a `tpwr` of 55.

`width` returns the bandwidth of that pulse, in Hz.

`power` returns the predicted 90° pulse power settings.

Examples: `pulseinfo('gauss', 1000):bw, pwr`

See also: *VNMR User Programming*

Related: `bandinfo` Shaped pulse information for calibration (M)
`pw90` 90° pulse width (P)
`tpwr` Observe transmitter power level with linear amplifiers (P)

pulsetool **RF pulse shape analysis (U)**

Syntax: `pulsetool <-shape filepath>`

Description: Enables examination of shaped rf pulses. It is started from a UNIX window.

Arguments: The optional `-shape filepath` specifies the name of an rf pulse template file that is displayed when `pulsetool` is started.

Examples: `pulsetool`
`pulsetool -shape /vnmr/shapelib/sinc.RF`

See also: *User Guide: Liquids NMR*

purge **Remove macro from memory (C)**

Syntax: `purge<(file)>`

Description: Removes one or more macros from memory, freeing extra memory space.

Arguments: `file` is the name of a macro file to be removed from memory. The default is to remove all macros that have been loaded into memory.

CAUTION: The `purge` command with no arguments should never be called from a macro. The `purge` command with an argument should never be called by the macro being purged.

Examples: `purge`
`purge('_sw')`

See also: *VNMR User Programming*

Related: `macrold` Load a macro into memory (C)

puttxt **Put text file into VNMR data file (C)**

Syntax: `puttxt(file)`

Description: Copies text from current experiment into a data file.

Arguments: `file` is the name of a VNMR data file (i.e., a directory with a `.fid` or `.par` suffix). Do not include the suffix in the name provided to `file`.

Examples: `puttxt('mydata')`

See also: *Getting Started*

Related: `gettxt` Get text file from another file (C)

putwave **Write a wave into Pbox.inp file (M)**

Syntax: `putwave(sh, bw, pw, ofs, st, ph, fla, trev, d1, d2, d0)`

Description: Sets up a single excitation band in the `Pbox.inp` file. An unlimited number of waves can be combined by reapplying `putwave`.

Arguments: 1 to 11 wave parameters in the following predefined order:

`sh` is the name of a shape file.

`bw` is the bandwidth, in Hz.

`pw` is the pulsewidth, in sec.

`ofs` is the offset, in Hz.

`st` is a number specifying the spin status: 0 for Mz, or 1 for Mxy.

`ph` is the phase (or phase cycle, see `wavelib/supercycles`).

`fla` is the flip angle. Note that `fla` can override the default flip angle.

`trev` concerns time reversal. It can be used to cancel time reversal if spin status (`st`) is set to 1 for Mxy.

`d1` is the delay, in sec, prior the pulse.

`d2` is the delay, in sec, after the pulse.

`d0` is a delay or command prior to `d1`. If `d0=a`, the wave is appended to the previous wave.

Examples: `putwave('eburp1')`
`putwave('GARP',12000.0)`
`putwave('esnob',600,-1248.2,1,90.0,'n','n',0.001)`

See also: *User Guide: Liquids NMR*

Related: `Pbox` Pulse shaping software (U)
`setwave` Write a wave definition string into the Pbox.inp file (M)

pw Enter pulse width pw in degrees (C)

Syntax: `pw(flip_angle,<90_pulse_width>)`

Description: Calculates the flip tim, in μs , given a desired flip angle and 90° pulse. The value is entered into the parameter `pw`.

Arguments: `flip_angle` is the desired flip angle, in degrees.

`90_pulse_width` is the 90° pulse length, in μs . The default is the value of parameter `pw90`, if it exists.

Examples: `pw(30)`
`pw(90,12.8)`

See also: *Getting Started*

Related: `ernst` Calculate the Ernst angle pulse (C)
`pw` Pulse width (P)
`pw90` 90° pulse width (P)

pw Pulse width (P)

Description: Length of the final pulse in the standard two-pulse sequence. In “normal” 1D experiments with a single pulse per transient, this length is the observe pulse width.

Values: On systems with Data Acquisition Controller boards: 0, 0.1 to 8190 μs , in 12.5-ns steps. On systems with Pulse Sequence Controller or Acquisition Controller boards: 0, 0.2 to 8190 μs , in 25-ns steps. On systems with Output boards: 0, 0.2 to 8190 μs , in 0.1- μs steps. (Refer to the `acquire` statement in the manual *VNMR User Programming* for a description of these boards.)

On *GEMINI 2000* systems: 0, 0.2 to 4095 μs , in 100-ns steps.

See also: *Getting Started*

Related: `p1` First pulse width (P)
`pw` Enter pulse width parameter pw in degrees (C)

pw90 90° pulse width (P)

Description: Length of the 90° pulse. `pw90` is not used by pulse sequences directly, but is used by a number of commands to assist in setting up special experiments. `pw90` is also used by certain output programs to be able to print the value of the pulse width in degrees instead of microseconds. Note that this parameter must

be updated by the user and is not automatically determined or magically correct under all circumstances.

Values: On systems with Data Acquisition Controller boards: 0, 0.1 to 8190 μ s, in 12.5-ns steps. On systems with Pulse Sequence Controller or Acquisition Controller boards: 0, 0.2 to 8190 μ s, in 25-ns steps. On systems with Output boards: 0, 0.2 to 8190 μ s, in 0.1- μ s steps. (Refer to the `acquire` statement in the manual *VNMR User Programming* for a description of these boards.)

On *GEMINI 2000* systems: 0, 0.2 to 4095 μ s, in 100-ns steps.

See also: *Getting Started*

Related: `AC1S-AC11S` Autocalibration macros (M)
`pw` Enter pulse width parameter `pw` in degrees (C)

`pwd` **Display current working directory (C)**

Syntax: `pwd<:directory>`

Description: Displays the path of the current working directory.

Arguments: `directory` is a string variable with the path of the current directory.

Examples: `pwd:$name`

See also: *Getting Started*

Related: `cd` Change working directory (C)
`dir` List files in current directory (C)
`lf` List files in current directory (C)
`ls` List files in current directory (C)

`pwpat` **Shape of refocusing pulse (P)**

Applicability: Systems with imaging capabilities.

Description: Specifies the shape of the refocusing pulse `pw` in imaging experiments

Values: 'hard', 'sinc', 'gauss', 'sech', 'sine', or any shape resident in the system pulse shape library or libraries.

See also: *User Guide: Imaging*

Related: `plpat` Shape of an excitation pulse (P)
`pw` Pulse width (P)

`pwr` **Set power mode in directly detected dimension (C)**

Syntax: `pwr`

Description: Selects the power spectra display mode by setting `dmg='pwr'`. In the *power mode*, each real point in the displayed spectrum is calculated as the sum of the squares of the real and imaginary points comprising each respective complex data point. All information, including noise, is positive and the relationship between signal and noise is non-linear.

For multidimensional data, `pwr` has no effect on data prior to the second Fourier transform. If `pmode='full'`, `pwr` acts in concert with the commands `ph1`, `av1` or `pwr1` to yield the resultant contour display for the 2D data.

See also: *Getting Started*

Related: `av` Set abs. value mode in directly detected dimension (C)
`av1` Set abs. value mode in 1st indirectly detected dimension (C)
`dmg` Data display mode in directly detected dimension (P)
`ft` Fourier transform 1D data (C)

<code>ft1d</code>	Fourier transform along f_2 dimension (C)
<code>ft2d</code>	Fourier transform 2D data (C)
<code>pa</code>	Set phase angle mode in directly detected dimension (C)
<code>pa1</code>	Set phase angle mode in 1st indirectly detected dimension (C)
<code>ph</code>	Set phased mode in directly detected dimension (C)
<code>ph1</code>	Set phased mode in 1st indirectly detected dimension (C)
<code>pmode</code>	Processing mode for 2D data (P)
<code>pwr1</code>	Set power mode in 1st indirectly detected dimension (C)
<code>pwr2</code>	Set power mode in 2nd indirectly detected dimension (C)
<code>wft</code>	Weight and Fourier transform 1D data (C)
<code>wft1d</code>	Weight and Fourier transform f_2 of 2D data (M)
<code>wft2d</code>	Weight and Fourier transform 2D data (M)

`pwr1` **Set power mode in 1st indirectly detected dimension (C)**

Syntax: `pwr1`

Description: Selects the power spectra display mode along the first indirectly detected dimension by setting `dmg1='pwr1'`. If the parameter `dmg1` does not exist, `pwr1` creates it and sets it to '`pwr1`'. In the *power mode*, each real point in the displayed trace is calculated as the sum of the squares of the real and imaginary points comprising each respective complex data point. For hypercomplex data, the real-real and imaginary-real points from each respective hypercomplex data point are used in the summation. In this mode, all information, including noise, is positive and the relationship between signal and noise is non-linear.

The `pwr1` command is only needed if mixed-mode display is desired. If the parameter `dmg1` does not exist or is set to the null string, the display mode along the first indirectly detected dimension defaults to the display mode of the directly detected dimension (characterized by the parameter `dmg`). For the contour display of multidimensional data, the result of `pwr1` is the same as for traces, provided that `pmode='partial'` or `pmode=''`.

See also: *User Guide: Liquids NMR*

Related:	<code>dmg1</code>	Data display mode in 1st indirectly detected dimension (P)
	<code>pa</code>	Set phase angle mode in directly detected dimension (C)
	<code>pa1</code>	Set phase angle mode in 1st indirectly detected dimension (C)
	<code>pmode</code>	Processing mode for 2D data (P)
	<code>pwr</code>	Set power mode in directly detected dimension (C)
	<code>pwr2</code>	Set power mode in 2nd indirectly detected dimension (C)

`pwr2` **Set power mode in 2nd indirectly detected dimension (C)**

Syntax: `pwr2`

Description: Selects the power spectra display mode along the second indirectly detected dimension by setting `dmg2='pwr2'`. If `dmg2` does not exist or is set to the null string, `pwr2` will create `dmg2` and set it equal to '`pwr2`'. In the *power mode*, all information, including noise, is positive and the relationship between signal and noise is non-linear. Each real point in the displayed trace is calculated as the sum of the squares of the real and imaginary points comprising each respective complex data point. For hypercomplex data, the real-real and imaginary-real points from each respective hypercomplex data point are used in the summation.

The `pwr2` command is only needed if mixed-mode display is desired. If the parameter `dmg2` does not exist or is set to the null string, the display mode along the second indirectly detected dimension defaults to the display mode of the

directly detected dimension (characterized by the parameter `dmg`). For the contour display of multidimensional data, the result of `pwr2` is the same as for traces, provided that `pmode='partial'` or `pmode=''`.

See also: *User Guide: Liquids NMR*

Related:	<code>av2</code>	Set abs. value mode in 2nd indirectly detected dimension (C)
	<code>dmg2</code>	Data display mode in 2nd indirectly detected dimension (P)
	<code>ft1d</code>	Fourier transform along f_2 dimension (C)
	<code>ft2d</code>	Fourier transform 2D data (C)
	<code>ph2</code>	Set phased mode in 2nd indirectly detected dimension (C)
	<code>pmode</code>	Processing mode for 2D data (P)
	<code>pwr</code>	Set power mode in directly detected dimension (C)

pwrlist Active pulse power level parameter list (P)

Applicability: Systems with imaging capabilities.

Description: Contains an array of strings that define the names of the power level parameters associated with `plist` and `patlist`. The `nD`, `seqcon`, `plist`, `patlist`, `pwrlist`, `fliplist` and `sslist` parameters configure a particular parameter set for an application sequence defined by the value of the `seqfil` parameter. The `plist`, `patlist`, `pwrlist`, `fliplist` and `sslist` parameters provide information concerning the rf pulse and conjugate gradients used by the sequence.

Values: String array such as `pwrlist='tpwr1','tpwr2','tpwr3'`.

See also: *User Guide: Imaging*

Related:	<code>fliplist</code>	Standard flip angle list (P)
	<code>nD</code>	Application dimension (P)
	<code>patlist</code>	Active pulse template parameter list (P)
	<code>plist</code>	Active pulse length parameter list (P)
	<code>seqcon</code>	Acquisition loop control (P)
	<code>seqfil</code>	Application object code name (P)
	<code>sslist</code>	Conjugate gradient list (P)

pwsadj Adjust pulse interval time (M)

Applicability: Systems with waveform generators.

Syntax: `pwsadj(shape_file,pulse_parameter)`

Description: Adjusts the pulse interval time so that the pulse interval for the specified shape is an integral multiple of 100 ns. This ensures there is no time truncation error in executing the shaped pulse by waveform generators.

Arguments: `shape_file` is a file name of a shaped pulse file. The name can be specified with or without the `.RF` file extension. `pwsadj` first looks for the file name specified by `shape_file` in the user's `shapelib` directory. If the file specified is not found there, `pwsadj` then looks in the VNMR system `shapelib` directory.

`pulse_parameter` is a string containing the adjusted pulse interval time.

Examples: `pwsadj('pulse12','pulseparam')`

See also: *VNMR User Programming*

Related:	<code>dmfadj</code>	Adjust decoupler tip-angle resolution time (M)
	<code>dmf2adj</code>	Adjust second decoupler tip-angle resolution time (M)

pwxcals Decoupler pulse calibration (M)

Applicability: All systems except *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Syntax: `pwxcals`

Description: Provides an interactive method of selecting the decoupler (first, second, or third) and the nucleus (^{13}C , ^{15}N , or ^{31}P) to calibrate. The `pwxcals` pulse sequence determines the pulse width characteristics of the probe's decoupler channel(s) in indirect detection or triple resonance experiments. `pwxcals` can also be used to determine the rf field homogeneity of the decoupler.

The parameter `pw1` is arrayed to calibrate the 90° pulse width on the first decoupler. If a second decoupler is present, the parameter `pw2` is arrayed to calibrate the 90° pulse width on that decoupler. If a third decoupler is present, the parameter `pw3` is arrayed to calibrate the 90° pulse width on that decoupler. Other parameters include: `jc13` is the ^{13}C - ^1H coupling, constant, `jN15` is the ^{15}N - ^1H coupling constant, `jP31` is the ^{31}P - ^1H coupling constant, and `jname` is a selected calibration nucleus.

See also: *Getting Started*

pxset Assign Pbox calibration data to experimental parameters (M)

Syntax: `pxset<(file.ext)>`

Description: Retrieves experimental settings from a file and assigns them to corresponding experimental parameters using a dialog form. If no file name is provided, `pxset` extracts data from the `Pbox.cal` file that contains the output data of the last created waveform

Arguments: `file.ext` is the name of a shape or pattern file.

Examples: `pxset`
`pxset('Pbox.RF')`

See also: *User Guide: Liquids NMR*

Related: [Pbox](#) Pulse shaping software (U)
[pboxget](#) Extract Pbox calibration data (M)

pxshape Generates a single-band shape file (M)

Syntax: `pxshape('sh bw/pw ofs st ph fla trev \ d1 d2 d0',name,disp)`

Description: Generates a single-band waveform based on wave definition provided as a single string of wave parameters.

Arguments: A single string of 1 to 12 wave parameters in predefined order. Note that a single quote is required at the start and the end of the entire string, but no single quotes are required surrounding characters and strings inside the entire string.

`sh` is the name of a shape file.

`bw/pw` is either the bandwidth, in Hz, or the pulsewidth, in sec.

`ofs` is the offset, in Hz.

`st` is a number specifying the spin status: 0 for Mz, or 1 for Mxy.

`ph` is the phase (or phase cycle, see `wavelib/supercycles`).

`fla` is the flip angle. Note that `fla` can override the default flip angle.

`trev` is a time reversal. This can be used to cancel time reversal if spin status (`st`) is set to 1 for Mxy.

`d1` is the delay, in sec, prior the pulse.

`d2` is the delay, in sec, after the pulse.

`d0` is a delay or command prior to `d1`. If `d0=a`, the wave is appended to the previous wave.

`name` is the output file name. An extension is optional and can be used to override an internally defined shape type.

`disp` is the shape is displayed by default in the graphics window. If `disp` is set to 'n', the shape is not displayed.

Examples: `pxshape('eburp1','myshape.RF')`
`pxshape('GARP 12000.0','shape2','y')`
`pxshape('esnob 600.0 -1248.2 n 180.0 n n 0.001','xxx')`

See also: *User Guide: Liquids NMR*

Related: [Pbox](#) Pulse shaping software (U)

Pxsim Simulate Bloch profile for a shaped pulse (U)

Syntax: `Pxsim file <simtime <num_steps <add/sub>>>`

Description: Used by the `dprofile` macro to simulate a Bloch profile for a shaped pulse. `Pxsim` extracts the information necessary for simulation from the shape header. Only shape files containing this information can be processed.

Arguments: `file` is the name of a shape or pattern file including an `.RF` or `.DEC` extension. `Pxsim` searches for the file in the user's `shapelib` (`~/vnmrsys/shapelib`), and if not found there, it searches in the system `shapelib` (`vnmr/shapelib`).

`simtime` is the maximum simulation time (in sec) that can be provided.

`num_steps` is the number of steps in the profile.

`add/sub` is add (a) or subtract (s) from the previous simulation.

Examples: `Pxsim myshape.RF`

See also: *User Guide: Liquids NMR*

Related: [Pbox](#) Pulse shaping software (U)

Pxspy Create shape definition using Fourier coefficients (U)

Syntax: `Pxspy file`

Description: An interactive program that converts shaped pulse files into a Fourier series and produces an output file `pbox.cf` in the user's `shapelib` (`~/vnmrsys/shapelib`), which can be used to create a wave definition file in the `wavelib` directory. `Pxspy` can also be used to convert hard pulse decoupling sequences into soft ("cool") decoupling waveforms. The resulting Fourier coefficients can depend on the number of points in the waveform.

Arguments: `file` is the name of a shape or pattern file, including an `.RF`, `.DEC`, or `.GRD` extension. The name can be given as a relative name, absolute name, or as a simple name (i.e., with a path). If given as a simple name, `Pxspy` searches for the file in the user's `shapelib` (`~/vnmrsys/shapelib`), and then if not found there, it searches in the system `shapelib` (`vnmr/shapelib`).

Examples: `Pxspy myshape.RF`
`Pxspy /vnmr/shapelib/myshape.RF`
`Pxspy ~vnmrsys/shapelib/myshape.RF`

See also: *User Guide: Liquids*

Related: [Pbox](#) Pulse shaping software (U)

Q

QKexp **Set up quick experiment (M)**

Syntax: `QKexp (arguments)`

Description: Set up parameters for quick experiment for a chained acquisition. Multiple arguments can be given to define the chain. Default parameter values are used by the macro and or the probe file is used.

Examples: `QKexp (' PROTON ' , ' COSY ' , ' HMQC ')`
`QKexp (' PROTON ' , ' CARBON ' , ' HETCOR ' , ' gCOSY ')`

qtune **Tune probe using swept-tune graphical tool (C)**

Applicability: `UNITYINOVA` and `UNITYplus` systems.

Syntax: `qtune < (gain < , power >) >`

Description: Displays a real-time graph showing reflected power versus frequency for tuning probes. If the acquisition system has been recently rebooted, enter `su` before running `qtune`. Refer to the manual *Getting Started* for a detailed description of this tool.

Arguments: `gain` specifies the gain value, typically 20 to 50. The default is 50.
`power` specifies the power value, typically 60 to 70. The default is 60.

Examples: `qtune`
`qtune (20)`
`qtune (38 , 65)`

See also: *Getting Started*

Related: `tugain` Amount of receiver gain used by `qtune` (P)
`su` Submit a setup experiment to acquisition (M)
`tune` Assign frequencies on `UNITYINOVA` and `UNITYplus` (C)

? (question mark) Display individual parameter value (C)

Syntax: `parameter_name < [index] > ?`

Description: Displays the current numerical or string value of a parameter when the parameter name is followed by a question mark. No change is made to the value of the parameter. To display an individual element of an parameter array, provide the index in square brackets (e.g., `nt [3] ?` might display “`nt [3] = 2`”).

Certain parameters can be “turned off” by setting the parameter to 'n'. The display of a parameter that is turned off will be the phrase “Not Used” followed by the actual value in parentheses. For example, if `lb` is set to 1.5 and then set to 'n', entering `lb?` will display `lb= Not Used (1.5)`. Such a parameter can be “turned on” by setting it to 'y'. It will then have its prior value.

To show a parameter’s array of values or learn about its attributes, use the `display` command.

Arguments: `index` is the integer for a selected member of an arrayed parameter.

Examples: `lb?`
`sw?`
`pw [2] ?`

See also: *Getting Started*

Related: `display` Display parameters and their attributes (C)
`getvalue` Get value of a parameter in a tree (C)

R

r **Recall display parameter set (M)**

Syntax: (1) `rset_number`
 (2) `r(set_number)`

Description: Recalls the parameters `sp`, `wp`, `sp1`, `wp1`, `sp2`, `wp2`, `sc`, `wc`, `sc2`, `wc2`, `ho`, `vo`, `vs`, and `ai/nm` of a selected display parameter set. Not recalled are phase parameters, drift correction parameters, integral reset parameters, and reference parameters. This allows, for example, saving a set of display parameters, adjusting the phase or drift correction, and later recalling the display parameters without undoing the new phase or drift correction.

Arguments: `set_number` is the number, from 1 to 9, of a display parameter set.

Examples: `r 2`
`r (3)`

See also: *Getting Started*

Related:

<code>ai</code>	Select absolute intensity mode (C)
<code>fr</code>	Full recall of a display parameter set (M)
<code>ho</code>	Horizontal offset (P)
<code>nm</code>	Select normalized intensity mode (C)
<code>s</code>	Save display parameters as a set (M)
<code>sc</code>	Start of chart (P)
<code>sc2</code>	Start of chart in second direction (P)
<code>sp</code>	Start of plot in directly detected dimension (P)
<code>sp1</code>	Start of plot in 1st indirectly detected dimension (P)
<code>sp2</code>	Start of plot in 2nd indirectly detected dimension (P)
<code>vo</code>	Vertical offset (P)
<code>vs</code>	Vertical scale (P)
<code>wc</code>	Width of chart (P)
<code>wc2</code>	Width of chart in second direction (P)
<code>wp</code>	Width of plot in directly detected dimension (P)
<code>wp1</code>	Width of plot in 1st indirectly detected dimension (P)
<code>wp2</code>	Width of plot in 2nd indirectly detected dimension (P)

r1–r7 **Real-value storage for macros (P)**

Description: The seven parameters `r1`, `r2`, `r3`, `r4`, `r5`, `r6`, and `r7` are available in each experiment for macros to store a real value.

See also: *VNMR User Programming*

Related:

<code>dgs</code>	Display group of special/automation parameters (M)
<code>n1, n2, n3</code>	Name storage for macros (P)

ra **Resume acquisition stopped with sa command (C)**

Syntax: `ra`

Description: Resumes an experiment acquisition that was stopped with the `sa` command. `ra` is not permitted after any parameters have been brought into the stopped experiment with the `rt` or `rtp` macros. The parameters `dp` and `np` may not be altered.

`ra` applies to the experiment that you are joined to at the time the command is entered. If experiment 1 has been previously stopped with `sa`, you must be joined to experiment 1 for `ra` to resume that acquisition. If you are in experiment 2, entering `ra` has no effect on experiment 1.

If an experiment has been stopped with `sa`, you can increase the number of transients `nt` and resume the acquisition with `ra`. You cannot, however, increase `nt` and enter `ra` if the experiment had completed in a normal fashion (i.e., it was not stopped with `sa`).

Note that the completion time and remaining time shown in the Acquisition Status window are not accurate after `ra` is executed.

See also: *Getting Started*

Related:	<code>dp</code>	Double precision (P)
	<code>np</code>	Number of data points (P)
	<code>nt</code>	Number of transients (P)
	<code>rt</code>	Retrieve FID (M)
	<code>rtp</code>	Retrieve parameters (M)
	<code>sa</code>	Stop acquisition (C)

rcvr Receiver version in system (P)

Applicability: *GEMINI 2000* systems only.

Description: Identifies the version of receiver in the system. To determine the receiver version in a particular system, open the back door and locate the Observe Receiver board in the rf card cage. If there are two small 4-turn potentiometers on the edge of the top half of the board, the system has the Part No. 00-991758-02 version of the board, standard on 400-MHz *GEMINI 2000* systems. If the potentiometers are not present, the system has the Part No. 00-966914-02 version, standard on 200- and 300-MHz *GEMINI 2000* broadband systems. `rcvr` is listed in the `conpar` file.

Values: 0 for the 00-966914-02 version; 1 for the 00-991758-02 version.

See also: *VNMR and Solaris Software Installation*

Related:	<code>attens</code>	Fast attenuators present (P)
	<code>pfiltr</code>	Programmable filters (P)

rcvrs Which receivers to use (P)

Applicability: Systems with multiple receivers.

Description: A string of 'y's and 'n's that indicates which receivers should be used in a multiple receiver acquisition. Setting `rcvrs='y'` uses only the first receiver, and is equivalent to the parameter being absent.

Examples: `rcvrs='ny'` uses only the second receiver.
`rcvrs='yyyy'` uses four receivers.

Related:	<code>numrcvrs</code>	Number of receivers in the system (P)
----------	-----------------------	---------------------------------------

rcvrwt Weighting for different receivers (P)

Applicability: Systems with multiple receivers.

Description: An array of real numbers giving weighting factors to use when combining multiple receiver data. The *i*'th array element is used to weight data from the *i*'th receiver. Applying a weight factor is like increasing the gain of the receiver by

the same factor (but the weights are specified as numerical factors rather than in dB).

Examples: `rcvrwt=10,12,8`

Related: `addrcvrs` Combine data from multiple receivers (M)

rcvry Pre-trigger delay (P)

Applicability: Systems with imaging capabilities.

Description: Delays the start of most Varian imaging sequences until after the external trigger (the parameter `ticks`) is received by the system. The delay is still active in the non-triggered mode (`ticks=0`). Setting `hold=0` removes the delay in the sequence. The delays `rcvry` and `hold` are executed once per scan in Varian-provided sequences. In multislice imaging mode, this occurs at the beginning of the multislice pass, but not between the acquisition of individual slices.

Values: 0.1 μ s to 8192 sec, in units of seconds.

See also: *User Guide: Imaging*

Related: `hold` Post-trigger delay (P)
`ticks` Number of trigger pulses (P)

react Recover from error conditions during werr processing (M)

Syntax: `react<('wait')>`

Description: When an acquisition error occurs, any action specified by the `werr` parameter is executed. The `react` macro is a prototype for handling these errors. This macro can be invoked for error handling by setting `werr='react'`. The `acqstatus` parameter is provided so that `react` can determine which specific error has occurred.

Arguments: `'wait'` is a keyword for a special type of error handling during an automation run. The `react` macro always uses the `'next'` option when it calls the command `au`. Under certain conditions, it is also appropriate to use the `'wait'` option. `react` checks to see if an argument was passed to it; that is, `werr='werr(\ 'wait\ ')` to determine whether to use the `'wait'` option of `au`.

See also: *Getting Started; User Guide: Liquids NMR*

Related: `acqstatus` Acquisition status (P)
`au` Submit experiment to acquisition and process data (C)
`werr` Specify action when error occurs (C)
`werr` When error (P)

readallshims Read all shims from hardware (M)

Applicability: Not available on *GEMINI 2000*.

Syntax: `readallshims`

Description: Reads all shims from the hardware and sets the values into the shim parameters in the current parameter tree. The shims used depend on the `shimset` configuration. For the shim set on the Ultra•nmr shim system, `readallshims` is active only if hardware-to-software shim communication is enabled.

See also: *Getting Started*

Related: `load` Load status of displayed shims (P)
`readhw` Read current values of acquisition hardware (C)

`setallshims` Set all shims into hardware (M)
`sethw` Set values for hardware in acquisition system (C)
`shimset` Type of shim set (P)
`su` Submit a setup experiment to acquisition (M)

readbrutape Read Bruker data files from 9-track tape (U)

Syntax: (From UNIX) `readbrutape file <number_skipped>`

Description: A shell script that reads one file from a Bruker tape into a UNIX file with the name specified. Bruker tapes are likely to be made at 1600 bpi, although 1600 bpi is not a requirement.

Arguments: `file` is the name of the file read into UNIX. For identification, the `.bru` extension is added to the file name.

`number_skipped` is the number of files skipped and *includes* the header file (which is assumed to be the first file on the tape). The default is the script reads the first file after the header file. If `number_skipped` equals 0, there is no rewinding and the first file (or the next file) on the tape is read.

See also: *Getting Started*

Related: `convertbru` Convert Bruker data (M,U)

readhw Read current values of acquisition hardware (C)

Syntax: `readhw(param1,param2,...)<:value1,value2,...>`

Description: Returns or displays the current values of the lock system parameters `lockpower`, `lockgain`, `lockphase`, and `z0`.

The values of the shims can also be obtained. The particular shims that can be read depends upon the type of shim hardware present in the system. See the description of `shimset` for a list of the shim names for each type of shim hardware.

`readhw` cannot be used when an acquisition is in progress or when `acqi` is connected to the acquisition system.

Arguments: `param1`, `param2`, ... are the names of the parameters to be read.

`value1`, `value2`, ... are return variables to store the settings of the parameters specified. The default is to display the setting in the VNMR status window.

Examples: `readhw('z1c','z2c','z1','z2')`
`readhw('z1c','z2c','z1','z2'):r1,r2,r3,r4`

See also: *Getting Started*

Related: `lockgain` Lock gain (P)
`lockphase` Lock phase (P)
`lockpower` Lock power (P)
`readallshims` Read all shims from hardware (M)
`sethw` Set values for hardware in the acquisition system (C)
`shimset` Type of shim set (P)

readlk Read current lock level (C)

Syntax: `readlk<:lock_level>`

Description: Returns the same information as would be displayed on the digital lock display using the manual shimming window. `readlk` can be used in developing

automatic shimming methods such as shimming via grid searching. It *cannot* be used during acquisition or manual shimming.

Arguments: `lock_level` returns the current lock level.

Examples: `readlk`
`readlk:$level1`

See also: *VNMR User Programming*

Related: `alock` Automatic lock status (P)

readultra Read shim coil setting for Ultra•nmr shim system (M)

Applicability: Systems with the Ultra•nmr shim system.

Syntax: `readultra<(file_number)>`

Description: Reads shim set files for a Ultra•nmr shim system from a Sun floppy disk into VNMR. The floppy disk for Ultra•nmr contains up to 63 shim sets named `file1.dac` to `file63.dac`.

Arguments: `file_number` is the number of the shim set file, from 1 to 63. The default is to read all of the shim set files.

Examples: `readultra`
`readultra(6)`

See also: *Getting Started*

Related: `shimset` Type of shim set (P)
`svs` Save shim coil settings (C)

real Create a real variable without a value (C)

Syntax: `real(variable)`

Description: Creates a real variable without a value.

Arguments: `variable` is the name of the variable to be created.

Examples: `real('realvall')`

See also: *VNMR User Programming*

Related: `create` Create a new parameter in a parameter tree (C)
`string` Create a string variable (C)

record Record keyboard entries as a macro (M)

Syntax: `record<(file|'off')>`

Description: Records keyboard entries and stores the entries as a MAGICAL macro in the user's `maclib` directory. To start recording keyboard entries, enter `record`. You are prompted for a macro name (you can also give the name as an argument to `record`). The command line prompt then becomes "Command?" to indicate that the `record` macro is active. Type the MAGICAL commands to be recorded on the keyboard. Function keys can be included by entering F1 to F8 for function keys 1 to 8, respectively. Enter `off` or `record('off')` to finish the recording.

Arguments: `file` is the name of the macro file in which the entries are saved. The default is that the user is prompted for a file name. If the macro file name already exists, the user is asked if the file should be overwritten.

'off' is a keyword to stop recording the entries.

Examples: `record`
`record('mymacro')`
`record('off')`

See also: *VNMR User Programming*

redor1 **Set up parameters for REDOR1 pulse sequence (M)**

Applicability: Three-channel ^{UNITY}INOVA and UNITY*plus* systems with a triple-tuned MAS solids probe. This sequence is not supplied with *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems.

Syntax: `redor1`

Description: Sets up a parameter set, obtained with XPOLAR or XPOLAR1, for REDOR (rotational echo double-resonance) experiment.

See also: *User Guide: Solid-State NMR*

Related: `xpolar` Set up parameters for XPOLAR pulse sequence (M)
`xpolar1` Set up parameters for XPOLAR1 pulse sequence (M)

redosy **Restore 2D DOSY display from subexperiment (M)**

Syntax: `redosy`

Description: Restores the previous 2D DOSY display (if one exists) by recalling the data stored by the `dosy` macro in the file `subexp/dosy2Ddisplay` in the current experiment. `undosy` and `redosy` enable easy switching between the 1D DOSY data (spectra as a function of `gz1v11`) and the 2D DOSY display (signal as a function of frequency and diffusion coefficient).

See also: *User Guide: Liquids NMR*

Related: `dosy` Process DOSY experiments (M)
`undosy` Restore original 1D NMR data from subexperiment (M)

reffrq **Reference frequency of reference line (P)**

Description: Reference frequency, in MHz, of the reference line. This parameter is set by the `r1` macro. By defining `reffrq` as the conversion factor between Hz and ppm using the `unit` command, ppm calculations can be made.

If referencing is on (i.e., `refpos` is not set to 'n'), the `go`, `ga`, and `au` macros calculate values of `rfl` and `rfp` based on `reffrq` and `refpos`. If referencing is off, `go`, `ga`, and `au` set `reffreq` to `sfrq`.

See also: *Getting Started*

Related: `au` Submit experiment to acquisition and process data (M)
`crl` Clear reference line in directly detected dimension (M)
`ga` Submit experiment to acquisition and FT the result (M)
`go` Submit experiment to acquisition (M)
`reffrq1` Ref. frequency of reference line in 1st indirect dimension (P)
`reffrq2` Ref. frequency of reference line in 2nd indirect dimension (P)
`refpos` Position of reference frequency (P)
`rfl` Reference peak position in directly detected dimension (P)
`rfp` Reference peak frequency in directly detected dimension (P)
`r1` Set reference line in directly detected dimension (M)
`sfrq` Transmitter frequency of observe nucleus (P)
`unit` Define conversion units (C)

- reffrq1** **Reference frequency of reference line in 1st indirect dimension (P)**
- Description: Reference frequency, in MHz, of the reference line in the first indirect dimension of a nD experiment. This parameter should be used as the conversion factor between hertz and ppm in the first indirect dimension.
- See also: *User Guide: Liquids NMR*
- Related: `cr11` Clear reference line in 1st indirectly detected dimension (M)
`reffrq` Reference frequency of reference line (P)
`refpos1` Position of reference frequency in 1st indirect dimension (P)
- reffrq2** **Reference frequency of reference line in 2nd indirect dimension (P)**
- Description: Reference frequency, in MHz, of the reference line in the second indirect dimension of a 2D experiment. This parameter should be used as the conversion factor between hertz and ppm in the second indirect dimension.
- See also: *User Guide: Liquids NMR*
- Related: `cr12` Clear reference line in 2nd indirectly detected dimension (M)
`reffrq` Reference frequency of reference line (P)
`refpos2` Position of reference frequency in 2nd indirect dimension (P)
- refpos** **Position of reference frequency (P)**
- Description: Position of reference frequency, set by the `setref` and `r1` macros. Setting `refpos='n'` indicates that referencing has been turned off. The `cr1` macro turns referencing off.
- Values: Because all spectra are (by definition) referenced to a frequency at 0 ppm, `refpos` is either 0 or “not used”.
- See also: *Getting Started*
- Related: `cr1` Clear reference line in directly detected dimension (M)
`reffrq` Reference frequency of reference line (P)
`refpos1` Position of reference frequency in 1st indirect dimension (P)
`refpos2` Position of reference frequency in 2nd indirect dimension (P)
`r1` Set reference line indirectly detected dimension (M)
`setref` Set frequency referencing (M)
- refpos1** **Position of reference frequency in 1st indirect dimension (P)**
- Description: Position of reference frequency in the first indirect dimension of a nD experiment, set by `setref1` and `r11` macros. Setting `refpos1='n'` indicates that f1 referencing has been turned off. The `cr11` macro turns f1 referencing off.
- Values: Because all spectra are (by definition) referenced to a frequency at 0 ppm, `refpos1` is either 0 or “not used”.
- See also: *User Guide: Liquids NMR*
- Related: `cr11` Clear reference line in 1st indirectly detected dimension (M)
`reffrq1` Ref. frequency of reference line in 1st indirect dimension (P)
`refpos` Position of reference frequency (P)
`r11` Set reference line in 1st indirect dimension (M)
`setref1` Set frequency referencing for 1st indirectly detected dimension (M)

- refpos2** **Position of reference frequency in 2nd indirect dimension (P)**
- Description: Position of reference frequency in the second indirect dimension of a 3D experiment, set by `setref2` and `r12` macros. Setting `refpos2='n'` indicates that f2 referencing has been turned off in 3D spectra. The `cr12` macro turns f2 referencing off.
- Values: Because all spectra are (by definition) referenced to a frequency at 0 ppm, `refpos2` is either 0 or “not used”.
- See also: *User Guide: Liquids NMR*
- Related: `cr12` Clear reference line in 2nd indirectly detected dimension (M)
`reffrq2` Ref. frequency of reference line in 2nd indirect dimension (P)
`refpos` Position of reference frequency (P)
`r12` Set reference line in 2nd indirect dimension (M)
`setref2` Set frequency referencing for 2nd indirectly detected dimension (M)
- refsource1** **Center frequency in 1st indirect dimension (P)**
- Description: Holds a parameter name to be used as the center frequency in the first indirect dimension of 2D experiments. If `refsource1` does not exist, the default is `'sfrq'`.
- For 2D experiments, the second dimension may be related to `sfrq` if it is a homonuclear experiment. The second dimension may also be related to `dfrq` if it is a heteronuclear experiment. `refsource1` would then be set as `refsource1='sfrq'` and `refsource1='dfrq'`, respectively.
- See also: *User Guide: Liquids NMR*
- Related: `dfrq` Transmitter frequency of first decoupler (P)
`refsource2` Center frequency in 2nd indirect frequency (P)
`sfrq` Transmitter frequency of observe nucleus (P)
- refsource2** **Center frequency in 2nd indirect dimension (P)**
- Description: Holds a parameter name to be used as the center frequency in the second indirect dimension. `refsource2` is analogous to `refsource1`.
- See also: *User Guide: Liquids NMR*
- Related: `refsource1` Center frequency in 1st indirect dimension (P)
- region** **Divide spectrum into regions (C)**
- Syntax: `region<(tail_length,relative_number,threshold, number_points,tail_size)><:number_regions >`
- Description: Breaks a spectrum up into regions containing peaks.
- Arguments: `tail_length` is the length from 0.0 to `sw`, in Hz, that is added to the start and end of each calculated peak region; default value is `sw/10`. The default value is used if a negative number is entered for this argument. If the addition of these wings would cause overlap between adjacent regions, the wings are reduced until the regions no longer overlap.
- `relative_number` is a number that, in combination with other factors, governs the relative number of regions to be found. The default is 12, which is used if 0 is entered for this argument. `relative_number` is used as part of a test to determine whether two spectral areas containing peaks are close enough together to be represented as a single region. There are no strict rules that associate the value of `relative_number` to the total number of regions that

will be found. In general, increasing this number decreases the number of regions that will be found and increases the size of an individual region. A value of 1 would give more regions; a value of 100 would give fewer regions.

`threshold` is a sensitivity factor used to decide if a data point is large enough, relative to the noise level, to qualify it as part of a peak. The default value is 0.6, which is used if 0 is entered for this argument. Smaller values of `threshold` make peak selection more sensitive; larger values make peak selection less sensitive.

`number_points` governs the number of successive data points, normally from 7 to 40, that must qualify as part of a peak (see the description of `threshold` above) in order for that spectral area to be considered a real peak. The default value is a function of `fn`, `sw`, weighting functions, and other values. The default is used if 0 is entered for this argument. For carbon spectra with large spectral windows, experimental peaks often contain only one or two data points. Adjust `number_points` to 1 or 2 in those cases.

`tail_size` is a number that, in combination with `relative_number` and other factors, governs whether two spectral areas that contain peaks are close enough together to be represented as a single region. The default value is used if 0 is entered for this argument.

`number_regions` is the total number of regions determined by `region`.

Examples: `region`
`region:$1`
`region(50,0,1)`
`region(-1,0,0,2):r1`

See also: *Getting Started*

Related: `fn` Fourier number in directly detection dimension (P)
`sw` Spectral width in directly detected dimension (P)

relayh Set up parameters for RELAYH pulse sequence (M)

Syntax: `relayh`

Description: Sets up parameters for absolute-value COSY, or a single or double RELAY-COSY pulse sequence.

See also: *User Guide: Liquids NMR*

Related: `cosy` Set up parameters for COSY pulse sequence (M)
`cosyps` Set up parameters for phase-sensitive COSY (M)
`dqcosy` Set up parameters for double quantum filtered COSY (M)

rename Move and/or rename a file (C)

Syntax: `rename(from_file,to_file)`

Description: Renames and/or moves a file or directory. `rename` is identical in function to the command `mv`.

Arguments: `from_file` is the name of the file to be moved to renamed.
`to_file` is the name of the file after moving or renaming it. If the `from_file` argument has an extension such as `.fid` or `.par`, be sure the `to_file` argument has the same extension.

Examples: `rename('/home/vnmr1/vnmrsys/seqlib/d2pul',
'/vnmr/seqlib/d2pul')`

See also: *Getting Started*

Related: `copy` Copy a file (C)
`cp` Copy a file (C)
`delete` Delete a file, parameter directory, or FID directory (C)
`mv` Move and/or rename a file (C)
`rm` Delete file (C)

rescal Calculate pixel size and spatial resolution (M)

Applicability: Systems with imaging capabilities.

Syntax: `rescal('silent')>>:pixrc,pixrd,pixpc,pixpd`

Description: Calculates the pixel sizes for the acquisition (spatial resolution) and display (digital resolution). The results are displayed in the text window. As an option, the results can be returned to variables, which allows the user to call `rescal` from within other macros and use it to calculate this basic information. This macro can be used before acquisition to check that the chosen conditions lead to the desired spatial resolution.

Arguments: `'silent'` is a keyword to suppress the text window output.

`pixrc` returns the readout pixel size (collected).

`pixrd` returns the readout pixel size (displayed).

`pixpc` returns the phase encode pixel size (collected).

`pixpd` returns the phase encode pixel size (displayed).

Examples: `rescal`
`rescal('silent'):r1,r2,r3,r4`

See also: *User Guide: Imaging*

resetf3 Reset parameters after a partial 3D Fourier transform (M)

Syntax: `resetf3`

Description: Restores the acquisition parameter `sw`, the processing parameter `fn`, and the display parameters `sp`, `wp`, `rfl`, and `rfp` in the 3D parameter set, which are read into VNMR by either the `select` command or the `dplane` or `dproj` macros. These parameters were modified due to the selection of regional f_3 processing (`ptspec3d = 'ynn'`). The original value for each of these parameters is stored in the parameter `$sv`, where `$` represents `sw`, `fn`, `sp`, `wp`, `rfl`, or `rfp` (e.g., `swsv`).

If a 2D plane into VNMR is retrieved from a 3D transformed data set that was processed with regional f_3 processing, `resetf3` must be run before executing `ft3d` in that particular VNMR environment.

See also: *User Guide: Liquids NMR*

Related: `dplane` Display a 3D plane (M)
`dproj` Display a 3D plane projection (M)
`fn` Fourier number in directly detected dimension (P)
`ft3d` Perform a 3D Fourier transform (M)
`ptspec3d` Region-selective 3D processing (P)
`rfl` Ref. peak position in directly detected dimension (P)
`rfp` Ref. peak frequency in directly detected dimension (P)
`select` Select a spectrum or 2D plane without displaying it (C)
`sp` Start of plot (P)
`sw` Spectral width in directly detected dimension (P)
`wp` Width of plot (P)

resolv **Set resolution enhancement parameters (M)**

Syntax: `resolv<(a,b)>`

Description: Calculates a default resolution enhancement function, setting up `lb` and `gf` based on the acquisition time `at`. “Zero-filling” is also accomplished, if possible, by making `fn` $\geq 2 * np$.

Arguments: `a` sets a value of `lb` using `lb = -0.318 / (a * sw)`. The default for `a` is 0.1.
`b` sets a value of `gf` using `gf = b * sw`. The default for `b` is 0.3.

Examples: `resolv`
`resolv(.2,.4)`

See also: *Getting Started*

Related: `at` Acquisition time (P)
`fn` Fourier number in directly detected dimension (P)
`gf` Gaussian function in directly detected dimension (P)
`lb` Line broadening in directly detected dimension (P)
`np` Number of data points (P)
`sw` Spectral width in directly detected dimension (P)

resto **NMR resonance offset frequency (P)**

Applicability: Systems with imaging capabilities.

Description: NMR resonance offset frequency, in Hz.

See also: *User Guide: Imaging*

Related: `tn` Transmitter nucleus (P)
`sfrq` Spectrometer frequency (P)

resume **Resume paused acquisition queue (C)**

Syntax: `resume`

Description: Enables continuing submitting experiments to the acquisition system. For experiments initiated with the command `au('wait')`, the acquisition is paused during the time of data processing in order to prevent the acquisition from submitting new experiments that might be queued. `resume` then allows the data processing macro to initiate another acquisition with `au('next')`, which is then performed immediately instead of at the end of the queue.

See also: *User Guide: Liquids NMR*

Related: `au` Submit experiment to acquisition and process data (C)

return **Terminate execution of a macro (C)**

Syntax: `return<(expression1,expression2,...)>`

Description: Terminates the execution of a macro and optionally returns values to another calling macro. This is usually used after testing some condition. `return` is used only in macros and not entered from the keyboard.

Arguments: `expression1,expression2,...` are return values to another calling macro.

See also: *VNMR User Programming*

Related: `abort` Terminate action of calling macro and all higher macros (C)

- rev** **System software revision level (P)**
- Description: Stores a string identifying the VNMR software version for the system. This parameter is not be entered by the user, but can be examined by typing `rev?`.
- Values: 'VERSION 6.1 REVISION A', etc.
- See also: *VNMR and Solaris Software Installation*
- Related: `revdate` System software preparation date (P)
-
- revdate** **System software preparation date (P)**
- Description: Stores a string identifying the date the current VNMR software version was prepared. This parameter is not be entered by the user, but can be examined by typing `revdate?`.
- Values: 'Jan 12, 1998', etc.
- See also: *VNMR and Solaris Software Installation*
- Related: `rev` System software revision level (P)
-
- rfband** **RF band in use (P)**
- Applicability: All systems except *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.
- Description: Indicates which rf band of the amplifier is in use for each channel.
- Values: A string, such as 'hlc', in which the first channel is determined by the first character, the second channel is determined by the second character, and so forth. The following values are available for each channel:
- 'h' indicates the high rf band is in use on the channel.
 - 'l' indicates the low rf band is in use on the channel.
 - 'c' indicates the system software will calculate whether to use the high band or the low band for the channel.
- See also: *Getting Started*
-
- rfblk** **Reverse FID block (C)**
- Syntax: `rfblk(<src_expno>, src_blk_no, dest_expno, dest_blk_no)`
- Description: Reverses and copies data from a source FID block specified by `src_blk_no` to a destination FID block specified by `dest_expno` and `dest_blk_no`, using memory-mapped input and output. The file header determines the size and type of data to reverse.
- `rfblk` searches for the source and destination FID file in the directory `$vnmruser/expN/acqfil`; N is the requested experiment number or the current experiment number. If the FID file is not open, `rfblk` opens the file, copies the data, and closes the file. If a number of blocks need to be copied, explicitly opening and closing the files with the commands `mfopen` and `mfclose` can significantly speed up the data reformatting process.
- `rfblk` can also be used to append blocks of data to a FID file by specifying that the `dest_blk_no` is greater than the number of blocks in a file.
- Be aware that `rfblk` can modify data returned to an experiment with the `rt` command. To avoid modification, enter the following sequence of VNMR commands before running `rfblk`:

```
cp(curexp+' /acqfil/fid' ,curexp+' /acqfil/fidtmp' )
rm(curexp+' /acqfil/fid' )
mv(curexp+' /acqfil/fidtmp' ,curexp+' /acqfil/fid' )
```

Arguments: `src_expno` specifies the experiment number of the source FID file. The default is the FID file of the current experiment.

`src_blk_no` specifies the source block of data to be copied. Block numbers run from 1 to the number of blocks in a file.

`dest_expno` specifies the experiment number of the destination FID file.

`dest_blk_no` specifies the destination block to send the copied data.

Examples: `rfblk(1,2,1)` reverses and copies block 1 from the current experiment to block 1 of experiment 2.

See also: *VNMR User Programming*

Related:	<code>mfbk</code>	Move FID block (C)
	<code>mfclose</code>	Memory map close FID file (C)
	<code>mfdata</code>	Move FID data (C)
	<code>mfopen</code>	Memory map open FID file (C)
	<code>mfttrace</code>	Move FID trace (C)
	<code>rfdata</code>	Reverse FID data (C)
	<code>rftrace</code>	Reverse FID trace (C)

rfchannel Independent control of rf channel selection (P)

Applicability: ^{UNITY}*INOVA* and *UNITYplus* systems.

Description: Gives override capability over the selection of rf channels. `rfchannel` does not normally exist but can be created by a user with the command `create('rfchannel','flag')`.

On ^{UNITY}*INOVA* and *UNITYplus* systems, the control of each rf channel is built around a collection of parameters and pulse sequence statements. The frequency of channel 1 is set by `sfrq` and `tof`, its power by `tpwr` and `tpwrf`. The first decoupler uses the corresponding parameters `dfrq`, `dof`, `dpwr`, and `dpwrf`, respectively. Furthermore, the decoupler can have modulation modes specified with the parameters `dmf`, `dm`, `dmm`, `dres`, `dseq`, and `homo`. The second decoupler has the same set of parameters as the first decoupler and they are distinguished by appending a 2 to each name. That is, the names are `dfrq2`, `dof2`, `dpwr2`, `dpwrf2`, `dmf2`, `dm2`, `dmm2`, `dres2`, `dseq2`, and `homo2`. The third decoupler would use parameters with a 3 appended: `dfrq3`, `dof3`, `dpwr3`, `dpwrf3`, `dmf3`, `dm3`, `dmm3`, `dres3`, `dseq3`, and `homo3`. The `rfchannel` parameter provides a mechanism to override the default parameter usage.

Values: A string of one to four characters in which the position of each character identifies the rf channel controlled.

- The first character selects which rf channel (1 to 4) the parameters `sfrq`, `tof`, `tpwr`, etc. control. The first character also identifies the rf channel used as the receiver.
- The second character selects which rf channel (1 to 4) the parameters `dfrq`, `dof`, `dpwr`, etc. control.
- The third character maps the parameter set `dfrq2`, `dof2`, `dpwr2`, etc. to an rf channel (1 to 4).
- The fourth character maps `dfrq3`, `dof3`, `dpwr3`, etc. to an rf channel (1 to 4).

For example, `rfchannel= '132'` would exchange control of the second and third rf channels from the default parameter usage.

The number of characters in the `rfchannel` parameter must match the number of real rf channels (defined by the parameter `numrfch`) and each rf channel must be selected by the parameter.

Besides remapping the parameters to different rf channels, pulse sequence statements are also remapped. For example, if `rfchannel= '132'`, then statements `decpulse`, `decshaped_pulse`, `decoffset`, `decpower`, `decspinlock`, and so on are applied on rf channel 3 and `dec2pulse`, `dec2shaped_pulse`, and so on are applied on rf channel 2.

An obvious use for this remapping is on systems with the decoupler set to U+ H1 Only in the CONFIG window. On these systems, if multinuclear pulses are needed and ^1H needs to be observed, the parameter sets that assume a dual-broadband system can be used and the parameters remapped by setting `rfchannel= '21'`. However, internal logic checks if the first decoupler is set to U+ H1 Only, `tn` is set to 'H1', and `dn` is not set to 'H1'. If these settings are the case, the parameter mapping for rf channels 1 and 2 is exchanged automatically.

See also: *Getting Started; VNMR User Programming*

Related:	<code>create</code>	Create new parameter in parameter tree (C)
	<code>dfrq</code>	Transmitter frequency for first decoupler (P)
	<code>dm</code>	Decoupler mode for first decoupler (P)
	<code>dmf</code>	Decoupler modulation frequency for first decoupler (P)
	<code>dmm</code>	Decoupler modulation mode for first decoupler (P)
	<code>dn</code>	Nucleus for first decoupler (P)
	<code>dof</code>	Frequency offset for first decoupler (P)
	<code>dpwr</code>	Power level for first decoupler with linear amplifier (P)
	<code>dpwrf</code>	First decoupler fine power (P)
	<code>dres</code>	Tip-angle resolution for first decoupler (P)
	<code>dseq</code>	Decoupler sequence for first decoupler (P)
	<code>homo</code>	Homodecoupling control for first decoupler (P)
	<code>numrfch</code>	Number of rf channels (P)
	<code>sfrq</code>	Transmitter frequency for observe nucleus (P)
	<code>tn</code>	Nucleus for observe transmitter (P)
	<code>tof</code>	Frequency offset for observe transmitter (P)
	<code>tpwr</code>	Observe transmitter power level with linear amplifiers (P)
	<code>tpwrf</code>	Observe transmitter fine power (P)

rfchtype **Type of rf channel (P)**

Applicability: ^{UNITY}INOVA and UNITYplus systems.

Description: Configuration parameter for type of rf on each channel. The value for a channel is set using the Type of RF label in the CONFIG window (opened by entering `config`). Pulse sequence programs check `rfchtype` to determine if indirect detection should be used for some experiments. Indirect detection occurs automatically on a ^{UNITY}INOVA and UNITYplus if the decoupler is set to U+ H1 Only in the CONFIG window, `tn` is set to 'H1', and `dn` is not set to 'H1'.

Values: The values of `rfchtype` parallel the `rfdtype` values. The only distinction is that the setting for `rfdtype` is 'd' on the U+ Direct Synthesis and U+ H1 Only entries.

'U+ Direct Synthesis' is the setting for a ^{UNITY}INOVA or UNITYplus with direct synthesis (U+ Direct Synthesis in the CONFIG window).

'U+ H1 Only' is a fixed-frequency proton ^{UNITY}INOVA or UNITYplus (U+ H1 Only in CONFIG window).

'Deuterium Decoupler' is the setting for a ^{UNITY}INOVA deuterium decoupler channel.

'Direct Synthesis' is the setting for direct synthesis (Direct Synthesis in the CONFIG window).

'Broadband' is the setting for broadband (Broadband in the CONFIG window).

'Fixed Frequency' is the setting for fixed frequency (Fixed Frequency in the CONFIG window).

'SIS Modulator' is the setting for imaging modulator (SIS Modulator in the CONFIG window).

See also: *VNMR and Solaris Software Installation*

Related:	<code>config</code>	Display current configuration and possibly change it (M)
	<code>dn</code>	Nucleus for first decoupler (P)
	<code>rftype</code>	Type of rf generation (P)
	<code>tn</code>	Nucleus for observe transmitter (P)

rfcoil **RF pulse calibration identity (P)**

Applicability: Systems with imaging capabilities.

Description: Contains a string identifying the rf pulse calibration.

See also: *User Guide: Imaging*

Related:	<code>gcoil</code>	Read data from gradient calibration tables (P)
	<code>plist</code>	Active pulse length parameter list (P)

rfdata **Reverse FID data (C)**

Syntax: `rfdata (<src_expno,>src_blk_no,src_start_loc, \ dest_expno,dest_blk_no,dest_start_loc,num_points)`

Description: Reverses and copies data specified by `src_start_loc` from a FID block specified by `src_blk_no` to a destination location specified by `dest_expno`, `dest_blk_no`, and `dest_start_loc`, using memory-mapped input and output. The data point locations and the `num_points` to be reversed are specified by data points corresponding to the `np` parameter, not bytes or complex points; however, when reversing the data, `rfdata` looks at the file header to determine the size and type of data to reverse.

`rfdata` searches for the source and destination FID file in the directory `$vnmruser/expN/acqfil`; N is the requested experiment number or the current experiment number. If the FID file is not open, `rfdata` opens the file, copies the data, and closes the file. If a number of blocks need to be copied, explicitly opening and closing the files with the commands `mfopen` and `mfclose` can significantly speed up the data reformatting process.

Be aware that `rfdata` can modify data returned to an experiment with the `rt` command. To avoid modification, enter the following sequence of VNMR commands before running `rfdata`:

```
cp(curexp+' /acqfil/ fid', curexp+' /acqfil/ fidtmp' )
rm(curexp+' /acqfil/ fid' )
mv(curexp+' /acqfil/ fidtmp', curexp+' /acqfil/ fid' )
```

Arguments: `src_expno` specifies the experiment number of the source FID file. The default is the FID file of the current experiment.

`src_blk_no` specifies the source block of data to be copied. Block numbers run from 1 to the number of blocks in a file.

`src_start_loc` specifies the starting data location within the specified block to copy the data. Data locations start from 0 and are specified as data points corresponding to the `np` parameter.

`dest_expno` specifies the experiment number of the destination FID file.

`dest_blk_no` specifies the destination block to send the copied data.

`dest_start_loc` specifies the starting data destination location within the specified block to send the copied data.

Examples: `rfdata(1,0,2,1,(nv-1)*np,np)` copies and reverses `np` points of data from the starting location 0 of block 1 of the current experiment to the data location $(nv-1) * np$ of block 1 of experiment 2.

See also: *VNMR User Programming*

Related:	<code>mfbk</code>	Move FID block (C)
	<code>mfclose</code>	Memory map close FID file (C)
	<code>mfdata</code>	Move FID data (C)
	<code>mfopen</code>	Memory map open FID file (C)
	<code>mftrace</code>	Move FID trace (C)
	<code>rfbk</code>	Reverse FID block (C)
	<code>rftrace</code>	Reverse FID trace (C)

rf1 Reference peak position in directly detected dimension (P)

Description: Actual position of the reference line in the spectrum (i.e., the distance from the right edge of the spectrum to the reference line). If there is no reference line in the spectrum, `rf1` can be used to enter the frequency where the reference line would appear if the line were present in the spectrum.

Values: Number, in Hz.

See also: *Getting Started*

Related:	<code>rf11</code>	Reference peak position in 1st indirectly detected dimension (P)
	<code>rf12</code>	Reference peak position in 2nd indirectly detected dimension (P)
	<code>rfp</code>	Reference peak frequency in directly detected dimension (P)

rf11 Reference peak position in 1st indirectly detected dimension (P)

Description: Analogous to the `rf1` parameter except that `rf11` applies to the first indirectly detected dimension of a multidimensional data set. `rf11` can either be set manually or be adjusted automatically when the macro `r11` is used to assign a reference line.

Values: Number, in Hz.

See also: *User Guide: Liquids NMR*

Related:	<code>rf1</code>	Reference peak position in directly detected dimension (P)
	<code>rf12</code>	Reference peak position in 2nd indirectly detected dimension (P)
	<code>rfp1</code>	Reference peak frequency in 1st indirectly detected dimension (P)

rf12 Reference peak position in 2nd indirectly detected dimension (P)

Description: Analogous to the `rf1` parameter except that `rf12` applies to the second indirectly detected dimension of a multidimensional data set. `rf12` can either be set manually or be adjusted automatically when the macro `r12` is used to assign a reference line.

Values: Number, in Hz.

See also: *User Guide: Liquids NMR*

Related: **rf1** Reference peak position in directly detected position (P)
rf11 Reference peak position in 1st indirectly detected dimension (P)
rfp2 Reference peak frequency in 2nd indirectly detected dimension (P)

rfp Reference peak frequency in directly detected dimension (P)

Description: Sets the frequency to be assigned to the reference line in the spectrum. **rfp** is always stored in Hz, but can be entered in ppm by using the **p** suffix (e.g., **rfp=2.1p**).

Values: Number, in Hz.

See also: *Getting Started*

Related: **rf1** Reference peak position in directly detected dimension (P)
rfp1 Ref. peak frequency in 1st indirectly detected dimension (P)
rfp2 Ref. peak frequency in 2nd indirectly detected dimension (P)
r1 Set reference line in directly detected dimension (M)

rfp1 Reference peak frequency in 1st indirectly detected dimension (P)

Description: Analogous to the **rfp** parameter except that **rfp1** applies to the first indirectly detected dimension of a multidimensional data set. **rfp1** can either be set manually or be assigned a value when **r11** is called with an argument (e.g., **r11(7.2p)** assigns the value of 7.2 ppm to **rfp1**).

Values: Number, in Hz.

See also: *User Guide: Liquids NMR*

Related: **rf11** Ref. peak position in 1st indirectly detected dimension (P)
rfp Ref. peak frequency in directly detected dimension (P)
rfp2 Ref. peak frequency in 2nd indirectly detected dimension (P)
r11 Set reference line in 1st indirectly detected dimension (M)

rfp2 Reference peak frequency in 2nd indirectly detected dimension (P)

Description: Analogous to the **rfp** parameter except that **rfp2** applies to the second indirectly detected dimension of a multidimensional data set. **rfp2** can be set manually or be assigned a value when **r12** is called with an argument. For example, entering **r12(7.2p)** assigns the value of 7.2 ppm to **rfp2**.

Values: Number, in Hz.

See also: *User Guide: Liquids NMR*

Related: **rf12** Reference peak position in 2nd indirectly detected dimension (P)
rfp Reference peak frequency in directly detected dimension (P)
rfp1 Reference peak frequency in 1st indirectly detected dimension (P)
r12 Set reference line in 2nd indirectly detected dimension (C)

rftrace Reverse FID trace (C)

Syntax: **rftrace**(<src_expno,<src_blk_no,<src_trace_no, \
dest_expno,<dest_blk_no,<dest_trace_no)

Description: Reverses and copies FID traces specified by **src_trace_no** from a FID block specified by **src_blk_no** to a destination location specified by **dest_expno**, **dest_blk_no**, and **dest_trace_no**, using memory-

mapped input and output. The file header determines the size and type of data to be reversed.

`rftrace` searches for the source and destination FID file in the directory `$vnmruser/expN/acqfil`; `N` is the requested experiment number or the current experiment number. If the FID file is not open, `rftrace` opens the file, copies the data, and closes the file. If a number of blocks need to be copied, explicitly opening and closing the files with the commands `mfopen` and `mfclose` can significantly speed up the data reformatting process.

You cannot use `rftrace` to append data to a FID file. Its purpose is for moving around data.

Be aware that `rftrace` can modify data returned to an experiment with the `rt` command. To avoid modification, enter the following sequence of VNMR commands before running `rftrace`:

```
cp(curexp+' /acqfil/fid' ,curexp+' /acqfil/fidtmp' )
rm(curexp+' /acqfil/fid' )
mv(curexp+' /acqfil/fidtmp' ,curexp+' /acqfil/fid' )
```

Arguments: `src_expno` specifies the experiment number of the source FID file. The default is the FID file of the current experiment.

`src_blk_no` specifies the source block of data to be copied. Block numbers run from 1 to the number of blocks in a file.

`src_trace_no` specifies the source trace of data within the specified block to be copied. Trace numbers run from 1 to number of traces in a file.

`dest_expno` specifies the experiment number of the destination FID file.

`dest_blk_no` specifies the destination block to send the copied data.

`src_trace_no` specifies the destination trace of data within the specified block to be copied. Trace numbers run from 1 to the number of traces in a file.

Examples: `rftrace(1,1,2,1,nv)` copies and reverses trace 1 from block 1 of the current experiment to trace `nv` of block 1 of experiment 2.

See also: *VNMR User Programming*

Related:	<code>mfbk</code>	Move FID block (C)
	<code>mfclose</code>	Memory map close FID file (C)
	<code>mfdata</code>	Move FID data (C)
	<code>mfopen</code>	Memory map open FID file (C)
	<code>mfttrace</code>	Move FID trace (C)
	<code>rfblk</code>	Reverse FID block (C)
	<code>rfdata</code>	Reverse FID data (C)

rfstype **Type of rf generation (P)**

Description: Configuration parameter for type of rf generation on each rf channel. On the *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems, the value is set using the System Type label in the CONFIG window (opened by entering `config`). On other systems, the value is set using the Type of RF label in the CONFIG window.

Values: The values of `rfstype` parallel the `rfchtype` values. The only distinction is that on ^{UNITY}*INOVA* and *UNITYplus*, the setting for `rfstype` is 'd' on the entries U+ Direct Synthesis and U+ H1 Only. On *UNITY* and *VXR-S*, 'b', 'a', or 'c' can be used for each channel. On the *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*, only 'ee' or 'fe' is used.

'd' is the setting for a ^{UNITY}INOVA or UNITYplus with direct synthesis (U+ Direct Synthesis in the CONFIG window) or a fixed-frequency proton ^{UNITY}INOVA or UNITYplus (U+ H1 Only in CONFIG window).

'l' is the setting for a ^{UNITY}INOVA deuterium decoupler channel.

'c' is the setting for direct synthesis (Direct Synthesis in the CONFIG window).

'b' is the setting for broadband (Broadband in the CONFIG window).

'a' is the setting for fixed frequency (Fixed Frequency in the CONFIG window).

'm' is the setting for imaging modulator (SIS Modulator in the CONFIG window).

'ee' is the setting for MERCURY-VX 4-nucleus, MERCURY 4-nucleus, and GEMINI 2000 ¹H/¹³C systems (4 Nucleus or 1H/13C in the CONFIG window).

'fe' is the setting for MERCURY-VX broadband, MERCURY broadband, and GEMINI 2000 broadband systems (Broadband in the CONFIG window).

See also: *VNMR and Solaris Software Installation*

Related: **config** Display current configuration and possibly change it (M)
rfctype Type of rf channel (P)

rfwg **RF waveform generator (P)**

| Applicability: Not available on MERCURY and GEMINI 2000.

Description: Configuration parameter for whether a waveform generator board is present or not on the current rf channel. The value for each channel is set using the Waveform Generator label in the CONFIG window (opened by entering **config**).

Values: 'n' is setting for no waveform generator board on the channel (Not Present choice in CONFIG window).

'y' is setting for a waveform generation board on the channel (Present choice in CONFIG window).

See also: *VNMR and Solaris Software Installation*

Related: **config** Display current configuration and possibly change it (M)

right **Set display limits to right half of screen (C)**

Syntax: **right**

Description: Sets the horizontal control parameters, **sc** and **wc**, to produce a display (and subsequent plot) in the right portion of the screen (and page). For 2D data, space is left for the scales.

Alternate: Right button on 1D Display Size Selection Menu.

See also: *User Guide: Liquids NMR*

Related: **center** Set display limits for center of screen (C)
full Set display limits for a full screen (C)
fullt Set display limits for full screen with room for traces (C)
left Set display limits for left half of screen (C)
sc Start of chart (P)
wc Width of chart (P)

- rinput** **Input data for a regression analysis (M)**
- Syntax: `rinput`
- Description: Formats data for regression analysis and places the data into the file `regression.inp`. The program is interactive. If a `regression.inp` already exists, `rinput` starts by asking if you want to overwrite the file. Type `y` and press the Return key. It then asks for an x-axis title and a y-axis title. Enter the titles as asked (for no title, simply press Return). Next, `rinput` asks you to input the data in pairs. Separate each pair of values with a blank and press Return after the second value. At the end of the data set, press Return in response to the request for data. If you have another data set, type `y` and press Return to the question and then type in the data when it is asked for.
- See also: *User Guide: Liquids NMR; VNMR User Programming*
- Related: `expl` Display exponential or polynomial curves (C)
`poly0` Find mean of data in the file `regression.inp` (C)
- r1** **Set reference line in directly detected dimension (M)**
- Syntax: `r1<(frequency)>`
- Description: Sets the direct dimension reference line, taking into account any frequency scaling with the `scalesw` parameter.
- Arguments: `frequency` is a value, in Hz, to assign to the reference line. The default is the cursor position `cr`. To enter the value in ppm, add a `p` suffix.
- Examples: `r1`
`r1(0)`
`r1(7.2p)`
- See also: *Getting Started*
- Related: `cr` Current cursor position in directly detected dimension (P)
`cr1` Clear ref. line in directly detected dimension (C)
`reffrq` Reference frequency of the reference line (P)
`r11` Set ref. line in 1st indirectly detected dimension (M)
`r12` Set ref. line in 2nd indirectly detected dimension (M)
`scalesw` Scale spectral width in directly detected dimension (P)
- r11** **Set reference line in 1st indirectly detected dimension (M)**
- Syntax: `r11<(frequency)>`
- Description: Sets the first indirect dimension reference line, taking into account any frequency scaling with the `scalesw1` parameter.
- Arguments: `frequency` is a value, in Hz, to assign to the reference line. The default is the cursor position `cr1`. You can enter the suffixes `p`, `d`, or `k` to mean ppm, decoupler ppm, and kilo, respectively. These suffixes are exactly equivalent to using `*sfrq`, `*dfrq`, and `*1000`. Thus, if you are doing a 2D experiment in which the indirect axis is determined by the decoupler channel, you might enter, for example, `r11(10d)`, which is equivalent to `r11(10*dfrq)`.
- Examples: `r11`
`r11(0)`
`r11(7.2p)`
- See also: *User Guide: Liquids NMR*
- Related: `cr1` Cursor position in 1st indirectly detected dimension (P)
`cr11` Clear ref. line in 1st indirectly detected dimension (M)
`dfrq` Transmitter frequency of first decoupler (P)

<code>refpos2d</code>	Position of reference frequency in 1st indirect dimension (P)
<code>r1</code>	Set ref. line in directly detected dimension (M)
<code>r12</code>	Set ref. line in 2nd indirectly detected dimension (M)
<code>scalesw1</code>	Scale spectral width in 1st indirectly detected dimension (P)
<code>sfrq</code>	Transmitter frequency of observe nucleus (P)

`r12` **Set reference line in 2nd indirectly detected dimension (M)**

Applicability: All systems; however, although `r12` is available on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*, such systems can only process 3D data and cannot acquire 3D data.

Syntax: `r12<(frequency)>`

Description: Sets the second indirect dimension reference line, taking into account any frequency scaling with the `scalesw2` parameter.

Arguments: `frequency` is a value, in Hz, to assign to the reference line. The default is the cursor position `cr2`. You can enter the suffixes `p`, `d`, or `k` to mean ppm, decoupler ppm, and kilo, respectively. These suffixes are exactly equivalent to using `*sfrq`, `*dfrq`, and `*1000`. Because there is no suffix for the second decoupler (i.e., the third channel), to reference the third axis using `r12` you might enter (e.g., `r12(45*dfrq2)`).

Examples: `r12`
`r12(0)`
`r12(7.2p)`

See also: *User Guide: Liquids NMR*

Related:	<code>cr2</code>	Cursor position in 2nd indirectly detected dimension (P)
	<code>cr1</code>	Clear ref. line in directly detected dimension (C)
	<code>cr11</code>	Clear ref. line in 1st indirectly detected dimension (C)
	<code>cr12</code>	Clear ref. line in 2nd indirectly detected dimension (C)
	<code>dfrq</code>	Transmitter frequency of first decoupler (P)
	<code>dfrq2</code>	Transmitter frequency of second decoupler (P)
	<code>r1</code>	Set ref. line in directly detected dimension (M)
	<code>r11</code>	Set ref. line in 1st indirectly detected dimension (M)
	<code>scalesw2</code>	Scale spectral width in 2nd indirectly detected dimension (P)
	<code>sfrq</code>	Transmitter frequency of observe nucleus (P)

`rm` **Delete file (C)**

Syntax: `rm(file1<,file2,....>)`

Description: Removes one or more files from the file system, functioning like the UNIX command of the same name. Because it allows wildcard characters (`*` and `?`) in the command argument and recursive file deletion with the `-r` option, `rm` is very powerful. But it can be quite dangerous—without warning important files can be inadvertently deleted, even by experienced users. **Using `rm` to delete files in VNMR is not recommended.** The `delete` command is provided as a safer alternative.

Arguments: `file1, file2, ...` are names of files to delete.

See also: *Getting Started*

Related:	<code>delete</code>	Delete a file, parameter directory, or FID directory (C)
	<code>delexp</code>	Delete an experiment (C)
	<code>exists</code>	Determine if a parameter, file, or macro exists (C)
	<code>mv</code>	Move and/or rename a file (C)
	<code>rename</code>	Move and/or rename a file (C)

rmdir **Remove directory (C)**

Syntax: `rmdir(directory)`

Description: Removes one or more empty directories (i.e., directories without files).

Arguments: `directory` is the name of the directory to be removed.

Examples: `rmdir('/home/dan/temp')`

See also: *Getting Started*

Related:	<code>delete</code>	Delete a file, parameter directory, or FID directory (C)
	<code>dir</code>	List files in current directory (C)
	<code>lf</code>	List files in current directory (C)
	<code>ls</code>	List files in current directory (C)
	<code>mkdir</code>	Create new directory (C)

rmsAddData **Add transformed data files with weighting (U)**

Applicability: Systems with multiple receivers.

Syntax: `rmsAddData`

Description: This command is not normally executed directly by the user, but is called by the 'addrcvrs' macro.

Related: `addrcvrs` Combine data from multiple receivers (M)

ROESY **Change parameters for ROESY experiment (M)**

Syntax: `ROESY(<'GLIDE'>)`

Description: Converts the current parameter set to a ROESY experiment.

Arguments: 'GLIDE' is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding proton spectrum for the experiment.

Related: `roesy` Set up parameters for ROESY experiment (M)

roesy **Set up parameters for ROESY pulse sequence (M)**

Applicability: All systems except *GEMINI 2000*.

Syntax: `roesy(<ratio>)`

Description: Sets up a rotating frame Overhauser effect spectroscopy experiment.

Arguments: `ratio` is the value of the parameter `ratio` used in the sequence (`ratio` is not used in the ROESY sequence provided with *MERCURY-VX* and *MERCURY*).

Alternate: ROESY button in the 2D Pulse Sequence Setup Menu.

See also: *User Guide: Liquids NMR*

rof1 **Receiver gating time preceding pulse (P)**

Description: Sets the period of time in most pulse sequences when the receiver is gated off before each pulse. This allows the amplifier to fully turn on before the start of the pulse. Such gating is needed on all 500-MHz and 600-MHz systems, systems with wideline solids, and systems with the most recent AP Interface board (with parameter `apinterface` greater than 1). Such systems are configured with linear amplifiers that are normally “blanked” to give the best possible signal-to-noise (i.e., the amplifiers are turned off when the receiver is turned on). The ¹H/¹⁹F amplifiers have a short turn-on time, usually 1 to 5 μs following the removal of blanking by turning the receiver off. The low-frequency amplifier modules have a longer turn-on time, about 40 to 60 μs.

Values: 0 to 8190, in μs , typically 10 for $^1\text{H}/^{19}\text{F}$ and 40 for ^{31}P and lower frequency nuclei. On *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems, 10 is recommended for both the high and the low band.

See also: *Getting Started*

Related: `apinterface` AP Interface board type (P)
`rof2` Receiver gating time following pulse (P)

rof2 Receiver gating time following pulse (P)

Description: Sets the time after the final pulse in each pulse sequence that the receiver is gated off before acquisition begins. If “pulse breakthrough” effects are seen (a spike in the beginning of the FID), increasing `rof2` can reduce or eliminate the problem, particularly for low-frequency nuclei.

Values: 0 to 8190, in μs , typically 10. On *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems, 10 is recommended for both the high and low band.

See also: *Getting Started*

Related: `rof1` Receiver gating time preceding pulse (P)

rotate Rotate 2D data (C)

Syntax: `rotate<(number_degrees)>`

Description: Rotates a 2D spectrum. Both complex and hypercomplex 2D data will work.

Arguments: `number_degrees` is the amount of counter-clockwise rotation, in degrees. The default is 45.

See also: *User Guide: Liquids NMR*

Related: `foldcc` Fold INADEQUATE data about 2-quantum axis (C)
`foldj` Fold J-resolved 2D spectrum about $f1=0$ axis (C)
`foldt` Fold COSY-like spectrum along diagonal axis (C)

rotorsync Rotor synchronization (P)

Applicability: Systems with the solids rotor synchronization module.

Description: Configuration parameter that identifies if the system has the optional solids rotor synchronization module. The value of `rotorsync` is set using the Rotor Synchronization label in the CONFIG window (opened by entering `config`). Rotor synchronization requires either the Acquisition Controller board (Part No. 969204) or the Pulse Sequence Controller board (Part No. 992560) in the system.

Values: 1 is setting that system has solids rotor synchronization (Present choice in the CONFIG window).

0 is setting that system does not have solid rotor synchronization (Not Present choice in the CONFIG window).

See also: *VNMR and Solaris Software Installation*

Related: `config` Display current configuration and possibly change it (M)

rp Zero-order phase in directly detected dimension (P)

Description: Specifies the right phase-correction angles along the directly detected dimension according to

$$\text{absorption spectrum}(\omega) = \text{real channel}(\omega) * \sin \theta + \text{imaginary channel}(\omega) * \cos \theta$$

where the phase angle θ is a function of frequency:

$$\theta = \text{rp} + (\omega - \omega_0) * \text{lp}$$

ω_0 is defined as the right end of the spectrum. This dimension is referred to as the f_2 dimension in 2D data sets, f_3 dimension in 3D data sets, and so on.

Values: -360 to $+360$, in degrees.

See also: *Getting Started; User Guide: Liquids NMR*

Related:	aph	Automatic phase adjustment of spectra (C)
	aph0	Automatic phase of zero-order term (C)
	lp	First-order phase in directly detected dimension (P)
	rp1	Zero-order phase in 1st indirectly detected dimension (P)
	rp2	Zero-order phase in 2nd indirectly detected dimension (P)

rp1 Zero-order phase in 1st indirectly detected dimension (P)

Description: Specifies the right phase parameter along the first indirectly detected dimension, in degrees, for the f_1 dimension of a multidimensional data set during the process of phase-sensitive 2D transformation.

See also: *User Guide: Liquids NMR*

Related:	lp1	First-order phase in 1st indirectly detected dimension (P)
	rp	Zero-order phase in directly detected dimension (P)
	rp2	Zero-order phase in 2nd indirectly detected dimension (P)

rp2 Zero-order phase in 2nd indirectly detected dimension (P)

Description: Controls the zero-order phase constant along the second indirectly detected dimension during a **ds**, **dconi**, or equivalent display operation on the 2D data or a 1D trace therein. This dimension is often referred to as the f_2 dimension.

See also: *User Guide: Liquids NMR*

Related:	dconi	Interactive 2D contour display (C)
	ds	Display a spectrum (C)
	lp2	First-order phase in 2nd indirectly detected dimension (P)
	rp	Zero order phase in directly detected dimension (P)

rsliceplan Generate absolute magnet frame data (M)

Applicability: Systems with imaging capabilities.

Description: **rsliceplan** is a helper macro to **iplan** image planning. It combines the **iplan** data with sequence parameters to generate the absolute magnet frame data. Users without imaging capabilities should use **sliceplan**.

See also: *User Guide: Imaging*

Related:	iplan	Open interactive image planning tools (M)
	sliceplan	Set slice parameters for target slice (M)

rt Retrieve FIDs (M)

Syntax: `rt<(file<,'nolog'>)>`

Description: Retrieves FIDs from a file into the current experiment.

The `rt` macro does not copy the FID into the experiment. Instead, it links access to the original FID from the experiment. Most of the time, this behavior is desired, because the FID file is seldom changed. By making a link, disk space is also conserved. However, if the FID file in the experiment is written to, the data in the original file is also written to. It is best to make a copy of a FID file before altering it. The `makefid` command alters the FID file. The manual entry for `makefid` gives details on how to make a copy of the FID.

As another somewhat subtle point, because the FID in the experiment is a link to another `.fid` file, if that `.fid` file is removed, the link from the experiment may be gone. If you expect the FID in the experiment to be there, even if you delete the `.fid` file from where it was retrieved using `rt`, you should explicitly copy the file into the experiment.

Arguments: `file` is the name of the file that, with the suffix `.fid` added, contains the FIDs to be retrieved. The default is that the system prompts for the name (in that case, the name can be given without single quotes). If `file.fid` does not exist and `file.par` does, `rt` retrieves the parameters from `file.par`.

'nolog' is a keyword specifying that the log file is not to be retrieved.

Examples: `rt`
`rt('/vnmr/fidlib/fidld')`

See also: *Getting Started*

Related:	<code>fixpar</code>	Correct parameter characteristics in experiment (M)
	<code>makefid</code>	Make a FID element using numeric text input (C)
	<code>rtp</code>	Retrieve parameters (M)
	<code>rtv</code>	Retrieve individual parameters (C)
	<code>svf</code>	Save FIDs in current experiment (M)

rtcmx **Return Spinsight data into current experiment (C)**

Syntax: `rtcmx<(file)>`

Description: Retrieves Spinsight data into the current experiment.

Arguments: `file` is the name of the file. The default is that the macro prompts for the file name.

Alternate: Load button in the `files` program.

Examples: `rtcmx`
`rtcmx('redor.data')`

See also: *Getting Started*

Related: `files` Interactively handle files (C)

rtp **Retrieve parameters (M)**

Syntax: `rtp<(file)>`

Description: Retrieves parameters from a file into the current experiment.

Arguments: `file` is the name of the file that, with the suffix `.par` added, contains the parameters to be retrieved;. The default is that the system prompts for the name (in that case, the name can be given without single quotes). If `file.par` does not exist and `file.fid` does, `rtp` retrieves the parameters only from `file.fid`.

Examples: `rtp`
`rtp('/vnmr/stdpar/P31')`

See also: *Getting Started*

Related:	<code>fixpar</code>	Correct parameter characteristics in experiment (M)
	<code>rt</code>	Retrieve FIDs (M)
	<code>rtv</code>	Retrieve individual parameters (C)
	<code>svp</code>	Save parameters from current experiment (M)

rtphf **Return stored phasefile to current VNMR phasefile (C)**

Applicability: Systems with imaging capabilities.

Syntax: `rtphf (file)`

Description: Copies a stored phasefile (`curexp+ '/planes/file'`, where `file` is the file name given in the argument) into the phasefile of the current experiment (`curexp+ '/datdir/phasefile'`). This allows the display and manipulation of previously transformed images, provided the parameter values in the current experiment are compatible with the parameter values present in the experiment that generated the stored phasefiles at the time they were stored.

Arguments: `file` is the file name of the stored phase file. Use only relative path names for `file`, not absolute path names (i.e., use path names beginning with “/”).

Examples: `rtphf ('waldo')`

See also: *User Guide: Imaging*

Related:	<code>curexp</code>	Current experiment directory (P)
	<code>imcalc</code>	Calculate 2D phasefiles (M,U)
	<code>makephf</code>	Transform and save images as phasefiles (M)
	<code>svphf</code>	Save current VNMR phasefile (C)

rts **Retrieve shim coil settings (C)**

Syntax: `rts (file) <:status>`

Description: Locates a preexisting file of shim settings and copies the settings into the current parameter set of the current experiment and sets `load= 'y'` to facilitate subsequent loading of shims with `su` (or related commands or macros). If the shim file is not found, `rts` displays the file names it tried.

The `rts` command returns shims from a `.fid` file or a `.par` file, selecting the shim parameters from the parameters stored there.

Arguments: `file` is the name of a file containing the shim coil settings to be retrieved. If the file name is an absolute path, `rts` uses it with no modifications. Otherwise, `rts` searches up to three different directories, as follows:

- First, `rts` looks for a `shims` subdirectory in your VNMR user directory. If `shims` exists, it looks for the requested file name there.
- Next, if `shims` does not exist, `rts` then looks for the global parameter `shimspath`. If `shimspath` is present, it is expected to contain the name of a directory. If this directory exists, `rts` looks for the file in that directory.
- Finally, if this does not work, `rts` searches in the `shims` subdirectory of the VNMR system directory.

`status` is a return variable with one of the following values after `rts` finishes searching for the shim coil settings file:

- 0 indicates that `rts` failed to find requested file.
- 1 indicates that `rts` found the requested file, either as an absolute path or in the `shims` subdirectory of the VNMR user directory.

- 2 indicates that `rts` found the requested file using the global parameter `shimspath`.
- 3 indicates that `rts` found the requested file in `shims` subdirectory of the VNMR system directory.

Examples: `rts('acetone')`
`rts('bb10mm'):r1`

See also: *Getting Started*

Related: `load` Load status of displayed shims (P)
`shimspath` Path to user's `shims` directory (P)
`su` Submit a setup experiment to acquisition (M)
`svs` Save shim coil settings (C)

rtshims Extract shim parameter values (obsolete)

Description: The `rtshims` command is no longer in VNMR. It is replaced by the `rts` command.

Related: `rts` Retrieve shim coil settings (C)

rttmp Retrieve experiment data from experiment subfile (M)

Syntax: `rttmp(file)`

Description: Retrieves experiment data—parameters, FID, and transformed spectrum—from the file specified in a subdirectory inside `curexp+ '/subexp'`.

Arguments: `file` is the name of the subfile from which to retrieve the experiment data.

Examples: `rttmp('H1')`
`rttmp('cosy')`

See also: *Getting Started*

Related: `captain` Copy experiment data into experiment subfile (M)
`curexp` Current experiment directory (P)
`svtmp` Move experiment data into experiment subfile (M)

rtv Retrieve individual parameters (C)

Syntax: `rtv<(file,par1<,index1<,par2,index2...>>><:val>`

Description: Retrieves one or more parameters from a parameter file. The file might have been made with `svf` or `svp` or `sd` commands, or it might be from another experiment. If no return argument is added, the parameters are copied into the experiment's current tree. If the parameter does not already exist in the current tree, it is created. If the returned parameter is an array, the entire array is returned.

If a return argument is added, `rtv` returns values into the macro. This form of `rtv` command, in which values are passed only to macro variables, is useful if you do not want additional parameters created in the experiment's current tree.

Arguments: `file` is the name of the directory or a VNMR parameter file. If the supplied value for `file` is a directory (with or without the `.fid` or `.par` extension), the parameters are retrieved from the `procpa` file in that directory. If the supplied value does not correspond to a directory but rather is a VNMR parameter file, that file is used. The default is that `rtv` prompts for a file name. In that case, the file name can be given without single quotes.

`par1,index1,par2,index2,...` are the name and array index of one or more parameters to be retrieved. The default for each array index argument is

the first index. Including the array index for a parameter is only useful when returning values to the macro through a return argument.

`val` is a return argument for values to return to the macro.

Examples: `rtv`
`rtv('/vnmr/parlib/cosy.par', 'phase')`

See also: *Getting Started*

Related:	<code>rt</code>	Retrieve FIDs (M)
	<code>rtp</code>	Retrieve parameters (M)
	<code>sd</code>	Set first decoupler frequency to cursor position (M)
	<code>svf</code>	Save FIDs in current experiment (M)
	<code>svp</code>	Save parameters from current experiment (M)

S

s **Save display parameters as a set (M)**

Syntax: (1) `sset_number`
 (2) `s (set_number)`

Description: Saves a copy of the current values of all display parameters. The set is data-independent because the parameters that govern a display (`sp`, `wp`, `vs`, etc.) are saved but no data is saved.

Arguments: `set_number` is number of the display parameter set to be saved.

Examples: `s2`
`s (3)`

See also: *Getting Started*

Related: `fr` Full recall of display parameter set (M)
`r` Recall display parameter set (M)

s2pul **Set up parameters for standard two-pulse sequence (M)**

Syntax: `s2pul`

Description: Converts the current experiment to an experiment suitable for the standard two-pulse sequence (S2PUL).

Alternate: S2PUL button in the 1D Pulse Sequence Setup Menu.

See also: *Getting Started*

s2pulr **Set up parameters for standard 2-pulse sequence in “reverse” (M)**

Applicability: UNITY and VXR-S systems only.

Syntax: `s2pulr`

Description: Sets up a standard two-pulse sequence in “reverse” configuration (S2PULR). In this setup, the observe channel uses the decoupler hardware and is controlled by the parameters `dn` (which must be set to 'H1 '), `dof`, `dpwr` (or `dhp`), `p1`, and `pw`. The local oscillator (L.O.) signal must be taken from the decoupler board. No decoupling is supported in this sequence.

Note that the macros `movetof` and `movesw` cannot be used with S2PULR except in the following way: `tof=dof movetof` (or `movesw dof=tof`).

See also: *VNMR User Programming*

Related: `dhp` Decoupler high power with class C amplifier (P)
`dn` Nucleus for first decoupler (P)
`dof` Frequency offset for first decoupler (P)
`dpwr` Power level for first decoupler with linear amplifiers (P)
`movesw` Move spectral window according to cursors (M)
`movetof` Move transmitter offset (M)
`p1` First pulse width (P)
`pw` Pulse width (P)
`tof` Frequency offset for observe transmitter (P)

sa **Stop acquisition (C)**

Applicability: All systems; however, the `option` and `number` arguments are unavailable on *MERCURY* and *GEMINI 2000* systems.

Syntax: `sa<(option|number)>`

Description: Stops an experiment that has been submitted to acquisition. If experiment is active, it is stopped. Data is retained. `sa` applies to the experiment that you are joined to at the time the `sa` command is entered. Thus, if experiment 1 is active, you must be joined to experiment 1 for `sa` to stop that acquisition. If you are in experiment 2, entering `sa` has no effect on experiment 1.

When experiments are queued, the behavior of `sa` is more complex. If an experiment is active in `exp1` and queued in `exp2`, entering `sa` from `exp1` stops that experiment and immediately begins acquisition on `exp2`. Entering `sa` from `exp2`, on the other hand, removes `exp2` from the queue, without affecting the active experiment 1.

Entering `sa` from an experiment that is not active or queued has no effect.

Arguments: `option` is one of the following:

- `'eos'`, `'ct'`, `'scan'` are keywords to stop at the next `ct`.
- `'eob'`, `'bs'` are keywords to stop at the next block size.
- `'eof'`, `'nt'`, `'fid'` are keywords to stop at the next complete FID.
- `'eoc'`, `'il'` are keywords to stop at next complete `il` cycle (i.e., the latest block size that has been completed for all FIDs in interleave cycle).

`number` is an integer number to stop at the next `ct`, where the value of `ct` is a multiple of `number`. This is useful when you want to complete a phasecycle before stopping.

Examples: `sa`
`sa('ct')`
`sa(4)`

See also: *Getting Started*

Related:	<code>bs</code>	Block size (P)
	<code>ct</code>	Completed transients (P)
	<code>il</code>	Interleave arrayed and 2D experiments (P)
	<code>nt</code>	Number of transients (P)
	<code>ra</code>	Resume acquisition stopped with <code>sa</code> command (C)

sample **Submit change sample, Autoshim experiment to acquisition (M)**

Applicability: Systems with a sample changer.

Syntax: `sample`

Description: Performs the combined operations `change`, `spin`, `lock`, and `shim`, making it a convenient setup command for a new sample.

See also: *Getting Started*

Related:	<code>au</code>	Submit experiment to acquisition and process data (C)
	<code>change</code>	Submit a change sample experiment to acquisition (M)
	<code>ga</code>	Submit experiment to acquisition and FT the result (C)
	<code>go</code>	Submit experiment to acquisition (C)
	<code>lock</code>	Submit an Autolock experiment to acquisition (C)
	<code>shim</code>	Submit an Autoshim experiment to acquisition (C)
	<code>spin</code>	Submit a spin setup experiment to acquisition (C)
	<code>su</code>	Submit a setup experiment to acquisition (M)

savefile **Base file name for saving files (P)**

Applicability: Systems with LC-NMR accessory.

Description: Contains the base file name using the format `savefile.001`, `savefile.002`, etc., to which a series of FIDs or data sets are saved. If `savefile` does not exist, the `parlc` macro can create it.

See also: *User Guide: Liquids NMR*

Related: `parlc` Create LC-NMR parameters (M)

saveglobal **Save selected parameters from global tree (P)**

Description: Saves an array of parameter names from the global or systemglobal tree. Whenever `go` is executed, the parameters listed are saved in the current tree with an underscore (`_`) appended. These parameters are copied back into the global tree (without the underscore) whenever processing by `wbs`, `wnt`, `wexp`, or `werr` occurs.

See also: *User Guide: Liquids NMR*

Related: `go` Submit experiment to acquisition (C)
`loc` Location of sample in tray (P)

sb **Sinebell constant in directly detected dimension (P)**

Description: Applies a sinebell constant along the directly detected dimension. This dimension is often referred to as the f_2 dimension in 2D data sets, the f_3 dimension in 3D data sets, etc.

Values: A positive value applies a sinebell of the form $\sin\left(\frac{t \cdot \pi}{2 \cdot sb}\right)$
A negative value applies a squared sinebell function of form $\sin^2\left(\frac{t \cdot \pi}{2 \cdot sb}\right)$
`sb` is given in seconds. Typical value is `sb='n'`.

See also: *Getting Started*

Related: `sb1` Sinebell constant in 1st indirectly detected dimension (P)
`sb2` Sinebell constant in 2nd indirectly detected dimension (P)
`sbs` Sinebell shift constant in directly detected dimension (P)
`sine` Find values for a sine window function (M)
`sinebell` Select default parameters for sinebell weighting (M)
`sinesq` Find values for a sine squared window function (M)

sb1 **Sinebell constant in 1st indirectly detected dimension (P)**

Description: Applies a sinebell constant along the first indirectly detected dimension. This dimension is often referred to as the f_1 dimension in multidimensional data sets. `sb1` works analogously to the parameter `sb`. The “conventional” parameters, such as `lb` and `gf`, operate on the detected FIDs, while this “2D” parameter is used during processing of the interferograms.

Values: A positive value applies a sinebell of the form $\sin\left(\frac{t \cdot \pi}{2 \cdot sb1}\right)$
A negative value applies a squared sinebell function of form $\sin^2\left(\frac{t \cdot \pi}{2 \cdot sb1}\right)$
`sb1` is given in seconds. Typical value is `sb1='n'`.

See also: *User Guide: Liquids NMR*

Related: `sb` Sinebell constant in the directly detected dimension (P)
`sb2` Sinebell constant in 2nd indirectly detected dimension (P)

sb2 Sinebell constant in 2nd indirectly detected dimension (P)

Description: Applies a sinebell constant along the second indirectly detected dimension. This dimension is often referred to as the f_2 dimension in multidimensional data sets. `sb2` works analogously to the parameter `sb`. The value of `sb2` can be set with `wti` on the 2D interferogram data.

Values: A positive value applies a sinebell of the form $\sin\left(\frac{t \cdot \pi}{2 \cdot sb2}\right)$
 A negative value applies a squared sinebell function of form $\sin^2\left(\frac{t \cdot \pi}{2 \cdot sb2}\right)$
`sb2` is given in seconds. Typical value is `sb2= 'n'`.

See also: *User Guide: Liquids NMR*

Related: `sb` Sinebell constant in directly detected dimension (P)
`sb1` Sinebell constant in 1st indirectly detected dimension (P)
`wti` Interactive weighting (C)

sbs Sinebell shift in directly detected dimension (P)

Description: Working in combination with the parameter `sb`, `sbs` allows shifting the origin of the sinebell function along the directly detected dimension. This dimension is often referred to as the f_2 dimension in 2D data sets, the f_3 dimension in 3D data sets, etc.

Values: The origin is shifted according to the formula $\sin\left(\frac{(t - sbs) \cdot \pi}{2 \cdot sb}\right)$
 The square of this function is applied if `sb` is negative. `sbs` is given in seconds. The typical value is `sbs= 'n'`.

See also: *Getting Started*

Related: `sb` Sinebell constant in directly detected dimension (P)
`sbs1` Sinebell shift in 1st indirectly detected dimension (P)
`sbs2` Sinebell shift in 2nd indirectly detected dimension (P)
`sine` Find values for a sine window function (M)
`sinesq` Find values for a sine squared window function (M)

sbs1 Sinebell shift in 1st indirectly detected dimension (P)

Description: Working in combination with the parameter `sb1`, `sbs1` allows shifting the origin of the sinebell function along the first indirectly detected dimension. This dimension is often referred to as the f_1 dimension in multidimensional data sets. `sbs1` works analogously to parameter `sbs`. The “conventional” parameters, such as `lb` and `gf`, operate on the detected FIDs, while this “2D” parameter is used during processing of the interferograms.

Values: The origin is shifted according to the form $\sin\left(\frac{(t - sbs1) \cdot \pi}{2 \cdot sb1}\right)$
 The square of this function is applied if `sb1` is negative. `sbs1` is given in seconds. The typical value is `sbs1= 'n'`.

See also: *User Guide: Liquids NMR*

Related: `sb1` Sinebell constant in 1st indirectly detected dimension (P)
`sbs` Sinebell shift constant in directly detected dimension (P)
`sb2` Sinebell constant in 2nd indirectly detected dimension (P)

sbs2 Sinebell shift in 2nd indirectly detected dimension (P)

Description: Working in combination with the parameter `sb2`, `sbs2` allows shifting the origin of the sinebell function along the second indirectly detected dimension. This dimension is often referred to as the f_2 dimension in multidimensional data

sets. `sbs2` works analogously to parameter `sbs`. `sbs2` can be set with `wti` on the 2D interferogram data.

Values: The origin is shifted according to the formula $\sin\left(\frac{(t - sbs2) \cdot \pi}{2 \cdot sb2}\right)$. The square of this function is applied if `sb2` is negative. `sbs2` is given in seconds. The typical value is `sbs2='n'`.

See also: *User Guide: Liquids NMR*

Related: `sbs` Sinebell shift constant in directly detected dimension (P)
`sb2` Sinebell constant in 2nd indirectly detected dimension (P)
`wti` Interactive weighting (C)

sc Start of chart (P)

Description: Positions of the start of the plotting position (the “chart”) with respect to the right edge of the plotter.

Values: 0 to `wcmax`, in mm

See also: *Getting Started; User Guide: Liquids NMR*

Related: `sc2` Start of chart in second direction (P)
`wc` Width of chart (P)
`wcmax` Maximum width of chart (P)

sc2 Start of chart in second direction (P)

Description: Controls the start of plotting position of the second axis (or y axis) of a 2D contour plot. The parameter `wc2` controls the width of the chart.

Values: 0 to `wc2max`, in mm.

See also: *User Guide: Liquids NMR*

Related: `sc` Start of chart (P)
`wc2` Width of chart in second direction (P)
`wc2max` Maximum width of chart in second direction (P)

scalelimits Set limits for scales in regression (M)

Syntax: `scalelimits(x_start,x_end,y_start,y_end)`

Description: Causes the command `expl`, which is used by regression to display data, to use typed-in scale limits. The limits are retained as long as an `expl` display is retained.

Arguments: `x_start,x_end,y_start,y_end` are x-axis and y-axis starting and ending limits. The default is that `scalelimits` prompts for the limits.

See also: *User Guide: Liquids NMR, VNMR User Programming*

Related: `autoscale` Resume autoscaling after limits set by `scalelimits` (M)
`expl` Display exponential or polynomial curves (C)

scalesw Set scaling factor for multipulse experiments (M)

Syntax: `scalesw`

Description: Sets the spectral width scaling factor for the multipulse sequences set up by macros `br24` and `mrev8`. The value of the scaling factor is stored in the parameter `scalesw`.

See also: *User Guide: solid-State NMR*

Related:	<code>br24</code>	Set up BR24 multiple pulse experiment (M)
	<code>mrev8</code>	Set up MREV8 multiple pulse experiment (M)
	<code>scalesw</code>	Scale spectral width in directly detected dimension (P)
	<code>scalesw1</code>	Set f_1 scaling factor for 2D multipulse experiments (M)

`scalesw` **Scale spectral width in directly detected dimension (P)**

Description: Adjusts the frequency scale dimension used with the parameter sets in the sequences set up by the `br24`, `mrev8`, `ssecho`, and `xpolar` macros. If `scalesw` is active, the labels for the frequency scales includes the letters `sc` in parentheses. A scaled frequency can be referenced using the `r1` macro.

Values: 'n', number greater than 0.0

See also: *User Guide: Solid-State NMR*

Related:	<code>br24</code>	Set up BR24 multiple pulse experiment (M)
	<code>mrev8</code>	Set up MREV8 multiple pulse experiment (M)
	<code>r1</code>	Set reference line (M)
	<code>scalesw</code>	Set scaling factor for multipulse experiments (M)
	<code>scalesw1</code>	Scale spectral width in 1st indirectly detected dimension (P)
	<code>scalesw2</code>	Scale spectral width in 2nd indirectly detected dimension (P)
	<code>ssecho</code>	Set up solid-state echo pulse sequence (M)
	<code>xpolar</code>	Set up parameters for XPOLAR pulse sequence (M)

`scalesw1` **Set f_1 scaling factor for 2D multipulse experiments (M)**

Syntax: `scalesw1`

Description: Sets the f_1 spectral width scaling factor for the multipulse sequences set up by the `br24` and `mrev8` macros. The value of the scaling factor is stored in the parameter `scalesw1`.

See also: *User Guide: Solid-State NMR*

Related:	<code>br24</code>	Set up BR-24 multiple pulse experiment (M)
	<code>mrev8</code>	Set up MREV8 multiple pulse experiment (M)
	<code>scalesw1</code>	Scale spectral width in 1st indirectly detected dimension (P)

`scalesw1` **Scale spectral width in 1st indirectly detected dimension (P)**

Description: Analogous to the `scalesw` parameter except that `scalesw1` applies to first indirectly detected dimension of a multidimensional data set. A scaled frequency along this dimension can be referenced using the `r11` macro.

Values: 'n', number greater than 0.0

See also: *User Guide: Solid-State NMR*

Related:	<code>r11</code>	Set reference line in 1st indirectly detected dimension (M)
	<code>scalesw</code>	Scale spectral width in directly detected dimension (P)
	<code>scalesw1</code>	Set f_1 scaling factor for 2D multipulse experiments (M)
	<code>scalesw2</code>	Scale spectral width in 2nd indirectly detected dimension (P)

`scalesw2` **Scale spectral width in 2nd indirectly detected dimension (P)**

Description: Analogous to the `scalesw` parameter except `scalesw2` applies to second indirectly detected dimension of a multidimensional data set. A scaled frequency along this dimension can be referenced using the `r12` macro.

Values: 'n', number greater than 0.0

See also: *User Guide: Solid-State NMR*

Related: **r12** Set reference line in 2nd indirectly detected dimension (M)
scalesw Set scaling factor for multipulse experiments (M)
scalesw1 Set f_1 scaling factor for 2D multipulse experiments (M)

sd Set first decoupler frequency to cursor position (M)

Syntax: **sd**

Description: Sets the first decoupler frequency offset parameter **dof** to place the first decoupler at the cursor position in the spectrum. This works only if the transmitter nucleus and first decoupler nucleus are the same (**tn=dn**).

See also: *Getting Started*

Related: **dof** Frequency offset for first decoupler (P)
dn Nucleus of first decoupler (P)
sd2 Set second decoupler frequency to cursor position (M)
sd3 Set third decoupler frequency to cursor position (M)
sda Set first decoupler frequency array (M)
tn Nucleus for observe transmitter (P)

sd2 Set second decoupler frequency to cursor position (M)

Applicability: Systems with a second decoupler.

Syntax: **sd2**

Description: Sets the second decouple frequency offset parameter **dof2** to place the second decoupler at the cursor position in the spectrum. This works only if the transmitter nucleus and second decoupler nucleus are the same (**tn=dn2**).

See also: *Getting Started*

Related: **dn2** Nucleus for second decoupler (P)
dof2 Frequency offset for second decoupler (P)
sd Set first decoupler frequency to cursor position (M)
sd2a Set second decoupler frequency array (M)
tn Nucleus for observe transmitter (P)

sd3 Set third decoupler frequency to cursor position (M)

Applicability: Systems with a third decoupler.

Syntax: **sd3**

Description: Sets the third decoupler frequency offset parameter **dof3** to place the third decoupler at the cursor position in the spectrum. This works only if the transmitter nucleus and third decoupler nucleus are the same (**tn=dn3**).

See also: *Getting Started*

Related: **dn3** Nucleus for third decoupler (P)
dof3 Frequency offset for third decoupler (P)
sd Set first decoupler frequency to cursor position (M)
sd3a Set third decoupler frequency array (M)
tn Nucleus for observe transmitter (P)

sda Set first decoupler frequency array (M)

Syntax: **sda**

Description: Sets up an array of offset values for the first decoupler, using `sd` for the first decoupler position and `sda` for subsequent positions. This works only if the transmitter nucleus and first decoupler nucleus are the same (`tn=dn`).

See also: *Getting Started*

Related: `dn` Nucleus for first decoupler (P)
`sd` Set first decoupler frequency to cursor position (M)
`sd2a` Set frequency array for second decoupler (M)
`sd3a` Set frequency array for third decoupler (M)
`tn` Nucleus for observe transmitter (P)

sd2a Set second decoupler frequency array (M)

Applicability: Systems with a second decoupler.

Syntax: `sd2a`

Description: Sets up an array of offset values for the second decoupler, using `sd2` for the first position and `sd2a` for subsequent positions. This works only if the transmitter nucleus and second decoupler nucleus are the same (`tn=dn2`).

See also: *Getting Started*

Related: `dn2` Nucleus for second decoupler (P)
`sd2` Set second decoupler frequency to cursor position (M)
`sda` Set first decoupler frequency array (M)
`tn` Nucleus for observe transmitter (P)

sd3a Set third decoupler frequency array (M)

Applicability: Systems with a third decoupler.

Syntax: `sd3a`

Description: Sets up an array of offset values for the third decoupler, using `sd3` for the first position and `sd3a` for subsequent positions. This works only if the transmitter nucleus and third decoupler nucleus are the same (`tn=dn3`).

See also: *Getting Started*

Related: `dn2` Nucleus for third decoupler (P)
`sd3` Set third decoupler frequency to cursor position (M)
`sda` Set first decoupler frequency array (M)
`tn` Nucleus for observe transmitter (P)

sdp Show diffusion projection (M)

Syntax: `sdp`

Description: Displays projection onto diffusion axis using the `dsp` facility. Use with 2D or 3D DOSY data after DOSY analysis. The unit of the resulting axis is D (10^{-10} m²/sec). Because `sdp` overwrites the parameters in the current experiment, use it in only an experiment in which it is okay for existing data to be overwritten.

See also: *User Guide: Liquids NMR*

Related: `dosy` Process DOSY experiments (M)

sediff Set up spin-echo diffusion imaging sequence (M)

Applicability: Systems with imaging capabilities.

Syntax: `sediff`

Description: Defines the excitation band from the position of cursors in the graphics window and reports them to user. It also sets `r1` to excitation bandwidth and `r2` to offset. `selex` is part of the Pbox software environment and uses the Pbox macros `pbox_bw` and `putwave`.

Arguments: `sh` is the name of a shape file.

`pw` is the pulsewidth, in sec.

`st` is the spin status: 0 for excitation, 0.5 for refocusing, or 1 for de-excitation.

`ph` is the phase (or phase cycle, see `wavelib/supercycles`).

`fla` is the flip angle.

`trev` is the time reversal. This argument can be used to cancel time reversal introduced by setting the spin status (`st`) to 1 for de-excitation.

Examples: `selex`
`selex('esnob', 0.0, 1, 90.0)`

See also: *User Guide: Liquids NMR*

Related: [Pbox](#) Pulse shaping software (U)

selexcit **Set up PFG selective excitation pulse sequence (M)**

Applicability: Systems with a pulsed field gradient module. Not available on *MERCURY-VX*, *MERCURY* and *GEMINI 2000* systems.

Syntax: `selexcit`

Description: Prepares an experiment for PFG (pulsed field gradient) selective excitation, with presaturation option.

See also: *User Guide: Liquids NMR*

sems **Set up basic imaging sequence with oblique capability (M)**

Applicability: Systems with imaging capabilities.

Syntax: `sems`

Description: Sets up a standard multislice spin-echo imaging sequence with oblique imaging capability.

See also: *User Guide: Imaging*

send2vnmr **Send a command to VNMR (U)**

Syntax: `send2Vnmr $vnmruser/.talk command`

Description: Sends a command from UNIX to VNMR using the port number stored in the `$vnmruser/.talk` file. This file is created when the macro `listenon` is entered on the VNMR command line.

Arguments: `command` is any character string (commands, macros, or if statements) normally typed into the VNMR command line.

Examples: `send2Vnmr $vnmruser/.talk dg`

See also: *VNMR User Programming*

Related: [bootup](#) Macro executed automatically when VNMR activated (M)
[listenon](#) Enable receipt of messages from `send2Vnmr` (M)
[listenoff](#) Disable receipt of messages from `send2Vnmr` (M)

seqcon Acquisition loop control (P)

Applicability: Systems with imaging capabilities.

Description: Controls the status of various looping processes used during sequence acquisition. The `nD`, `seqcon`, `plist`, `patlist`, `pwrlist`, `fliplist` and `sslist` parameters configure a particular parameter set for an application sequence defined by the value of the `seqfil` parameter.

Values: String with five characters, consisting of the characters 'n', 's', and 'c', that control where and when the looping occurs:

- 'n' (null loop) specifies a sequence that has no such loop function.
- 's' (standard loop) sets the looping operation to occur during the execution of pulse sequence generation in the host computer. Each loop execution generates a new acode set for execution in the acquisition computer. Each acode set will ultimately give rise to its own data block in the FID file. A standard loop operation therefore lies outside the signal averaging (transient counter loop). Parameter arrays and use of the 2D implicit loop are standard loops. The `multiecho` loop *cannot* be a standard loop.
- 'c' (compressed loop) sets the looping operation to occur dynamically in the acquisition computer, and each loop execution generates a new data “trace” within the current data “block”. This requires space in the on-board HAL memory. Compressed loops lie inside the signal averaging loop.

Each character position has place value and thus affects a different looping operation:

- First character: `multiecho` looping.
- Second character: `multislice` looping.
- Third character: 2D phase encode loop.
- Fourth character: 3D phase encode loop.
- Fifth character: 4D phase encode loop.

For example, `seqcon='ncsnn'` is 2D imaging with compressed multislice.

See also: *User Guide: Imaging*

Related:	<code>fliplist</code>	Standard flip angle list (P)
	<code>nD</code>	Application dimension (P)
	<code>patlist</code>	Active pulse template parameter list (P)
	<code>plist</code>	Active pulse length parameter list (P)
	<code>pwrlist</code>	Active pulse power level parameter list (P)
	<code>seqfil</code>	Acquisition object code name (P)
	<code>sslist</code>	Conjugate gradient list (P)

seqfil Pulse sequence name (P)

Description: Identifies the name of the pulse sequence to be used. The value of `seqfil` is displayed on the top line of the screen after the “Seq:” label. Macros used to set up new pulse sequences, such as `dept` and `apt`, automatically change the `seqfil` parameter.

See also: *Getting Started*

Related:	<code>pslabel</code>	Pulse sequence label (P)
----------	----------------------	--------------------------

seqgen **Initiate compilation of user's pulse sequence (M,U)**

Syntax: (From VNMR) `seqgen(<-static,>file<.c>)`
 (From UNIX) `seqgen <-static> file<.c> <file1,...>`

Description: Begins compilation of a user pulse sequence. When used from VNMR, the VNMR macro `seqgen` calls the UNIX shellscrip `seqgen`, which can also be called directly from UNIX, as shown above. The `seqgen` shellscrip then calls the compilation makefile `seqgenmake`, located in the directory `/vnmr/acqbin`.

The specified pulse sequence can be located in `~/vnmrsys/psglib` or in `/vnmr/psglib`. If two files with the same name exist in these two directories, the local directory (`~/vnmrsys/psglib`) takes precedence. For sequences in `/vnmr/psglib`, `seqgen` first copies the file into the local directory `~/vnmrsys/psglib` and then compiles it there; the resulting executable is then placed in `~/vnmrsys/seqlib`. A copy of the pulse sequence is also copied into the `seqlib` directory along with the executable. As it is running, `seqgen` reports where it found the specified sequence(s).

`seqgen` uses library files (object modules) found in `/vnmr/lib`. If `setuserpsg` and `psggen` has been run, the library files in the local directory `~/vnmrsys/psg` take precedence of those in `/vnmr/lib`.

Error messages are written into the file `file.errors`, where `file` is the name of the pulse sequence in `psglib` in which compilation is performed.

Note that `seqgen` not only accepts file names with and without extensions, but also accepts files specified with wildcards and complex paths (`seqgen` strips the directory part, and `seqgen /vnmr/psglib/apt` will compile `~/vnmrsys/psglib/atp.c` if it exists).

Arguments: `-static` is a keyword for `seqgen` to use static rather than dynamic binding. Static binding results in larger executables in `seqlib` (several hundred Kbytes), but these sequences execute slightly faster (i.e., the `go` command). While insignificant generally, faster execution is helpful in some special applications such as the Scout Scan™ mode of LC-NMR, where the time spent on the `go` command becomes critical. Static binding results in a fixed-size time gain, regardless of the number of increments; for large multidimensional experiments, the speed difference is not noticeable.

`file` is the file name of a standard two-pulse sequence.

`.c` is the extension on the file name.

`file1, file2, ...` are the names of files containing more sequences.

Examples: (From VNMR) `seqgen('/vnmr/psglib/*.c')`
 (From UNIX) `seqgen /vnmr/psglib/*.c`
 (From UNIX) `seqgen apt dept noesy`
 (From UNIX) `seqgen -static lcld`

See also: *VNMR User Programming*

Related: `go` Submit experiment to acquisition (M)
`psggen` Compile a user PSG object library (M,U)

set2D **General setup for 2D experiments (M)**

Syntax: `set2D<(F2_dig_res<,F1_dig_res>)>`

Description: Similar to `set2d` but does not execute `par2d` and does not make `sw1`, `rf11`, and `rfp1` decisions based on `tn=dn` condition.

Arguments: `F2_dig_res` is the f_2 digital resolution desired, in Hz/pt. Default is 6.

`F1_dig_res` is the f_1 digital resolution desired, in Hz/pt. Default is 12.

Related:	<code>rf11</code>	Reference peak position in 1st indirectly detected dimension (P)
	<code>rfp1</code>	Reference peak frequency in 1st indirectly detected dimension (P)
	<code>set2d</code>	General setup for 2D experiments (M)
	<code>sw1</code>	Spectral width in 1st indirectly detected dimension (P)

set2d **General setup for 2D experiments (M)**

Syntax: `set2d(experiment< , F2_dig_res< , F1_dig_res>>)`

Description: Runs the macro `par2d` to create new parameters needed for 2D experiments, then selects starting values for a number of parameters. The `set2d` macro is “internal” and not normally typed directly by the user.

Arguments: `experiment` is the name of a 2D experiment (e.g., 'noesy').

`F2_dig_res` is the f_2 digital resolution desired, in Hz/pt.

`F1_dig_res` is the f_1 digital resolution desired, in Hz/pt.

Examples: `set2d('cosyps')`
`set2d('hetcor' , 16)`
`set2d('het2dj' , 16 , (2*sw1) / fn1)`

See also: *User Guide: Liquids NMR*

Related: `par2d` Create 2D acquisition parameters (M)

set3dproc **Set 3D processing (C)**

Syntax: `set3dproc((< 'nocoeff' >< , directory>>)`

Description: Creates the file `procdat` that contains binary 3D information used by `ft3d` in processing the 3D FID data. It also creates the 3D parameter set `procp3d` that is used by the `select` command to display the 2D planes from the 3D transformed data. `set3dproc` can only create the proper 3D coefficient file if the parameters `phase` and `phase2` are used to generate States-Haberhorn (hypercomplex) or TPPI data along the t_1 and t_2 dimensions.

`set3dproc` creates the coefficient file for the following five values of `array` (where SH is States-Haberhorn):

- if `array` = ' ' (null string), type of 3D data is TPPI(t_1) – TPPI(t_2)
- if `array` = 'phase', type of 3D data is SH(t_1) – TPPI(t_2)
- if `array` = 'phase2', type of 3D data is SH(t_2) – TPPI(t_1)
- if `array` = 'phase2 , phase', type of 3D data is SH(t_1) – SH(t_2)

If `array` is set to some other value, `set3dproc` cannot create the 3D coefficient file and an error is reported within VNMR.

Arguments: 'nocoeff' is a keyword that the 3D coefficient file `coef` is not to be created.

`directory` is the name of the directory for `procdat` and `procp3d`. The default is the subdirectory `info` in the directory `curexp`.

Examples: `set3dproc`
`set3dproc('nocoeff' , 'curexp/info3d')`

See also: *User Guide: Liquids NMR*

Related:	<code>array</code>	Parameter order and precedence (P)
	<code>ft3d</code>	Perform a 3D Fourier transform (M,U)
	<code>phase</code>	Phase selection (P)
	<code>phase2</code>	Phase selection for 3D acquisition (P)

`select` Select a spectrum or 2D plane without displaying it (C)
`wftt3` Process f_3 dimension during 3D acquisition (M)

setallshims Set all shims into hardware (M)

Syntax: `setallshims`

Description: Sets shims from the current parameter tree into hardware. `setallshims` is equivalent to entering `load='y'su` but without setting all the hardware parameters normally set by `su` (temperature, decoupling, transmitter initialization, etc.). The shims used depend on the `shimset` configuration. For the shim set on the Ultra•nmr shim system, `setallshims` is active only if hardware-to-software shim communication is enabled.

See also: *Getting Started*

Related: `load` Load status of displayed shims (P)
`readallshims` Read all shims from hardware (M)
`readhw` Read current values of acquisition hardware (C)
`sethw` Set values for hardware in acquisition system (C)
`shimset` Type of shim set (P)
`su` Submit a setup experiment to acquisition (M)

setarray Set up a parameter array (M)

Applicability: Systems with imaging capabilities.

Syntax: `setarray<(name, start, step, elements)>`

Description: Sets up an array of a numeric acquisition parameter in single-arrayed experiments.

Arguments: `name` is the name of the parameter to be arrayed. The default (not entering any arguments) is the system prompts for the argument values.

`start` is the starting value for the array.

`step` is the step value for the array.

`elements` is the number of elements in the array.

Examples: `setarray`
`setarray('d1', 1, 1, 10)`

See also: *User Guide: Imaging*

setcenter Set up parameters for center sequence calibration (M)

Applicability: Systems with imaging capabilities.

Syntax: `setcenter`

Description: Loads parameter sets for center sequence calibration during imaging installation.

See also: *User Guide: Imaging*

setcolor Set colors for graphics window and for plotters (C)

Applicability: `UNITYINOVA` and `MERCURY-VX` and `MERCURY` systems

Syntax: (1) `setcolor('pcl', item_index, 'color')`
(2) `setcolor('hpgl', item_index, 'color')`
(3) `setcolor('pen', pen_number, 'color')`
(4) `setcolor('graphics', item_index, red, green, blue)`

```
(5) setcolor('ps', item_index, red, green, blue)
(6) setcolor('plotter', black_plane, color_planes)
```

Description: Sets colors used on the graphics window and on plotters. This command is a utility program used by the `color` macro and other macros. It is not expected that `setcolor` would be entered directly from the input window.

Arguments: `'pcl'` is a keyword to set colors on a plotter device that uses the PCL language. PCL plotters are the laser type of plotter.

`'hpgl'` is a keyword to set colors on a plotter device that uses the HPGL language. HPGL plotters are the pen type of plotter.

`'pen'` is a keyword that next two arguments set the color for a physical pen on a plotter device that uses the HPGL language.

`'graphics'` is a keyword to set colors on the graphics window.

`'ps'` is a keyword to set colors on a plotter using the PostScript language.

`red`, `green`, `blue` are three integers between 0 and 255 that set the amount of red, green, and blue color on the graphics window or PostScript plotter.

`'plotter'` is a keyword that the next two arguments set the black mode and number of colors available for a plotter device.

`item_index` is an index number from the following list that represents a specific drawing item.

8	background of images
9	real channel of an FID
10	imaginary channel of an FID
11	spectrum
12	integral
13	parameters
14	scale
15	threshold line (graphics device only)
16	second spectrum or FID in <code>addi</code> (graphics device only)
17	result spectrum or FID in <code>addi</code> (graphics device only)
18	cursors (graphics device only)
19	foreground of images
20	background color of graphics window (graphics device only)
20-35	contour 0 to contour 15 of absolute value 2D display
36-42	contours -7 to -1 of phased 2D display
44-50	contours 1 to 7 of phased 2D display

`pen_number` is an integer from 1 to 8 that specifies the physical pen used.

`color` is a string for the color set for the device: `'red'`, `'green'`, `'blue'`, `'cyan'`, `'magenta'`, `'yellow'`, `'white'`, or `'black'`.

`black_plane` is 1 or 0, specifying whether the plotter has a separate black mode. Because all currently supported plotters have this feature, the value is usually 1.

`color_planes` specifies how many colors are available. Use 3 for color plotters and 0 for black and white plotters.

Examples:

```
setcolor('pcl', 11, 'green')
setcolor('hpgl', 11, 'red')
setcolor('pen', 2, 'red')
```

```
setcolor('graphics',11,255,0,0)
setcolor('ps',11,255,255,0)
setcolor('plotter',1,0)
```

See also: *Getting Started*

Related: `addi` Start interactive add/subtract mode (C)
`color` Select plotting colors from a graphical interface (M)

setdecpars Set decoupler parameter values from probe file (M)

Syntax: `setdecpars`

Description: Reads from the probe file `pwx1vl`, `pwx`, `pplvl`, `pp`, `dpwr`, `dmf`, `dmm`, `dres`, and `dseq` values, if they exist, and updates the current experiment parameters.

Related: `setdec2pars` Set decoupler 2 parameter values from probe file (M)

setdec2pars Set decoupler 2 parameter values from probe file (M)

Syntax: `setdec2pars`

Description: Reads from the probe file `pwx2vl`, `pwx2`, `dpwr2`, `dmf2`, `dmm2`, `dres2`, and `dseq2` values, if they exist, and updates the current experiment parameters.

Related: `setdecpars` Set decoupler parameter values from probe file (M)

setdgroup Set the Dgroup of a parameter in a tree (C)

Syntax: `setdgroup(parameter, dgroup<, tree>)`

Description: Sets the Dgroup of a parameter in a tree. The application determines the usage of `setdgroup`. Only Tcl-dg currently uses this feature.

Arguments: `parameter` is the name of the parameter.

`dgroup` is an integer.

`tree` is 'current', 'global', 'processed', or 'systemglobal'. The default is 'current'. Refer to the description of the `create` command for more information on types of trees.

Examples: `setdgroup('a',1)`
`setdgroup('b',3,'global')`

See also: *VNMR User Programming*

Related: `create` Create new parameter in a parameter tree (C)

setenumeral Set values of a string parameter in a tree (C)

Syntax: `setenumeral(parameter, N, enum1, enum2, ..., enumN<, tree>)`

Description: Sets the possible values of a string parameter in a parameter tree. To remove enumerated values from a parameter, set argument `N` to 0 (see example below).

Arguments: `parameter` is the name of the parameter.

`N` is the number of enumeral values to be assigned to `parameter` (or removed from `parameter` if `N` is set to 0).

`enum1` to `enumN` are the possible string values of the parameter.

`tree` is 'current', 'global', 'processed', or 'systemglobal'. The default is 'current'. Refer to the description of the `create` command for more information on types of trees.

Examples: `setenumeral('size',0)`
`setenumeral('size',2,'large','small')`
`setenumeral('user',3,'user','superuser','master',`
`'global')`

See also: *VNMR User Programming*

Related: `create` Create new parameter in a parameter tree (C)

setether Connect or reconnect host computer to Ethernet (U)

Syntax: `setether`

Description: Connects or reconnects the host computer to the Ethernet network. Only `root` can execute this shellscript properly. If the system is already connected to the Ethernet network, `setether` does nothing.

On systems running Solaris, `setether` undoes the work of `setnoether`. You cannot use `setether` unless you previously entered the `setnoether` command. `setether` restores the files `hostname.le0`, `defaultdomain`, and `defaultrouter` so that Ethernet is activated on the host computer when UNIX is rebooted.

See also: *VNMR and Solaris Software Installation*

Related: `setnoether` Disconnect host computer from Ethernet (U)

setflip Set rf power levels to desired flip angle (M)

Applicability: Systems with imaging capabilities.

Syntax: `setflip(name,patname,pwrname,flip)`

Description: Sets up the rf power levels for a given pulse to obtain a desired flip angle. Power levels are calculated from the calibration data for a square pulse. The calibration data should be located in the file `pulsecal`, which should reside in the `vnmr/sys` directory. The macro `setflip` also looks for the `pulsecal` file in the system directory.

Arguments: `name` is the name of the pulse parameter.

`patname` is the name of the pattern parameter.

`pwrname` is the name of the power parameter.

`flip` is the flip angle, in degrees.

Examples: `setflip('pw','pwpat','tpwr',90)`

See also: *User Guide: Imaging*

Related: `pulsecal` Update and display pulse calibration data file (M)

setfrq Set frequency of rf channels (C)

Syntax: `setfrq<(channel)><('nucleus')>`

Description: Calculates frequencies based on the nucleus (`tn`, `dn`, `dn2`, etc.), referencing (`lockfreq`), solvent, and the offset parameter (`tof`, `dof`, etc.). The result of the calculation is stored in parameters `sfrq`, `dfrq`, `dfrq2`, etc. The parameters are rounded to the resolution of the channel—either 0.1 Hz or 100 Hz (either 0.1 Hz or 0.0745... Hz on *GEMINI 2000* systems).

The `setfrq` command should never need to be entered from the keyboard. It is called automatically when the appropriate parameters are changed or a parameter set is returned. If a parameter is entered that affects a single frequency, `setfrq` is called from an internal underscore macro (e.g., `_tn`,

`_tof`, `_dn`, `_dof`) to recalculate the frequency for that channel. Likewise, if a parameter is entered that affects all frequencies, `setfrq` is called from an internal underscore macro (e.g., `_solvent`, `_lockfreq`) to recalculate the frequencies.

Arguments: `channel` is a single integer specifying the rf channel to be set. The default is to calculate the frequencies for all rf channels.

`nucleus` displays or returns the frequency of the supplied nucleus. Channel 1 is assumed for rounding information and an offset (e.g., `tof` or `dof`) is not added to the result.

Examples: `setfrq`
`setfrq(2)`
`setfrq('P31'):freq`

See also: *Getting Started*

Related: `spcfrq` Display frequencies of rf channels (M)

setgauss Set a Gaussian fraction for lineshape (M)

Syntax: (1) `setgauss(fraction)`
 (2) `setgauss(fraction*)`

Description: Modifies the output of a deconvolution using pure Lorentzian lineshape (`fitspec.outpar`) and makes it the input for a subsequent analysis (`fitspec.inpar`), after first modifying the Gaussian fraction. To allow this fraction to vary, use syntax 1; to fix the fraction, use syntax 2.

Arguments: `fraction` is the Gaussian fraction of the lineshape, a number from 0 to 1. To fix the fraction (syntax 2), suffix the value with an asterisk (*) and enclose the value in single quotes (see the second example below).

Examples: `setgauss(0.4)`
`setgauss('1.0*')`

See also: *User Guide: Liquids NMR*

Related: `fitspec` Perform spectrum deconvolution (C)

setgcal Set the gradient calibration constant (M)

Applicability: Systems with pulsed field gradients (PFG) or imaging capabilities.

Syntax: `setgcal`

Description: Determines the gradient calibration constant `gcal` by using a proton phantom of known dimensions. `setgcal` requests the linear dimension of the phantom in the readout direction. It uses the value entered, together with cursor separation of this dimension from the image profile and the strength of the readout gradient `gro`, or `gzlvl1` if pulsed field gradients, to calculate `gcal` in units of gauss/cm-DAC units. You are then prompted whether this value should be entered. If you answer yes, it is stored as a system constant in the your global file.

Note that a particular value of `gcal` is closely related to the current eddy current compensation settings. If these settings are changed (e.g., reading in a new `curecc` file), a different value of `gcal` should be expected.

Before running `setgcal`, use the pulse sequence set up by `profile` to acquire a signal from a known sized object while the gradient is on.

See also: *Pulsed Field Gradient Modules Installation; VNMR User Guide: Imaging*

Related: `gcal` Gradient calibration constant (P)
`gro` Readout gradient strength in DAC units (P)
`profile` Set up pulse sequence for gradient calibration (M)

setgcoil Assign sysgcoil configuration parameter (M)

Syntax: `setgcoil<(file)>`

Description: Allows VNMR users to change the configured `gcoil` for the system. `setgcoil` updates the systemglobal parameter `sysgcoil` to the named table and updates the assignment values for the hardware-specific gradient calibration parameters `gcoil`, `gxcal`, `gycal`, `gzcal`, `griserate`, and `boresize` to their corresponding values, described in the named table. The directory `$vnmrsystem/imaging/gradtables` must have write permission for all VNMR users for the macro to be effective. This table now exists in the system local `/var/VNMR/gradtables` directory, with a soft link from `$vnmrsystem/imaging/gradtables` to that directory.

Arguments: `file` is the any legal file name defined for the parameter `gcoil`.

See also: *User Guide: Imaging*

Related: `boresize` Magnet bore size (P)
`config` Display current configuration and possible change it (M)
`gcoil` Read data from gradient calibration tables (P)
`griserate` Gradient rise rate (P)
`gxcal,gycal,gzcal` Gradient strength for X, Y, Z gradients (P)
`sysgcoil` System value for `gcoil` parameter (P)

setglideexp Set up GLIDE experiment from command line (M)

Syntax: `setglideexp(experiment)`

Description: Sets up a *GLIDE* experiment from the command line or from Tcl-dg. The `acquire def` file is read from the `glide/exp` experiment directory and a dialog is opened.

Arguments: `experiment` is the name of the *GLIDE* experiment.

Examples: `setglideexp('AuH')`

setGgrp Add user to specific GLIDE group (U)

Syntax: (From UNIX) `setGgrp group user`

Description: Adds a user to a specific *GLIDE* group. If a group does not exist, `setGgrp` adds the new group name to the group file and puts the user in the new group. If a user does not belong to any group, `setGgrp` makes that user public. If a user belongs to another group, `setGgrp` moves the user to a specified group. `setGgrp` can be executed only by `vnmr1`.

Arguments: `group` is the name of a *GLIDE* group.

`user` is a name of the individual to be added to the *GLIDE* group.

Examples: `setGgrp glidel mark`
`setGgrp public Sam`

See also: *Getting Started*

setgpe **Set phase encode gradient levels (M)**

Applicability: Systems with imaging capabilities.

Syntax: `setgpe`

Description: Provides for selection of the phase encode gradient step size levels (`gpe`, `gpe2`, `gpe3`) and gradient pulse timing (`tpe`, `tpe2`, `tpe3`) from the FOV parameters (`lpe`, `lpe2`, `lpe3`).

The program requires no inputs and automatically calculates the values of `gpe` and `tpe` (2D, 3D, 4D), `gpe2` and `tpe2` (3D and 4D), and `gpe3` and `tpe3` (4D) from the corresponding FOV parameters and requested acquisition matrix sizes (`nv1`, `nv2`, `nv3`). Defaults are supplied for 2D, 3D, and 4D matrix sizes if these have not been set by the user.

The result of the `setgpe` calculations results in setting the phase encode gradient levels so as to give the shortest possible phase encode timing. This prepares the sequence to collect data at the minimum te. Sequence applications, however, are free to rescale the values of the gradient level and timing parameters to meet their own requirements. Rescaling requires that:

$$gpe * tpe = gpe' * tpe'$$

The product of the gradient set size and phase encode pulse remain constant.

See also: *User Guide: Imaging*

Related:	<code>gpe</code>	Phase encoding gradient increment (P)
	<code>lpe</code>	Field of view for phase encode axis (P)
	<code>tpe</code>	Duration of phase encoding gradient pulse (P)

setgrid **Divide graphics window into rows and columns (C)**

Syntax: `setgrid(row<, column>)`

Description: Divides VNMR graphics window into an array of rows and columns (or window panes). Only one pane is active at a time. An individual pane can be activated by double-clicking in it with the left mouse button or by entering `setwin` in the input window.

Arguments: `row` is the number of rows (maximum is 3) in the graphics window. If 0 is entered, the number of rows remains the same; e.g., in `setgrid(0, 2)`, the number of rows is unchanged and two columns are created in each row.

`column` is the number of columns (maximum is 3) in the graphics window.

Alternate: Buttons 1 Row, 2 Rows, 3 Rows, 1 Column, 2 Columns, and 3 Columns in the Windows menu.

Examples: `setgrid(3)`
`setgrid(3, 3)`
`setgrid(0, 2)`

See also: *Getting Started*

Related:	<code>curwin</code>	Current window (P)
	<code>fontselect</code>	Open FontSelect window (C)
	<code>jwin</code>	Activate current window (M)
	<code>mapwin</code>	List of experiment numbers (P)
	<code>setwin</code>	Activate selected window (C)

setgro **Set readout gradient (M)**

Applicability: Systems with imaging capabilities.

Syntax: `setgro(<'min' | level>)`

Description: Sets the readout gradient by adjusting the values of `gro`, `sw`, and `at`. If entered without arguments, `setgro` operates in the automatic mode and uses a novel algorithm to estimate the maximum usable readout gradient. The algorithm is designed to provide a compromise between chemical shift artifact and S/N ratio in the image.

Arguments: `'min'` is a keyword to operate `setgro` in the automatic mode, to use simple algorithms to estimate the maximum usable readout gradient, and to set `gro`, `sw`, and `at` based on the estimate. Typical usage would be when operating at the shortest practical echo time.

`levels` is a real number that is interpreted as a gradient level in gauss/cm. Provided that the number is in the range 0 to `gmax`, `setgro` then calculates `sw` and sets `gro` and `at`.

Examples: `setgro`
`setgro('min')`
`setgro(1.0)`

See also: *User Guide: Imaging*

Related:

<code>at</code>	Acquisition time (P)
<code>gmax</code>	Maximum gradient strength (P)
<code>gro</code>	Readout gradient strength (P)
<code>sw</code>	Spectral width (P)

setgroup **Set group of a parameter in a tree (C)**

Syntax: `setgroup(parameter, group<, tree>)`

Description: Sets the group of a parameter in a tree.

Arguments: `parameter` is the name of the parameter.

`group` is one of the following keywords: `'all'`, `'sample'`, `'acquisition'`, `'processing'`, `'display'`, or `'spin'`.

`tree` is one of the keywords `'current'`, `'global'`, or `'processed'`. The default is `'current'`. See the `create` command for information on the types of trees.

Examples: `setgroup('a', 'sample')`
`setgroup('b', 'all', 'global')`

See also: *VNMR User Programming*

Related:

<code>create</code>	Create new parameter in a parameter tree (C)
<code>destroy</code>	Destroy a parameter (C)
<code>destroygroup</code>	Destroy parameters of a group in a tree (C)
<code>display</code>	Display parameters and their attributes (C)
<code>groupcopy</code>	Copy parameters of group from one tree to another (C)
<code>paramvi</code>	Edit a parameter and its attributes using <code>vi</code> text editor (M)
<code>setlimit</code>	Set limits of a parameter in a tree (C)
<code>setprotect</code>	Set protection mode of a parameter (C)

setgss **Select slice or voxel selection gradient levels (M)**

Applicability: Systems with imaging capabilities.

Syntax: `setgss(<gradient_name><, thickness_name>)>`

Description: Sets slice or voxel selection gradient levels, given the gradient level parameter and the thickness parameter. `setgss` searches the configuration list `sslist` (conjugate gradients) for the desired gradient level name.

If the gradient name is found (possibly multiple times), `setgss` calculates the bandwidth, in Hz, “cut” by each corresponding rf template on the list (`patlist`), at the length pointed to by the list (`plist`), and for the flip angle on the list (`fliplist`). The minimum bandwidth is assumed to define the “thickness” of the “cut.” The gradient level is then calculated from the minimum bandwidth selected by the rf pulses.

If `setgss` fails to find the supplied `gradient_name`, it returns the message “All RF templates used with `gradient_name` are nonselective.”

Arguments: `gradient_name` is the name of the gradient level parameter whose value is to be set. The default is the user is prompted for the parameter name.

`thickness_name` is the name of the thickness parameter from which to compute the gradient level. The default is the user is prompted for the parameter name.

Examples: `setgss`
`setgss('gss', 'thk')`

See also: *User Guide: Imaging*

Related: `fliplist` Standard flip angle list (P)
`patlist` Active pulse template parameter list (P)
`sslist` Conjugate gradient list (P)

sethw Set values for hardware in acquisition system (C)

Applicability: Syntax 1 through 5 apply to all systems (except that syntax 3, 4, and 5 are not available on *MERCURY-VX* or *GEMINI 2000* systems that lack automated spinner control hardware). Syntax 6 applies only to systems with a sample changer. Syntax 7 and 8 apply only to systems with a variable temperature (VT) controller. Syntax 9 applies only to *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*. Syntax 10 applies only to *UNITYINOVA*, *MERCURY-VX*, and *MERCURY* systems.

Syntax: (1) `sethw(<'wait' | 'nowait', >par1, val1<, par2, val2, ...)`
 (2) `sethw('lock', 'on' | 'off')`
 (3) `sethw('spin', speed)`
 (4) `sethw('spinner', 'bump')`
 (5) `sethw('eject', 'on' | 'off')`
 (6) `sethw('loc', location)`
 (7) `sethw('vt', 'reset' | 'off')`
 (8) `sethw('temp', temperature)`
 (9) `sethw('tune', mode)`
 (10) `sethw('lockfreq'<, lockfreq_value>)`

Description: Sets acquisition system hardware values. `sethw` cannot be used when an acquisition is in progress or when the `acqi` program is active.

Syntax 1 can be used to set the lock system parameters `lockpower`, `lockgain`, `lockphase`, and `z0`. This syntax can also be used to set the values of the shims. The particular shim that can be set depends upon the type of shim hardware present in the system. See the description of `shimset` for a list of the shim names for each type of shim hardware.

Syntax 2 turns the hardware lock on or off.

Syntax 3 controls spinning speed.

Syntax 4 carries the sample to bump by giving it a short burst of eject air. This is sometimes useful to reseal the sample if it is failing to spin.

Syntax 5 ejects and inserts samples into the probe. Entering the command `sethw('eject','on')` is equivalent in function to macros `eject` and `e`; and `sethw('eject','off')` is equivalent to macros `insert` and `i`.

Syntax 6 sets a location for the sample currently in the magnet on a system with a sample changer. The parameter `loc` is updated.

Syntax 7 resets the VT controller, useful when changing the probe in a system with VT regulation. By entering `sethw('vt','reset')` after installing a new probe in the magnet and attaching the VT controller interface to the probe, the VT controller is ready to regulate the temperature. No other parameters can be modified by the command. As an alternate, you can manually turn the VT controller unit off and then back on. Syntax 7 also turns the VT controller off by entering `sethw('vt','off')`.

Syntax 8 sets the temperature in degrees celsius. The host computer does not wait for the temperature to regulate.

Syntax 9 places the *MERCURY-VX*, *MERCURY*, or *GEMINI 2000* console into the tune mode. This syntax is used in the `btune`, `ctune`, `dtune`, `htune`, and `tuneoff` macros and normally is not entered by the user directly.

Syntax 10 sets the lock frequency, in MHz, on the ^{UNITY}*INOVA*, *MERCURY-VX*, or *MERCURY*.

Arguments: 'wait' or 'nowait' keyword must be either the first or last argument.

- 'wait' sends the new values to the acquisition console, verifies these values, and updates the corresponding parameters. This is the default.
- 'nowait' sends the new values to the console without verifying them or changing VNMR parameters.

`parameter1,value1,parameter2,value2,...` are pairs of parameter names and their values (see the first two examples below). At least one parameter name and its value must be specified. A maximum of ten parameters can be set.

'lock','on' is a keyword pair to turn the hardware lock on.

'lock','off' is a keyword pair to turn the hardware lock off.

'spin' is a keyword that identifies the next argument, `speed`, as the sample spinning speed, in Hz.

'spinner','bump' is a keyword pair to bump the sample.

'eject','on' is a keyword pair to eject the sample from the probe.

'eject','off' is a keyword pair to insert the sample into the probe.

'loc' is a keyword to identify that the next argument, `location`, is a number for the sample currently in the magnet ('loc' is unrelated to the `loc` parameter).

'vt','reset' is a keyword pair to reset the VT controller after the controller has been disconnected from the probe. This is equivalent to turning the VT controller power off and on.

'vt','off' is a keyword pair to turn the VT controller off.

'temp' is a keyword that identifies the next argument, `temperature`, as the requested sample temperature, in degrees celsius.

'tune' is a keyword that identifies the next argument, `mode`, as the tune mode to perform probe tuning on the *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*. On *MERCURY-VX* and *MERCURY*, mode is 1 for high band, 2 for low band, and 3 for off. On *GEMINI 2000*, mode is 1 for `htune`, 2 for `ctune`, 3 for `dtune`, 4 for `btune` (low band), 5 for `btune` (high band), 0 and 6 for off.

'lockfreq' is a keyword that the next argument is the lock frequency.

lockfreq_value is the **lockfreq** value, in MHz, for the lock frequency.

Examples: `sethw('z1c',30,'z2c',-50)`
`sethw('wait','z1',150,'z2',-400)`
`sethw('lock','on')`
`sethw('spin',20)`
`sethw('spinner','bump')`
`sethw('eject','on')`
`sethw('loc',5)`
`sethw('vt','reset')`
`sethw('lockfreq',46.042)`

See also: *Getting Started; User Guide: Liquids NMR*

Related:	btune	Tune broadband channel on <i>MERCURY-Series, GEMINI 2000</i> (M)
	ctune	Tune carbon channel on $^1\text{H}/^{13}\text{C}$ <i>GEMINI 2000</i> (M)
	dtune	Tune lock channel on <i>GEMINI 2000</i> (M)
	htune	Tune proton channel on <i>GEMINI 2000</i> (M)
	loc	Location of sample in tray (P)
	lockpower	Lock power (P)
	lockfreq	Lock frequency (P)
	lockgain	Lock gain (P)
	lockphase	Lock phase (P)
	readhw	Read current values of acquisition hardware (C)
	spin	Sample spin rate (P)
	tuneoff	Turn off probe tuning mode on <i>MERCURY-Series, GEMINI 2000</i> (M)
	z0	Z0 field position (P)

setint **Set value of an integral (M)**

Syntax: `setint(int_number<,value>)`

Description: Sets the value of an integral.

Arguments: `int_number` is the integral number. It corresponds to the index number displayed by **dli** if all integrals are shown (i.e., `intmod='full'`) or the region if alternating integrals are shown (i.e., `intmod='partial'`).
`value` sets the actual value of the selected integral. The default is **ins**.

Examples: `setint(2)`
`setint(1,3)`

See also: *Getting Started*

Related:	dli	Display list of integrals (C)
	ins	Integral normalization scale (P)
	intmod	Integral display mode (P)

setlimit **Set limits of a parameter in a tree (C)**

Syntax: (1) `setlimit(parameter,max,min,step_size<,tree>)`
(2) `setlimit(parameter,index<,tree>)`

Description: If syntax 1 is used, when a parameter value is changed, the new value is checked against the limits set by `max` and `min`. The new value must also be a multiple of `step_size + min` (e.g., `setlimit('r1',80,10,20)` allows the values 10, 30, 50, and 70). The value of the parameter can be further modified by a macro called `_parameter` if the proper protection bit is set (see the **setprotect** command).

If syntax 2 is used, the `max`, `min`, and `step_size` for a parameter are obtained from the `index`-th entry of a table set for the parameter by `parmax`, `parmin`, and `parstep` in `compar`.

Arguments: `parameter` is the name of the parameter.

`max` and `min` are the maximum and minimum limits on a parameter value.
`step_size` is the size of the steps allowed for a parameter within the limits `max` and `min`.

`tree` is one of the keywords 'global', 'current', 'processed', or 'systemglobal'. The default is 'current'. Refer to the `create` command for a more information on the types of parameter trees.

`index` is an index into a lookup table. When a single `index` argument is given, the parameter's protection bits (see the `setprotect` command) are set so that the table lookup is turned on.

Examples: `setlimit('a',80,10,20)`
`setlimit('b',1e5,-3e2,1,'global')`
`setlimit('dpwr',9)`

See also: *VNMR User Programming*

Related:	<code>create</code>	Create new parameter in a parameter tree (C)
	<code>destroy</code>	Destroy a parameter (C)
	<code>display</code>	Display parameters and their attributes (C)
	<code>fread</code>	Read parameters from file and load them into a tree (C)
	<code>fsave</code>	Save parameters from a tree to a file (C)
	<code>paramvi</code>	Edit a parameter and its attributes using vi text editor (M)
	<code>parmax</code>	Parameter maximum values (P)
	<code>parmin</code>	Parameter minimum values (P)
	<code>parstep</code>	Parameter step size values (P)
	<code>prune</code>	Prune extra parameters from current tree (C)
	<code>setgroup</code>	Set group of a parameter in a tree (C)
	<code>setprotect</code>	Set protection mode of a parameter (C)
	<code>settype</code>	Change type of a parameter (C)
	<code>setvalue</code>	Set value of any parameter in a tree (C)

setlk Set up lock parameters (M)

Syntax: `setlk(solvent)`

Description: Called from other macros to provide adjustment of locking and shimming as a function of solvent. Removing quotation marks from around different parts of the text file of the macro places that particular section into effect. If the macro is left unchanged, setting `alock='s'` is required in the parameter sets where used.

Arguments: `solvent` is the solvent to be used.

See also: *User Guide: Liquids NMR*

Related: `alock` Automatic lock status (P)

setlockfreq Set lock frequency on systems other than UNITY and VXR-S (M)

Applicability: `UNITY INOVA`, `MERCURY-VX`, `MERCURY`, `UNITYplus`, and `GEMINI 2000` systems.

Syntax: `setlockfreq`

Description: Calculates and sets the lock frequency parameter `lockfreq`. Before using `setlockfreq`, you must acquire a signal using ^1H as the transmitter nucleus

(`tn= 'H1 '`). To avoid errors in calculating frequencies, set `lockfreq= 'n '` before starting the acquisition.

See also: *VNMR and Solaris Software Installation*

Related: `lockfreq` Lock frequency (P)
`tn` Nucleus for observe transmitter (P)

setloop Control arrayed and real-time looping (M)

Applicability: Systems with imaging capabilities.

Syntax: `setloop`

Description: Set the values for `nf` and `ni` to control arrayed and real-time looping.

Loop control in imaging experiments, such as multislice, multiecho, and phase encoding, is set through a series of parameters (`ne`, `ns`, `nv`, `nv2`, `nv3`) directly set by the user. Underlying these parameters are two lower level parameters, `nf` and `ni`, used during pulse sequence execution to determine the mode of data acquisition. `setloop` manages the values of `nf` and `ni` as required to be consistent with the experiment parameters `ne`, `nv`, etc.

Two modes of data acquisition are supported in VNMR: arrayed and compressed. The difference between the modes is mainly in the data flow timing between host and acquisition computers:

- Arrayed data acquisition involves continuous communications between host and acquisition computers as pulse sequence instructions are sent to the acquisition CPU and data is returned to the host Sun for each element in the arrayed experiment. All explicitly arrayed experiments (e.g., `pw=10, 20, 30`) run in this manner. 2D experiments, including most high-resolution liquids and many imaging experiments, also run as “implicit” arrays, with the array size set by the parameter `ni`. Although communications between acquisition and host computers are quite fast, a small delay (typically a few milliseconds) is required to accommodate the communications and reinitialization between array elements. Certain fast imaging experiments, such as turboflash, Echo Planar Imaging (EPI), or even conventional multislice, often require loop timing similar to this interelement delay. These experiments use a second mode of data acquisition: the compressed mode.
- In compressed data acquisition, a single pulse sequence instruction set is sent to the acquisition computer, which then manages the entire experiment through real-time loops and pulse sequence elements. All data accumulated in the real-time loops is retained in the acquisition data memory until the experiment or array element is complete, at which time the data is sent back to the host. No timing overhead is associated with a real-time loop, and extremely short timing intervals may therefore be achieved with the compressed mode. Compressed data acquisition is controlled by the parameter `nf`, which requires that the number of points acquired must be $nf * np$. Experiments may be run completely in arrayed acquisition mode, or completely in compressed acquisition mode, or in a combination of the two.

`setloop` uses the `seqcon` parameter to determine which acquisition loops, if present, are arrayed and which are compressed. It then computes `nf` as the product of all compressed loop counts, and sets `ni` appropriately as either `nv` in the case of uncompressed phase-encode, or zero in the case of compressed phase-encode.

Each of the parameters `ne`, `ns`, `nv`, `nv2`, and `nv3` have corresponding underscore macros that execute `setloop`. Therefore, `setloop` is a lower level “management” macro that is run automatically each time one of these parameters is entered, and will not normally be run explicitly by the user. The comprehensive setup macro `imprep` also performs the `setloop` function. If `imprep` has been executed, there is no need to run `setloop`.

See also: *User Guide: Imaging*

Related:	<code>d0</code>	Overhead delay between FIDs (P)
	<code>flashc</code>	Convert compressed 2D data to standard 2D format (C)
	<code>ne</code>	Number of echoes to be acquired (P)
	<code>nf</code>	Number of FIDs (P)
	<code>ns</code>	Number of slices to be acquired (P)
	<code>nv</code>	umber of 2D phase encode steps to be acquired (P)
	<code>seqcon</code>	Acquisition loop control (P)

setLP1 Set F1 linear prediction parameters

Syntax: `setLP1<(current_length<, extended_length)>`

Description: Sets F1 linear prediction parameters. If no arguments are specified, the interferograms are quadrupled in length.

Arguments: `current_length` is the number of complex points now existing (`ni`).
`extended_length` is the number of points desired after the (forward) linear prediction.

Related: `ni` Number of increments in 1st indirectly detected dimension (P)

setnoether Disconnect host computer from Ethernet (U)

Syntax: `setnoether`

Description: Disconnects the host computer from the Ethernet network. Only `root` can execute this shellscript properly. `setnoether` does nothing if the system is already disconnected from the Ethernet network.

On systems running Solaris, `setnoether` renames the `hostname.le0`, `defaultdomain`, and `defaultrouter` files so that Ethernet is not activated when the system is rebooted. `setnoether` does not affect the second Ethernet interface on *GEMINI 2000* systems.

See also: *VNMR and Solaris Software Installation*

Related: `setether` Connect or reconnect host computer to Ethernet (U)

setoffset Calculate offset frequency for given nucleus and ppm (M)

Syntax: `setoffset(nucleus, ppm):offsetfreq`

Description: Using the `setref` macro, `setoffset` calculates the offset frequency for a given chemical shift and returns the value.

Arguments: `nucleus` is the given nucleus.

`ppm` is the chemical shift.

`offsetfreq` returns the offset frequency for the given chemical shift.

Examples: `setoffset(tn, 5):tof`
`setoffset('C13', 85):dof`

See also: *Getting Started*

Related: `setref` Set frequency referencing for proton spectra (M)

setparams Write parameter to current probe file (M)

Syntax: `setparams (param, value<, nucleus>)`

Description: Writes the value of a parameter to the current probe file. The name of the probe file is referenced from the parameter `probe`.

Arguments: `param` is the name of the parameter to write.
`value` is a string with the value to be written for the parameter.
`nucleus` is the nucleus to write in the probe file. The default is the current value of the parameter `tn`.

Examples: `setparams ('pw90' , '10')`
`setparams ('pplvl' , '60')`
`setparams ('dpwr' , $strdpwr , 'H1')`

See also: *Getting Started*

Related: `addnucleus` Add new nucleus to existing probe file (M)
`addparams` Add parameter to current probe file (M)
`addprobe` Create new probe directory and probe file (M)
`getparam` Retrieve parameter from probe file (M)
`probe` Probe type (P)
`tn` Nucleus for the observe transmitter (P)
`updateprobe` Update probe file (M)

setpen Set maximum number of HP plotter pens (M)

Syntax: `setpen<(maxpen, max_number_pens)>`

Description: Allows the user to interactively define the maximum number of pens when changing to a Hewlett-Packard plotter.

Arguments: `maxpen` is the current value of the parameter `maxpen`.
`maximum_number_pens` is the maximum number of pens to be used. If the value of `max_number_pens` is less than or equal to the current value of the parameter `maxpen`, this value becomes the new value of `maxpen`.

See also: *Getting Started*

Related: `color` Select plotting colors from a graphical interface (M)
`maxpen` Maximum number of pens to use (P)

setplotdev Return characteristics of a named plotter (C)

Syntax: `setplotdev<:plotter_type>`

Description: Returns information from the `devicenames` and `devicetable` files to identify the characteristics of a plotter. This command need never be entered directly by a user because it is automatically called whenever the `plotter` parameter is set. Note that different “types” of plotters (and printers) are characterized in `devicetable`. The `devicenames` file associates different “names” to a given “type.”

Arguments: `plotter_type` returns the characteristics of the named plotter.

See also: *VNMR and Solaris Software Installation*

Related: `plotter` Plotter device (P)

setpower Set power and pulsewidth for a given γ B1 value (M)

Syntax: `setpower (γ B1, nucleus)`

Description: Sets power level and `pw90` values. For `tn`, `setpower` uses `ref_pwr` and `ref_pw90` from the parameter set or from the probe table. For `dn`, it uses `ref_pwx1v1` and `ref_pwx90` from the parameter set or from the probe table. For `dn2`, it uses `ref_pwx21v1` and `ref_pwx290` from the parameter set or from the probe table. If the reference power levels and pulse width do not exist, `setpower` uses `tpwr` (`pw90`), `dpwr` (`1/dmf`) or `dpwr2` (`1/dmf2`) (if the nucleus is `tn`, `setpower` uses `tpwr`; if the nucleus is `dn`, it uses `dpwr`; if the nucleus is `dn2`, it uses `dpwr2`).

Arguments: `γB1` is a given `γB1` value.
`nucleus` is a given nucleus.

Examples: `setpower (sw, tn)`
`setpower (5000, H1)`

Related:

<code>dn</code>	Nucleus for first decoupler (P)
<code>dn2</code>	Nucleus for second decoupler (P)
<code>dpwr</code>	Power level for first decoupler with linear amplifiers (P)
<code>dpwr2</code>	Power level for second decoupler (P)
<code>pw90</code>	90° pulse width (P)
<code>sw</code>	Spectral width in directly detected dimension (P)
<code>tpwr</code>	Observe transmitter power level with linear amplifiers (P)

setprotect Set protection mode of a parameter (C)

Syntax: `setprotect(parameter, 'set' | 'on' | 'off', bit_vals<, tree>)`

Description: Enables changing the protection bits associated with a parameter.

Arguments: `parameter` is the name of the parameter.

'set' causes the current protection bits for the parameter to be completely replaced with the bits specified by `bit_vals`.

'on' causes the bits specified in `bit_vals` to be turned on without affecting any other protection bits.

'off' causes the bits specified in `bit_vals` to be turned off without affecting any other protection bits.

`bit_vals` is the *sum* of the *values* of bits selected from the following list:

<i>Bit</i>	<i>Value</i>	<i>Description</i>
0	1	Cannot array the parameter
1	2	Cannot change active/not active status
2	4	Cannot change the parameter value
3	8	Causes <code>_parameter</code> macro to be executed (e.g., if parameter is named <code>sw</code> , macro <code>_sw</code> is executed when <code>sw</code> is changed)
4	16	Avoids automatic redisplay
5	32	Cannot delete parameter
6	64	System ID for spectrometer or data station
7	128	Cannot copy parameter from tree to tree
8	256	Will not set <code>array</code> parameter
9	512	Cannot set parameter enumerals values
10	1024	Cannot change the parameter's group
11	2048	Cannot change protection bits
12	4096	Cannot change the display group
13	8192	Look up minimum, maximum, step values in table

For example, to change the first two protection bits, with values 1 and 2, either enter `setprotect` twice (once for each value) with the keyword 'on', or enter `setprotect` once with `bit_vals` set to 3 (sum of 1 and 2) with the keyword 'set'.

`tree` is one of the keywords 'global', 'current', 'processed', or 'systemglobal'. The default is 'current'. Refer to the `create` command for more information on the types of parameter trees.

Examples: `setprotect('syn','on',2)`
`setprotect('pslabel','on',8)`

See also: *VNMR User Programming*

Related:	<code>array</code>	Parameter order and precedence (P)
	<code>create</code>	Create new parameter in a parameter tree (C)
	<code>destroy</code>	Destroy a parameter (C)
	<code>display</code>	Display parameters and their attributes (C)
	<code>fread</code>	Read parameters from file and load them into a tree (C)
	<code>fsave</code>	Save parameters from a tree to a file (C)
	<code>paramvi</code>	Edit a parameter and its attributes using vi text editor (M)
	<code>prune</code>	Prune extra parameters from current tree (C)
	<code>setlimit</code>	Set limits of a parameter in a tree (C)

setref **Set frequency referencing (M)**

Syntax: `setref<(nucleus)>:$rfl,$rfp,$reffrq,$refpos`

Description: Calculates the referencing for a given parameter or FID data set, for samples locked on deuterium, and based on the chemical shift of the lock solvent line. `setref` uses information in `/vnmr/solvents` (^2H chemical shift for current solvent) and `/vnmr/nuctables/nuctabref` (absolute reference frequencies for NMR nuclei) to predict the position of the reference frequency with the current solvent, spectral window, and spectrometer frequency. `setref` assumes a locked sample.

Arguments: An argument and return values are beneficial for the use of `setref` within other macros such as `setref1` and `setref2`. By default (i.e., without an argument), `setref` calculates the referencing for 1D spectra or for the directly detected dimension in nD spectra (f2 in 2D, f3 in 3D).

When only `nucleus` is used as an argument, `setref` returns values without setting parameters.

`$rfl`, `$rfp`, `$reffrq`, `$refpos` are return values for reference peak position, reference peak frequency, reference line frequency, and reference line position, respectively.

Examples: `setref`
`setref('C13'):$rfl,$rfp`

See also: *Getting Started*

Related:	<code>reffrq</code>	Reference frequency of reference line (P)
	<code>refpos</code>	Position of reference frequency (P)
	<code>rfl</code>	Reference peak position (P)
	<code>rfp</code>	Reference peak frequency (P)
	<code>rl</code>	Set reference line in directly detected dimension (M)
	<code>setref1</code>	Set frequency referencing for 1st indirectly detected dimension (M)
	<code>setref2</code>	Set frequency referencing for 2nd indirectly detected dimension (M)
	<code>setup</code>	Set up parameters for basic experiments (M)
	<code>tmsref</code>	Reference 1D proton or carbon spectrum to TMS (M)

that the scout run is analyzed, parameters adjusted, and an appropriate solvent-suppressed experiment run.

If parameters already exist in the current experiment for performing the `lc1d` pulse sequence, `setscout` turns off the solvent suppression portion of the sequence; if they do not exist, they are created and set to default values using `lc1d`.

See also: *User Guide: Liquids NMR*

Related: `lc1d` Pulse sequence for LC-NMR (M)
`setwet` Set up a solvent-suppressed experiment (M)

setssfilter Set `sslsfrq` to the frequencies of each of the suppressed solvents (M)

Applicability: Systems with LC-NMR accessory.

Syntax: `setssfilter`

Description: Sets `sslsfrq` to the frequencies of each of the suppressed solvents.

See also: *User Guide: Liquids NMR*

setsw Set spectral width (M)

Syntax: `setsw(downfieldppm,upfieldppm)`

Description: Sets `sw` and `tof` for the given spectral window and also does referencing.

Arguments: `downfieldppm` is the downfield frequency, in ppm.

`upfieldppm` is the upfield frequency, in ppm.

Examples: `setsw(12,0)`
`setsw(235,-15)`

See also: *Getting Started*

Related: `setsw1` Set spectral width in evolution dimension (M)
`setsw2` Set spectral width in 2nd evolution dimension (M)
`sw` Spectral width in directly detected dimension (P)
`tof` Frequency offset for observe transmitter (P)

setsw1 Set spectral width in evolution dimension (M)

Syntax: `setsw1(nucleus,downfieldppm,upfieldppm):offset`

Description: Sets `sw1` for the given spectral window and also does referencing.

Arguments: `nucleus` returns the nucleus.

`downfieldppm` is the downfield frequency, in ppm.

`upfieldppm` is the upfield frequency, in ppm.

`offset` returns the appropriate offset.

Examples: `setsw1(tn,12,0)`
`setsw1(dn,235,-15):dof`

See also: *User Guide: Liquids NMR*

Related: `setsw` Set spectral width (M)
`sw1` Spectral width in 1st indirectly detected dimension (P)

setsw2 Set spectral width in 2nd evolution dimension (M)

Syntax: `setsw2(nucleus,downfieldppm,upfieldppm):offset`

Description: Sets `sw2` for the given spectral window and also does referencing.

Arguments: `nucleus` returns the nucleus.

`downfieldppm` is the downfield frequency, in ppm.

`upfieldppm` is the upfield frequency, in ppm.

`offset` returns the appropriate offset.

Examples: `setsw2(tn, 12, 0)`
`setsw2(dn, 235, -15):dof`

See also: *User Guide: Liquids NMR*

Related: `setsw` Set spectral width (M)
`sw2` Spectral width in 2nd indirectly detected dimension (P)

setselfrqc Set selective frequency and width (M)

Syntax: `setselfrqc`

Description: Sets selective frequency and width of the excitation bandwidth for selective excitation. Used after `TOCSY1D` and `NOESY1D` selection. Selected frequencies and widths of the excitation bandwidth are used by `suselfrq`.

Related: `NOESY1D` Change parameters for NOESY1D experiment (M)
`suselfrq` Select peak, continue selective excitation experiment (M)
`TOCSY1D` Change parameters for TOCSY1D experiment (M)

setselinv Set up selective inversion (M)

Syntax: `setselinv`

Description: Sets power, pulsewidth, and shape for selective inversion; used by `suselfrq`. By default, `setselinv` selects a q3 gaussian cascade pulse if a waveform generator or linear modulator is present (`UNITYINOVA` and `UNITYplus`). Otherwise, `setselinv` selects a “rectangular” pulse.

Related: `setselfrqc` Select selective frequency and width (M)
`suselfrq` Select peak, continue selective excitation experiment (M)

settclddefault Select default display templates for pulse sequence (M)

Syntax: `settclddefault(<default><, sequence>)>`

Description: Selects the display templates to use as the default for a pulse sequence.

Arguments: `default` is the name of the set of display templates to use for the default display of the current pulse sequence (defined by the parameter `seqfil`). If no arguments are given, the user is prompted for the name of the display templates.
`sequence` defines which pulse sequence will use the default displays of the pulse sequence given as the first argument. The default is the pulse sequence defined by the parameter `seqfil`.

Examples: `settclddefault`
`settclddefault('cosy')`
`settclddefault('default2d', 'HMQC8')`

See also: *VNMR User Programming*

Related: `seqfil` Pulse sequence name (P)

settype Change type of a parameter (C)

Syntax: `settype(parameter, type<, tree>)`

Description: Changes the type of an existing parameter. A string parameter can be changed into a string or flag type, or a real parameter can be changed into a real, delay, frequency, pulse, or integer type. Note that `settype` cannot change a string parameter into a real, or change a real into a string.

Arguments: `parameter` is the name of an existing parameter.
`type` is one of the keywords 'string', 'flag', 'real', 'delay', 'frequency', 'pulse', or 'integer'.
`tree` is one of the keywords 'global', 'current', 'processed', or 'systemglobal'. The default is 'current'. Refer to the [create](#) command for more information on the types of parameter trees.

Examples: `settype('in','flag','global')`
`settype('p12','pulse')`

See also: *VNMR User Programming*

Related:	create	Create new parameter in a parameter tree (C)
	display	Display parameters and their attributes (C)
	setgroup	Set group of a parameter in a tree (C)
	setlimit	Set limits of a parameter in a tree (C)
	setprotect	Set protection mode of a parameter (C)
	setvalue	Set value of any parameter in a tree (C)

setup **Set up parameters for basic experiments (M)**

Syntax: `setup<(nucleus< , solvent>)>`

Description: Returns a parameter set to do the experiment requested, complete with positioning of the transmitter and decoupler. Parameters set by `setup` are recalled from the `/vnmr/stdpar` directory or from the user's `stdpar` directory if the appropriate file exists there. Any changes made to the files in these directories are reflected in `setup`. The default parameters for carbon and proton survey spectra are in files `/vnmr/stdpar/C13.par` and `/vnmr/stdpar/H1.par`, respectively. These files should be modified as desired to produce spectra under desirable conditions.

Arguments: `nucleus` is a nucleus chosen from the files in `/vnmr/stdpar` or in the user's `stdpar` directory (e.g., 'H1', 'C13', 'P31'). The default is the system displays a menu with choices of nuclei. After you chose a nucleus, the system displays a menu with choices of solvents, and you choose one.
`solvent` is a solvent chosen from the file `/vnmr/solvents` (e.g., 'CDC13', 'C6D6', 'D2O'). The default is 'CDC13'.

Alternate: Nucleus,Solvent button in the Setup Menu.

Examples: `setup`
`setup('H1')`
`setup('C13','DMSO')`

See also: *Getting Started*

setup_dosy **Set up gradient levels for DOSY experiments (M)**

Syntax: `setup_dosy`

Description: Initiates a dialogue to set up an array of `gzlv11` values for DOSY experiments. `setup_dosy` requests the number of array increments and an initial and a final `gzlv11` value and sets up an array that gives increments in `gzlv11` squared between these limits. `setup_dosy` retrieves the gradient strength from the probe calibration file if `probe<>'` and stores it in the local

experimental parameter `DAC_to_G`. If `probe=` ' ' (i.e., the probe is not defined), then `DAC_to_G` is set to the current value of the global parameter `gcal`.

See also: *User Guide: Liquids NMR*

Related: `dosy` Process DOSY experiments (M)
`DAC_to_G` Parameter to store gradient calibration value in DOSY sequences (P)
`setgcal` Set the gradient calibration constant (M)

setvalue Set value of any parameter in a tree (C)

Syntax: `setvalue(parameter, value<, index><, tree>)`

Description: Sets the value of any parameter in a tree. This command bypasses the normal range checking for parameter entry, as well as bypassing any action that would be invoked by the parameter's protection mode (see the `setprotect` command). If the parameter entry normally causes a `_parameter` macro to be executed, this action also is bypassed.

Arguments: `parameter` is the name of the parameter.
`value` is the value to set to the parameter.
`index` is the number of a single element in an arrayed parameter. The default is 1.
`tree` is one of the keywords 'global', 'current', 'processed', or 'systemglobal'. The default is 'current'. Refer to the `create` command for more information on the types of parameter trees.

Examples: `setvalue('arraydim', 128, 'processed')`

See also: *VNMR User Programming*

Related: `create` Create new parameter in a parameter tree (C)
`setprotect` Set protection mode of a parameter (C)

setwave Write a wave definition string into Pbox.inp file (M)

Syntax: `setwave('sh bw/pw ofs st ph fla trev d1 d2 d0')`

Description: Sets up a single excitation band in the `Pbox.inp` file. An unlimited number of waves can be combined by reapplying `setwave`.

Arguments: A single string of 1 to 10 wave parameters in predefined order. Note that a single quote is required at the start and the end of the entire string, but no single quotes are required surrounding characters and strings inside the entire string.

`sh` is the name of a shape file.

`bw/pw` is either the bandwidth, in Hz, or the pulsewidth, in sec.

`ofs` is the offset, in Hz.

`st` is a number specifying the spin status: 0 for excitation, 1 for de-excitation, or 0.5 for refocusing.

`ph` is the phase (or phase cycle, see `wavelib/supercycles`).

`fla` is the flip angle. Note that `fla` can override the default flip angle.

`trev` is a time reversal. This can be used to cancel time reversal if spin status (`st`) is set to 1 for `Mxy`.

`d1` is the delay, in sec, prior the pulse.

`d2` is the delay, in sec, after the pulse.

d0 is a delay or command prior to d1. If d0=a, the wave is appended to the previous wave.

Examples: `setwave('eburp1')`
`setwave('GARP 12000.0')`
`setwave('esnob 600 -1248.2 1 90.0 n n 0.001')`

See also: *User Guide: Liquids NMR*

Related: [Pbox](#) Pulse shaping software (U)

| **setwin** **Activate selected window (C)**

Syntax: `setwin(row<,column>)`

Description: Activates a specific pane in the VNMR graphics window. Panes are numbered sequentially from left to right and top to bottom.

Arguments: `row` is the number of the row containing the pane to be activated.
`column` is the number of the column containing the pane to be activated.

Examples: `setwin(3)`
`setwin(1,2)`

See also: *Getting Started*

Related: [curwin](#) Current window (P)
[fontselect](#) Open FontSelect window (C)
[jwin](#) Activate current window (M)
[mapwin](#) List of experiment numbers (P)
[setgrid](#) Activate selected window (M)

sf **Start of FID (P)**

Description: Sets the start of the FID display. This parameter can be entered in the usual way or interactively controlled by the sf wf button during a FID display.

Values: 0 to the value of `at`, in seconds.

See also: *Getting Started*

Related: [at](#) Acquisition time (P)
[dcon](#) Display noninteractive color intensities map (C)
[dconi](#) Interactive 2D data display (C)
[df](#) Display a single FID (C)
[sf1](#) Start of interferogram in 1st indirectly detected dimension (P)
[sf2](#) Start of interferogram in 2nd indirectly detected dimension (P)
[vf](#) Vertical scale of FID (P)
[wf](#) Width of FID (P)

sf1 **Start of interferogram in 1st indirectly detected dimension (P)**

Description: Sets the start of the interferogram display in the first indirectly detected dimension.

Values: 0 to $(2 \times ni) / sw1$, in seconds.

See also: *User Guide: Liquids NMR*

Related: [ni](#) Number of increments in 1st indirectly detected dimension (P)
[sf](#) Start of FID (P)
[sw1](#) Spectral width in 1st indirectly detected dimension (P)
[wf1](#) Width of interferogram in 1st indirectly detected dimension (P)

sf2 Start of interferogram in 2nd indirectly detected dimension (P)

Description: Sets the start of the interferogram display in the second indirectly detected dimension.

Values: 0 to $(2 \times ni2)/sw2$, in seconds.

See also: *User Guide: Liquids NMR*

Related: **ni2** Number of increments in 2nd indirectly detected dimension (P)
sf Start of FID (P)
sw2 Spectral width in 2nd indirectly detected dimension (P)
wf2 Width of interferogram in 2nd indirectly detected dimension (P)

sfrq Transmitter frequency of observe nucleus (P)

Description: Contains the frequency for the observe transmitter. **sfrq** is automatically set when **tn** is changed, and it should not be necessary for the user to manually set this parameter.

Values: Number, in MHz.

See also: *Getting Started*

Related: **dfrq** Transmitter frequency of first decoupler (P)
dfrq2 Transmitter frequency of second decoupler (P)
dfrq3 Transmitter frequency of third decoupler (P)
tn Nucleus for observe transmitter (P)
tof Frequency offset for observe transmitter (P)
spcfrq Display frequencies of rf channels (M)

sh2pul Set up for a shaped observe excitation sequence (M)

Applicability: Systems with waveform generators.

Syntax: **sh2pul**

Description: Behaves like standard two-pulse sequence S2PUL but with the normal hard pulses changed into shaped pulses from the waveform generator. The name of the shaped pulse associated with **pw** is **pwpat** and **p1** is **p1pat**. Information about the specifics of power settings and bandwidths is available from the macros **bandinfo** and **pulseinfo**.

See also: *VNMR User Programming*

Related: **bandinfo** Shaped pulse information for calibration (M)
p1pat Shape of an excitation pulse (P)
pwpat Shape of refocusing pulse (P)
pulseinfo Shaped pulse information for calibration (M)

shdec Set up for shaped observe excitation sequence (M)

Applicability: Systems with waveform generators.

Syntax: **shdec**

Description: Sets up the SHDEC pulse sequence that generates a shaped pulse on the observe channel using the waveform generator. It also allows for programmed (e.g.: multiselective) homodecoupling or solvent presaturation using the observe transmitter, and an optional gradient pulse following the excitation pulse.

See also: *User Guide: Liquids NMR*

Related: **Pbox** Pulse shaping software (U)

shell **Start a UNIX shell (C)**

Syntax: `shell<(command)>:$file1,$file2,...`

Description: Brings up a normal UNIX shell for the user. On the Sun, a pop-up window is created. On the GraphOn terminal, the entire terminal is used.

Arguments: `command` is a UNIX command line to be executed by `shell`. The default is to bring up a UNIX shell. If the last character in the command line is the symbol `&`, the command is executed in background, which allows VNMR commands to be entered and executed while the `shell` command is still running. Note that if this background feature is used, any printed output should be redirected to a file. Otherwise, the output may pop up in the text window at random times.

`$file1, $file2, ...` are names of files to hold text lines generated as a result of the UNIX command. The default is to display the text lines. Each file receives a single display line.

Examples: `shell`
`shell('ps')`
`shell('ls -lt'):$filelist`
`shell('ls -t | grep May')`
`shell('cd /vnmr/psglib; cat '+seqfil+'.c')`
`shell(systemdir+'/acqbin/Acqstat '+hostname+' &')`

See also: *Getting Started, VNMR User Programming*

Related: [shell](#) Start an interactive UNIX shell (C)

shell **Start an interactive UNIX shell (C)**

Syntax: `shelli(command)`

Description: On a terminal, runs interactively the UNIX command line given as the argument. No return or output variables are allowed. On window-based VNMR, `shell` is identical to the `shell` command.

Arguments: `command` is a UNIX command line to be executed.

Examples: `shelli('vi myfile')`

See also: *Getting Started, VNMR User Programming*

Related: [shell](#) Start a UNIX shell (C)

shellreturn **Run UNIX shell program and return arguments (obsolete)**

Description: This macro is no longer in VNMR. It is replaced by the `shell` command.

Related: [shell](#) Start a UNIX shell (C)

shim **Submit an Autoshim experiment to acquisition (C)**

Syntax: `shim`

Description: Performs validity checks on the acquisition parameters and then submits an Autoshim experiment to acquisition.

See also: *Getting Started*

Related: [au](#) Submit experiment to acquisition and process data (C)
[change](#) Submit a change sample experiment to acquisition (M)
[ga](#) Submit experiment to acquisition and FT the result (C)
[go](#) Submit experiment to acquisition (C)
[lock](#) Submit an Autolock experiment to acquisition (C)
[sample](#) Submit change sample, autoshim experiment to acquisition (M)

`spin` Submit a spin setup experiment to acquisition (C)
`su` Submit a setup experiment to acquisition (M)

shimset **Type of shim set (P)**

Description: Configuration parameter for the type of shims on the system. The value of `shimset` is set using the Shimset label in the CONFIG window (opened from `config`).

Values: 1 to 14, where the value identifies one of the following shim sets:

1 is a shim set in a Varian 13-shim supply with computer-controlled axial shims $z1$, $z1c$, $z2$, $z2c$, $z3$, $z4$, and radial shims $x1$, $y1$, xz , yz , xy , $x2y2$, $x3$, $y3$. Shims can be adjusted from -2047 to $+2047$. This value is set implicitly for the *GEMINI 2000* and is also used with the Ultra•nmr shim system when operated from the HIM box (Varian 13 Shims choice in CONFIG window).

2 is a shim set in a Oxford 18-shim supply with computer-controlled axial shims $z1$, $z1c$, $z2$, $z2c$, $z3$, $z4$, $z5$, and radial shims $x1$, $y1$, xz , yz , xy , $x2y2$, $x3$, $y3$, $xz2$, $yz2$, zxy , $zx2y2$. Shims can be adjusted from -2047 to $+2047$ (Oxford 18 Shims choice in CONFIG window).

3 is a shim set in a Varian 23-shim supply with computer-controlled axial shims $z1$, $z2$, $z3$, $z4$, $z5$, $z6$, and radial shims $x1$, $y1$, xz , yz , xy , $x2y2$, $x3$, $y3$, $xz2$, $yz2$, zxy , $zx2y2$, $z3x$, $z3y$, $z2x2y2$, $z2xy$. Shims can be adjusted from -32767 to $+32767$ (Varian 23 Shims choice in CONFIG window).

4 is a shim set in a Varian 28-shim supply with computer-controlled axial shims $z1$, $z2$, $z3$, $z4$, $z5$, $z6$, $z7$, and radial shims $x1$, $y1$, xz , yz , xy , $x2y2$, $x3$, $y3$, $xz2$, $yz2$, zxy , $zx2y2$, $z3x$, $z3y$, $z2x2y2$, $z2xy$, $zx3$, $zy3$, $z4x$, $z4y$. Shims can be adjusted from -32767 to $+32767$ (Varian 28 Shims choice in CONFIG window).

5 is a shim set in an Ultra•nmr shim system (39 shim channels) with computer-controlled axial shims $z1$, $z1c$, $z2$, $z2c$, $z3$, $z3c$, $z4$, $z4c$, $z5$, $z6$, $z7$, $z8$, and radial shims $x1$, $y1$, xz , yz , xy , $x2y2$, $x3$, $y3$, $xz2$, $yz2$, zxy , $zx2y2$, $z3x$, $z3y$, $z2x2y2$, $z2xy$, $zx3$, $zy3$, $z4x$, $z4y$, $z3x2y2$, $z3xy$, $z2x3$, $z2y3$, $z3x3$, $z3y3$, $z4x2y2$, $z4xy$, $z5x$, $z5y$. Shims can be adjusted from -32767 to $+32767$ (Ultra Shims choice in CONFIG window).

6 is a shim set in a Varian 18-shim supply with computer-controlled axial shims $z1$, $z2$, $z3$, $z4$, $z5$, and radial shims $x1$, $y1$, xz , yz , xy , $x2y2$, $x3$, $y3$, $xz2$, $yz2$, zxy , $zx2y2$. Shims can be adjusted from -32767 to $+32767$ (Varian 18 Shims choice in CONFIG window).

7 is a shim set in a Varian 20-shim supply with computer-controlled axial shims $z1$, $z2$, $z3$, $z4$, $z5$, and radial shims $x1$, $y1$, xz , yz , xy , $x2y2$, $x3$, $y3$, $xz2$, $yz2$, zxy , $zx2y2$, $z3x$, $z3y$. Shims can be adjusted from -32767 to $+32767$ (Varian 20 Shims choice in CONFIG window).

8 is a shim set in a Oxford 15-shim supply with computer-controlled axial shims $z1$, $z2$, $z3$, $z4$, and radial shims $x1$, $y1$, xz , yz , xy , $x2y2$, $zx2y2$, $xz2$, $yz2$, zxy . Shims can be adjusted from -2047 to $+2047$ (Oxford 15 Shims choice in CONFIG window).

9 is a shim set in a Varian Ultra•nmr shim system II (40 shim channels) with computer-controlled axial shims $z1$, $z1c$, $z2$, $z2c$, $z3$, $z3c$, $z4$, $z4c$, $z5$, $z6$, $z7$, $z8$, and radial shims $x1$, $y1$, xz , yz , xy , $x2y2$, $x3$, $y3$, $x4$, $y4$, $xz2$, $yz2$, zxy , $zx2y2$, $z3x$, $z3y$, $z2x2y2$, $z2xy$, $zx3$, $zy3$, $z4x$, $z4y$, $z3x2y2$, $z3xy$, $z2x3$, $z2y3$, $z3x3$, $z3y3$, $z4x2y2$, $z4xy$, $z5x$, $z5y$. Shims can be adjusted from -32767 to $+32767$ (Varian 40 Shims choice in CONFIG window).

10 is a shim set in a Varian 14-shim supply with computer-controlled axial shims z1, z1c, z2, z2c, z3, z4, z5, and radial shims x1, y1, xz, yz, xy, x2y2, x3, y3. Shims can be adjusted from -2047 to +2047 (Varian 14 Shims choice in CONFIG window).

11 is a shim set in a Varian 8-shim supply with computer-controlled axial shims z1, z2, and radial shims x1, y1, xz, yz, xy, x2y2. Shims can be adjusted from -32767 to +32767 (Whole Body Shims choice in CONFIG window).

12 is a shim set in a Varian 26-shim supply with computer-controlled axial shims z1, z2, z3, z4, z5, and radial shims x1, y1, xz, yz, xy, x2y2, x3, y3, xz2, yz2, zxy, zx2y2, z3x, z3y, z2x2y2, z2xy, zx3, zy3, x4, y4. Shims can be adjusted from -32767 to +32767 (Varian 26 Shims choice in CONFIG window).

13 is a shim set in an Varian 29-shim supply with computer-controlled axial shims z1, z2, z3, z4, z5, z6, and radial shims x1, y1, xz, yz, xy, x2y2, x3, y3, xz2, yz2, zxy, zx2y2, z3x, z3y, z2x2y2, z2xy, zx3, zy3, z4x, z4y, z5x, z5y. Shims can be adjusted from -32767 to +32767 (Varian 29 Shims choice in CONFIG window).

14 is a shim set in a Varian 35-shim supply with computer-controlled axial shims z1, z2, z3, z4, z5, z6, and radial shims x1, y1, xz, yz, xy, x2y2, x3, y3, x4, y4, xz2, yz2, zxy, zx2y2, z3x, z3y, z2x2y2, z2xy, zx3, zy3, z4x, z4y, z3x2y2, z3xy, z4x2y2, z4xy, z5x, z5y. Shims can be adjusted from -32767 to +32767 (Varian 35 Shims choice in CONFIG window).

See also: *VNMR and Solaris Software Installation*

Related: `config` Display current configuration and possibly change it (M)

shimspath Path to user's shims directory (P)

Description: Contains an absolute path to a user's shims directory, which has files of shim settings. If `shimspath` exists for a user, it must be defined in the user's global parameter file. To create `shimspath`, enter:
`create('shimspath', 'string', 'global')`.

See also: *Getting Started*

Related: `rts` Retrieve shim coil settings (C)
`svs` Save shim coil settings (C)

showconsole Show ^{UNITY}INOVA console configuration parameters (U)

Applicability: ^{UNITY}INOVA and *MERCURY-VX* systems.

Syntax: (From UNIX) `showconsole`

Description: Displays console hardware configuration parameters and system versions. This information is recorded during console bootup and represents the system hardware options recognized by the acquisition computer. The command is used mainly when troubleshooting or performing diagnostics.

See also: *Getting Started*

Related: `ihwinfo` Hardware status of ^{UNITY}INOVA console (C)

showfit Display numerical results of deconvolution (M)

Syntax: `showfit`

Description: After a deconvolution, the results are written into file `fitspec.outpar` in an abbreviated format. `showfit` converts these data to an output format more suitable for examination and printing.

Alternate: Results button in the Deconvolution Menu.

See also: *User Guide: Liquids NMR*

Related: **fitspec** Perform spectrum deconvolution (C)
plfit Plot deconvolution analysis (M)
usemark Use “mark” output as deconvolution starting point (M)

showoriginal Restore first 2D spectrum in 3D DOSY experiment (M)

Syntax: `showoriginal`

Description: Restores the first 2D spectrum in a 3D DOSY experiment (if it has been saved by the **dosy** macro).

See also: *User Guide: Liquids NMR*

Related: **dosy** Process DOSY experiments (M)

showplotq Display plot jobs in plot queue (M)

Syntax: `showplotq`

Description: Displays current plot jobs in the plot queue for the active plotter in VNMR.

See also: *Getting Started*

Related: **killplot** Stop plot jobs and remove from plot queue (C)
showprintq Display print jobs in print queue (C)

showprintq Display print jobs in print queue (M)

Syntax: `showprintq`

Description: Displays current print jobs in the print queue for the active printer in VNMR.

See also: *Getting Started*

Related: **killprint** Stop print jobs and remove from print queue (C)
showplotq Display plot jobs in plot queue (M)

showstat Display information about status of acquisition (M,U)

Syntax: (From VNMR) `showstat<(remote_system)>`
(From UNIX) `showstat <remote_system>`

Description: Displays information in the text screen about the status of acquisition on a spectrometer. The command is similar to **Acqstat**, but displays the information in a non-graphical manner and only once.

Arguments: `remote_system` is the host name of a remote spectrometer. The default is to display information about acquisition on the local system.

See also: *Getting Started*

Related: **Acqstat** Bring up the acquisition status display (U)

sin Find sine value of an angle (C)

Syntax: `sin(angle)<:n>`

Description: Finds the sine value of an angle.

Arguments: `angle` is the angle given in radians.

`n` is a return value giving the sine of `angle`. The default is to display the sine value in the status window.

Examples: `sin(.5)`
`sin(val):sin_val`

See also: *VNMR User Programming*

Related: `acos` Find arc cosine of number (C)
`arccos` Calculate arc cosine of real number (M)
`arcsin` Calculate arc sine of real number (M)
`arctan` Calculate arc tangent of real number (M)
`asin` Find arc sine of number (C)
`atan` Find arc tangent of a number (C)
`cos` Find cosine value of an angle (C)
`exp` Find exponential value (C)
`ln` Find natural logarithm of a number (C)
`tan` Find tangent value of an angle (C)

sine Find values for a sine window function (M)

Syntax: `sine<(shift<, number_points<, domain)>>`

Description: Calculates appropriate values for parameters `sb` and `sbs` (if the domain argument is 'f2') or for parameters `sb1` and `sbs1` (if the domain argument is 'f1') in order to achieve a sine window function. The value of the parameter `trace` is used if the domain argument is not entered.

Arguments: `shift` sets the starting value for the window function. If `shift` is greater than 0, the starting value is given by `sin p/shift`; if `shift` is less than or equal to 0, the starting value is 0. The default value is 0.

`number_points` specifies the number of real points that the window function spans. The value of the window function for subsequent points is 0.

`number_points` must be greater than 0 and a multiple of 2. The default is `ni*2` if `trace='f1'`, or `np` if `trace='f2'`.

`domain` is 'f1' or 'f2'. The default is the current setting of `trace`.

See also: *User Guide: Liquids NMR*

Related: `np` Number of data points (P)
`sb` Sinebell const. in directly detected dimension (P)
`sb1` Sinebell const. in 1st indirectly detected dimension (P)
`sbs` Sinebell shift const. in directly detected dimension (P)
`sbs1` Sinebell shift const. in 1st indirectly detected dimension (P)
`sinesq` Find values for a sine squared window function (M)
`trace` Mode for *n*-dimensional data display (P)

sinebell Select default parameters for sinebell weighting (M)

Syntax: `sinebell`

Description: Generates initial guess at good sinebell weighting parameters by setting the `sb` and `sb1` parameters to one-half the acquisition time and turning off all other weighting. Use `sinebell` in absolute-value 2D experiments only.

Alternate: Sinebell button in the 2D Processing Parameter Setup Menu.

See also: *User Guide: Liquids NMR*

Related: `pseudo` Set default parameters for pseudo-echo weighting (M)
`sb` Sinebell const. in directly detected dimension (P)
`sb1` Sinebell const. in 1st indirectly detected dimension (P)

sinesq Find values for a sine-squared window function (M)

Syntax: `sinesq<(shift<,number_points<,domain)>>`

Description: Calculates appropriate values for parameters `sb` and `sbs` (if the domain argument is 'f2') or for parameters `sb1` and `sbs1` (if the domain argument is 'f1') in order to achieve a sine-squared window function. The value of parameter `trace` is used if the domain argument is not entered.

Arguments: `shift` sets the starting value for the window function. If `shift` is greater than 0, the starting value is given by $\sin p/\text{shift}$; otherwise, if `shift` is less than or equal to 0, the starting value is 0. The default value is 0.

`number_points` specifies the number of real points that the window function spans. The value of the window function for subsequent points is 0. The `number_points` argument must be greater than 0 and a multiple of 2. The default is `ni*2` if `trace='f1'`, or `np` if `trace='f2'`.

`domain` is 'f1' or 'f2'. The default is the current setting of `trace`.

See also: *User Guide: Liquids NMR*

Related:	<code>ni</code>	Number of increments in 1st indirectly detected dimension (P)
	<code>np</code>	Number of data points (P)
	<code>sb</code>	Sinebell const. in directly detected dimension (P)
	<code>sb1</code>	Sinebell const. in 1st indirectly detected dimension (P)
	<code>sbs</code>	Sinebell shift const. in directly detected dimension (P)
	<code>sine</code>	Find values for a sine window function (M)
	<code>trace</code>	Mode for <i>n</i> -dimensional data display (P)

slamp Measured line amplitudes (obsolete)

Description: This parameter is no longer used.

slfreq Measured line frequencies (P)

Description: Contains a list of measured line frequencies. In iterative spin simulation, a calculated spectrum is matched to the lines in the list. The `spinll` macro fills in `slfreq` from the last line listing or a `mark` operation. Use `assign` to make assignments between the measured lines and the calculated transitions. `slfreq` is a global parameter and is displayed by `dla`.

See also: *User Guide: Liquids NMR*

Related:	<code>assign</code>	Assign transitions to experimental lines (M)
	<code>cla</code>	Clear all line assignments (M)
	<code>dla</code>	Display spin simulation parameter arrays (M)
	<code>fitspec</code>	Perform spectrum deconvolution (C)
	<code>mark</code>	Determine intensity of a spectrum at a point (C)
	<code>spinll</code>	Set up an <code>slfreq</code> array (M)

sliceorder Reorder the slice position list (M)

Applicability: Systems with the imaging capabilities.

Syntax: `sliceorder<('a'|'d'|'i')>`

Description: Reorders the slice position list, `pss`, in ascending, descending, or alternating odd/even order.

Alternating order is often used for multislice excitation to separate physically adjacent slices in time to reduce saturation effects. For example, if `pss=-3,-2,-1,0,1,2,3` is reordered by alternating odd/even order, the result is

`pss=-3,-1,1,3,-2,0,2` so that the adjacent slices `-1` and `-2`, for example, are separated by three time intervals instead of just one.

Arguments: `'a'` is a keyword to reorder the list in alternating odd/even order. This is the default.

`'d'` is a keyword to reorder the list in descending order.

`'i'` is a keyword to reorder the list in ascending order.

Examples: `sliceorder('d')`

See also: *User Guide: Imaging*

Related: `pss` Slice position (P)

sliceplan Set slice parameters for target slice (M)

Applicability: Systems with imaging capabilities.

Syntax: `sliceplan`

Description: Calculates and sets the slice parameters for the target slice defined in the file `curexp+' /mark2d.out'`. The slice parameters (i.e., `pss`, `psi`, `phi`, `theta`) are calculated and set by using `sliceplan`. The Calculate Target button of the slice planner menu also uses `sliceplan`. See the `plan` macro for further details.

See also: *User Guide: Imaging*

Related: `curexp` Current experiment directory (P)
`drawslice` Display target slices (M)
`drawvox` Display target voxels (M)
`plan` Display menu for planning a target scan (M)
`voxplan` Set voxel parameters for voxel defined by 2D box cursor (M)

slp Family of offset Frequencies of SLP shapes (P)

Applicability: Systems with LC-NMR or VAST accessory.

Syntax: `slp(frequency offset from the trans transmitter)`

Description: Specifies frequencies, in Hz, of Shifted Laminar Pulses (SLP) shapes used for suppression of solvent peaks. There are 6 members of the `slp` family, `slp0` (solvent 1), `slp` (solvent 2), `slp2` (solvent 3), `slp3` (solvent 4), `slp4` (solvent 5), `slp6` (solvent 6), and `slp6` (solvent 7). There is no `slp1` parameter.

slw Spin simulation linewidth (P)

Description: Sets linewidth for individual transitions in the displayed spectrum. Only one linewidth is provided, so all transitions must be given the same linewidth. If the Set Params button is used in setting up spin simulation parameters, `slw` is automatically set to the measured linewidth of the tallest line displayed on the screen.

`slw` is also the starting default linewidth for deconvolution calculations. This linewidth will be set automatically when deconvolution is operated using the menu mode and is bypassed if the `usemark` command has been used in conjunction with two cursor input.

Values: 0.01 to 1e6. The typical value is 1.

See also: *User Guide: Liquids NMR*

Related: `usemark` Use "mark" output as deconvolution starting point (M)

small **Use small graphics window (C)**

Syntax: `small`

Description: Sets the Sun graphics window to a partial screen, which allows room for the text window and the acquisition window. `small` is only executed after any other commands have been processed, and any current display is lost and has to be recalculated.

Alternate: Small button in the Permanent Menu.

See also: *Getting Started*

Related: `large` Use large graphics window (C)

smaxf **Maximum frequency of any transition (P)**

Description: Sets the maximum frequency limit for the calculation of the final simulated spectrum. It should be set before the calculation is performed. If the Set Params button is used in setting up spin simulation parameters, `smaxf` is initialized to `sp+wp`; which assumes that you have already expanded the region of the spectrum that you wish to simulate before beginning the spin simulation process.

Values: $-1e10$ to $1e10$, in Hz. The typical value is the maximum chemical shift + 50.

See also: *User Guide: Liquids NMR*

Related: `sminf` Minimum frequency of any transition (P)
`sp` Start of plot (P)
`wp` Width of plot (P)

sminf **Minimum frequency of any transition (P)**

Description: Sets the minimum frequency limit for the calculation of the final simulated spectrum. It should be set before the calculation is performed. If the Set Params button is used in setting up spin simulation parameters, `sminf` is initialized to `sp`, which assumes that you have already expanded the region of the spectrum that you wish to simulate before beginning the spin simulation process.

Values: $-1e10$ to $1e10$, in Hz. The typical value is 0.

See also: *User Guide: Liquids NMR*

Related: `smaxf` Maximum frequency of any transition (P)
`sp` Start of plot (P)
`wp` Width of plot (P)

smsport **Sample Management System serial port connection (P)**

Applicability: ^{UNITY}*INOVA* systems only.

Description: Sets which serial port on the host computer is connected to a Sample Management System (i.e., a sample changer). The value of `smsport` is set using the Sample Changer Serial Port label in the CONFIG window (opened from `config`).

Values: 'a' sets the connection for serial port A. This value is the default.
'b' sets the connection for serial port B.

See also: *VNMR and Solaris Software Installation; User Guide: Liquids NMR*

Related: `config` Display current configuration and possibly change it (M)

sn Signal-to-noise ratio (P)

Description: Sets a ratio for testing signal-to-noise. The `testsn` macro checks whether a signal-to-noise ratio equal to `sn` has been achieved.

Values: Typical value is 35.

See also: *User Guide: Liquids NMR*

Related: `dsn` Measure signal-to-noise (C)
`getsn` Get signal-to-noise estimate of a spectrum (M)
`testsn` Test signal-to-noise of a spectrum (M)
`testct` Check `ct` for resuming signal-to-noise testing (M)

solppm Return ppm and peak width of solvent resonances (M)

Syntax: `solppm:chemical_shift,peak_width`

Description: Returns to the calling macro information about the chemical shift and peak spread of solvent resonances in various solvents for either ^1H or ^{13}C , depending on the observe nucleus `tn` and the parameter `solvent`. This macro is used “internally” by other macros only.

Arguments: `chemical_shift` returns the chemical shift of the solvent in ppm.

`peak_width` returns the approximate peak spread of solvent resonances.

See also: *VNMR User Programming*

Related: `solvent` Lock solvent (P)
`tn` Nucleus for observe transmitter (P)

solvent Lock solvent (P)

Description: Contains one of a series of lock solvents from the `/vnmr/solvents` file, which contains the ^2H chemical shift of each lock solvent. By editing the file, additional solvents can be added. Values for `solvent` are not case-sensitive (e.g., `solvent='C6D6'` and `solvent='c6d6'` are identical)

The `auto_dir` macro now controls most of the automation features, including setting the value of `solvent`.

Values: Standard values in `/vnmr/solvents` include:

Deuterium Oxide	CDCI3	MethyleneChloride
D2O	Cyclohexane	MethylAlcohol-d4
Acetone	C6D12	CD2C12
CD3COCD3	Toluene	CD3OD
Benzene	C6D5CH3	Chloroform
C6D6	Acetic_Acid	
DMSO	CD3COOD	

See also: *Getting Started*

Related: `auto_dir` Controlling macro for automation (M)
`lastlk` Last lock solvent used (P)
`solvinfo` Retrieve information from solvent table (C)
`tof` Frequency offset for observe transmitter (P)

solvfactor Solvent correction factor (obsolete)

Description: This parameter has been removed from VNMR because a change in the method of setting the frequency makes it unnecessary.

See also: *Getting Started*

Related: `setfrq` Set frequency of rf channels (C)

solvinfo Retrieve information from solvent table (C)

Syntax: `solvinfo(solvent):$chemical_shift,$name`

Description: Retrieves solvent shift and solvent name from the solvent table.

Arguments: `solvent` is the name of a solvent from the `/vnmr/solvents` file. This argument is not case-sensitive (e.g., 'c6d6' is the same as 'C6D6').
`chemical_shift` returns the chemical shift of the solvent, in ppm.
`name` returns the name of the solvent. The name returned will match the case of the letters (upper or lower) in `/vnmr/solvents`.

Examples: `solvinfo('acetone'):$shift`
`solvinfo('d2o'):$shift,solvent`

See also: *Getting Started*

Related: `lookup` Look up words and lines from a text file (C)
`solvent` Lock solvent (P)

sp Start of plot in directly detected dimension (P)

Description: Low-frequency limit of the display or plotted region of the spectrum. `sp` is always stored in Hz, but can be entered in ppm by using the `p` suffix (e.g., `sp=2p` sets the start of plot to 2 ppm).

See also: *Getting Started; User Guide: Liquids NMR*

Related: `sp1` Start of plot in 1st indirectly detected dimension (P)
`sp2` Start of plot in 2nd indirectly detected dimension (P)

sp1 Start of plot in 1st indirectly detected dimension (P)

Description: Analogous to the `sp` parameter except that `sp1` applies to the first indirectly detected dimension of a multidimensional data set.

See also: *User Guide: Liquids NMR*

Related: `sp` Start of plot in directly detected dimension (P)
`sp2` Start of plot in 2nd indirectly detected dimension (P)

sp2 Start of plot in 2nd indirectly detected dimension (P)

Description: Analogous to the `sp` parameter except that `sp2` applies to the second indirectly detected dimension of a multidimensional data set.

See also: *User Guide: Liquids NMR*

Related: `sp` Start of plot in directly detected dimension (P)

spadd Add current spectrum to add/subtract experiment (C)

Syntax: (1) `spadd<(multiplier<,>,shift>)>`
 (2) `spadd('new')`
 (3) `spadd('trace',index)`

Description: Performs noninteractive spectral addition. The last displayed or selected spectrum is added to the current contents of the add/subtract experiment (`exp5`). A multi-element add/subtract experiment can be created using the

'new' keyword. Individual spectra in a multi-element add/subtract experiment can be subsequently added to using the 'trace' keyword followed by an index number of the spectrum.

Arguments: `multiplier` is a value to multiply each spectrum being added to the add/subtract experiment (`exp5`). The normal range of `multiplier` would be +1 to -1 but the range is actually unlimited. The default is 1.0.

`shift` is the number of data points to shift each spectrum. A positive value shifts the spectrum being added to a higher frequency, or to the left. A negative value shifts the spectrum to a lower frequency, or to the right. The default is 0.

'new' is a keyword to create a new spectrum in the add/subtract experiment.

'trace' is a keyword to select the spectrum given by the index number argument (`index`) and add it to the add/subtract experiment. The default is to add to the first spectrum in the add/subtract experiment.

`index` is the index number of the spectrum to be used as a target in a multi-element add/subtract experiment.

Examples: `spadd`
`spadd(.5 , 25)`
`spadd('new')`
`spadd('trace' , 2)`

Alternate: Add Spectrum button in the Add/Subtract Menu.

See also: *User Guide: Liquids NMR*

Related:	<code>add</code>	Add current FID to add/subtract experiment (C)
	<code>addi</code>	Start interactive add/subtract mode (C)
	<code>clradd</code>	Clear add/subtract experiment (C)
	<code>ds</code>	Display a spectrum (C)
	<code>jexp</code>	Join existing experiment (C)
	<code>select</code>	Select a spectrum without displaying it (C)
	<code>spmin</code>	Take minimum of two spectra in add/subtract experiment (C)
	<code>spsub</code>	Subtract current spectrum from add/subtract experiment (C)

`spcfrq` **Display frequencies of rf channels (M)**

Description: Displays the parameters `sfrq`, `dfrq`, `dfrq2`, and `dfrq3` with seven decimal points (to nearest 0.1) to provide the exact frequencies of each rf channel. The number of values displayed depends on `numrfch`.

Prior to VNMR version 4.3, `spcfrq` set the frequency of the observe channel. The parameter `sfrq` now sets the frequency instead of `spcfrq`.

See also: *Getting Started*

Related:	<code>dfrq</code>	Transmitter frequency of first decoupler (P)
	<code>dfrq2</code>	Transmitter frequency of second decoupler (P)
	<code>dfrq3</code>	Transmitter frequency of third decoupler (P)
	<code>numrfch</code>	Number of rf channels (P)
	<code>setfrq</code>	Set frequency of rf channels
	<code>sfrq</code>	Transmitter frequency of observe nucleus (P)

`specdc3d` **3D spectral dc correction (P)**

Description: Sets whether a 3D spectral dc correction occurs. The spectral dc correction is the last operation to be performed upon the data prior to forming linear combinations of the data, using the coefficients in the 3D coefficient file

(coef), and then writing the data to disk. If `specdc3d` does not exist, it is created by the macro `par3d`.

Values: A three-character string selected from 'nnn', 'nny', 'nyn', etc. Each character may take one of two values: n for no spectral dc correction along the relevant dimension, and y for spectral dc correction along the relevant dimension. The first character refers to the f_3 dimension (`sw`, `np`, `fn`), the second character refers to the f_1 dimension (`sw1`, `ni`, `fn1`), and the third character refers to the f_2 dimension (`sw2`, `ni2`, `fn2`). The default is 'nnn'.

See also: *User Guide: Liquids NMR*

Related:	<code>dc</code>	Calculate spectral drift correction (C)
	<code>fiddc3d</code>	3D time-domain dc correction (P)
	<code>fn</code>	Fourier number in directly detected dimension (P)
	<code>fn1</code>	Fourier number in 1st indirectly detected dimension (P)
	<code>fn2</code>	Fourier number in 2nd indirectly detected dimension (P)
	<code>ft3d</code>	Perform a 3D Fourier transform (M)
	<code>ni</code>	Number of increments in 1st indirectly detected dimension (P)
	<code>ni2</code>	Number of increments in 2nd indirectly detected dimension (P)
	<code>np</code>	Number of data points (P)
	<code>par3d</code>	Create 3D acquisition, processing, display parameters (C)
	<code>ptspec3d</code>	Region-selective 3D processing (P)
	<code>sw</code>	Spectral width in directly detected dimension (P)
	<code>sw1</code>	Spectral width in 1st indirectly detected dimension (P)
	<code>sw2</code>	Spectral width in 2nd indirectly detected dimension (P)

spin **Submit a spin setup experiment to acquisition (C)**

Applicability: All systems; however, it applies to *GEMINI 2000* only if spin automation hardware is installed.

Syntax: `spin`

Description: Regulates sample spinning according to the *parameter spin*, using the acquisition computer. It also sets rf frequency, decoupler status, and temperature.

See also: *Getting Started*

Related:	<code>au</code>	Submit experiment to acquisition and process data (C)
	<code>change</code>	Submit a change sample experiment to acquisition (M)
	<code>ga</code>	Submit experiment to acquisition and FT the result (C)
	<code>go</code>	Submit experiment to acquisition (C)
	<code>lock</code>	Submit an Autolock experiment to acquisition (C)
	<code>sample</code>	Submit change sample, autoshim experiment to acquisition (M)
	<code>shim</code>	Submit an Autoshim experiment to acquisition (C)
	<code>spin</code>	Sample spin rate (P)
	<code>su</code>	Submit a setup experiment to acquisition (M)

spin **Sample spin rate (P)**

Applicability: All systems; however, it applies to *GEMINI 2000* only if spin automation hardware is installed (if not installed, the value of `spin` is ignored).

Description: Selects a regulated spin rate. The rate is changed when a sample is inserted or `spin`, `go`, `ga`, `au`, or `sample` are entered.

Values: 0 indicates non-spinning operation.

5 to 39 are spinning rates.

'n' leaves the spin rate at the currently used value and does not wait for regulated spinning before performing acquisition.

See also: *Getting Started*

Related:	au	Submit experiment to acquisition and process data (C)
	ga	Submit experiment to acquisition and FT the result (C)
	go	Submit experiment to acquisition (C)
	sample	Submit change sample, Autoshim experiment to acquisition (M)
	sethw	Set values for hardware in acquisition system (C)
	spin	Submit a spin setup experiment to acquisition (C)
	spinopt	Spin automation (P)

spinll **Set up a slfreq array (M)**

Syntax: `spinll<'mark'>`

Description: Copies a list of frequencies to the **slfreq** parameter in iterative spin simulation and runs **dla**. This macro also clears previous line assignments.

Arguments: 'mark' is a keyword to copy the list of frequencies from the `markld.out` file to **slfreq**. The default is to copy the frequencies from the last line listing by **nll** or **dll** to the **slfreq**. Use the cursor and the mark button to place the lines to be assigned in `markld.out`. Enter **mark**('reset') to clear the file, and use **nl** to move the cursor to the center of a selected line.

Alternate: use **ll** button in the Spin Simulation Line Assignment Menu.

See also: *User Guide: Liquids NMR*

Related:	dla	Display line assignments (M)
	dll	Display listed line frequencies and intensities (C)
	mark	Determine intensity of the spectrum at a point (C)
	nl	Position the cursor at the nearest line (C)
	nll	Find line frequencies and intensities (C)
	slfreq	Measured line frequencies (P)

spinner **Open the Spinner Control window (C)**

Applicability: All systems except *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Syntax: `spinner`

Description: Opens the Spinner Control window. This window has the following capabilities:

- Turn the sample spinner off.
- Turn the sample spinner on at a specified speed, in Hz.
- Enable spinner control from within an experiment using the **spin** parameter and the **spin**, **go**, **ga**, or **au** commands. This mode is the default.
- Alternatively, turn off experiment control of the sample spinner and allow only the Spinner Control window (and **acqi** and **sethw**) to set the spinning speed. This mode has the advantage that, often times, the **spin** parameter is different between experiments. Joining a different experiment and entering **go** can unexpectedly change the spinning speed. This alternate mode prevents this problem. In this mode, when a **go**, **su**, **ga**, or **au** is entered, the **spin** parameter is first set to the speed selected in the Spinner Control window and then the **spin** parameter is set to "Not Used."
- Select the style of spinner: low-speed style or a high-speed style. If the high-speed style of spinner (used for solids) is selected, the choice of

setting the spinning speed or the air flow rate is provided. Setting the air flow rate is useful when setting up the solids spinning apparatus.

If the spinning speed is controlled only through the Spinner Control window, the action to be taken after a spinner error can be selected:

- Display a warning but continue acquisition.
- Stop acquisition and display a warning.

If experiment control of spinning speed is selected, these selections are faded because they are inoperative, and the selection of the action to be taken after a spinning speed error is provided by the parameter `in`.

See also: *Getting Started*

Related:	<code>acqi</code>	Interactive acquisition display process (C)
	<code>au</code>	Submit experiment to acquisition and process data (C)
	<code>change</code>	Submit a change sample experiment to acquisition (M)
	<code>ga</code>	Submit experiment to acquisition and FT the result (C)
	<code>go</code>	Submit experiment to acquisition (C)
	<code>in</code>	Lock and spin interlock (P)
	<code>lock</code>	Submit an Autolock experiment to acquisition (C)
	<code>sample</code>	Submit change sample, autoshim experiment to acquisition (M)
	<code>sethw</code>	Set values for hardware in acquisition system (C)
	<code>shim</code>	Submit an Autoshim experiment to acquisition (C)
	<code>spin</code>	Sample spin rate (P)
	<code>su</code>	Submit a setup experiment to acquisition (M)

spinopt Spin automation (P)

Applicability: *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems.

Description: Specifies whether spin hardware is installed on the system. The value is set by the Auto Spinner label in the CONFIG window (opened from `config`). For *MERCURY-VX*, the value must be 'y'.

Values: 'n' for no spin hardware is installed (Not Present choice in the CONFIG window).

'y' for spin hardware is installed (Present choice in the CONFIG window).

See also: *VNMR and Solaris Software Installation*

Related: `config` Display current configuration and possible change it (M)

spins Perform spin simulation calculation (C)

Syntax: `spins<(options)>`

Description: Performs a spin simulation, using the current spin system parameters. Refer to the description of `spsm` for setting up the parameters. Use `dsp` to display the spectrum resulting from the simulation. The output file is `spins.list` in the current experiment. This file includes the calculated transitions ordered by frequency and is most easily displayed by the list button in the Spin Simulation Secondary menu.

Line assignments are required for the iteration. These consist of a list of observed frequencies, which is stored in the arrayed parameter `slfreq`, and the line assignments stored in the array `clindex`. `spinll` copies the frequencies from the last line listing by `nll` or `dll` into the parameter `slfreq`. The line listing can be from an observed spectrum or from the results of deconvolution. After `spinll`, line assignments are most easily made by entering `assign` or by using the Spin Simulation Line Assignment menu. `dla`

displays the assignments. Single assignments can also be made by `assign(transition_number, line_number)`, where `transition_number` is the index of a transition and `line_number` is the index of the measured line. Setting the `line_number` argument to 0 deletes assignments. `dla('long')` produces an expanded display of assignments.

Be aware that spin simulation line numbers and line list line numbers are *not* the same. Conventional line lists produced by `dll` number the lines from left to right (low- to high-field). The spin simulation software numbers lines according to a more complicated scheme, and these numbers are rarely if ever in frequency order.

The parameters to be iterated are chosen by setting the string parameter `iterate` (e.g., `iterate='A,B,JAB'`). If several parameters have the same value due to symmetry, use `iterate='A,B,C,JAB,JAC=JAB'`. This string sets the iterated parameter JAC to JAB during the iteration. JAB must be defined as an iterated parameter in the string before it can be used at the right side of the equal sign. Sets of parameters with up to six members may be set up in this way. The member in the set that is used on the right side of the equal sign must always come first in the parameter display (e.g., JAB=JAC would be wrong). A parameter is held constant during iteration if it is not included in the `iterate` string.

The command `initialize_iterate` sets `iterate` to iterate all spins not named X, Y, or Z and the associated coupling constants.

Following an iterative spin simulation, `dga` displays the new values of the coupling constants and chemical shifts. `undospins` restores a spin system as it was before the last iterative run. It returns the chemical shifts, coupling constants, and line assignments, making it possible to continue from this state with modified line assignments.

Note that major changes in the starting values of parameters may change the numbering of the energy levels and hence the line numbers. The line assignments would then be incorrect and would have to be reentered.

For a successful iteration, it is often necessary to keep some parameters fixed. For example, it is sometimes useful to alternately iterate couplings and shifts, keeping one group fixed while the other is iterated independently.

Arguments: The following variations of `spins` are available:

- `spins('calculate','energy')` puts an energy-level table in the output file.
- `spins('calculate','transitions')` puts a second table of transitions ordered by transition number in the output file.
- `spins('display')` and `dsp` are equivalent.
- `spins('system','spinsystemname')` and `spsm('spinsystemname')` are equivalent.
- `spins('iterate')` runs interactively to match experimental and calculated lines.
- `spins('iterate','iteration')` lists parameters after each iteration in the output file.
- `spins('iterate'<,options>)` provides for determining the chemical shifts and coupling constants to produce a spectrum that matches a table of observed lines. `spins` iterates until the rms (root-mean-square) error of the line matching meets a built-in test, unless it first reaches the value given by `number_iterations`. Iteration also stops if the rms error increases.

- Put multiple list options into the second argument, separated by a blank (e.g., `spins('calculate', 'transitions energy')`).

Examples: `spins`
`spins('calculate', 'energy')`
`spins('iterate')`

See also: *User Guide: Liquids NMR*

Related:	<code>assign</code>	Assign transitions to experimental lines (M)
	<code>clindex</code>	Index of experimental frequency of a transition (P)
	<code>dga</code>	Display parameter groups (spin simulation) (C)
	<code>dla</code>	Display line assignments (M)
	<code>dll</code>	Display listed line frequencies and intensities (C)
	<code>dsp</code>	Display calculated spectrum (C)
	<code>initialize_iterate</code>	Set <code>iterate</code> to contain relevant parameters (M)
	<code>iterate</code>	Parameters to be iterated (P)
	<code>niter</code>	Number of iterations (P)
	<code>nll</code>	Find line frequencies and intensities (C)
	<code>slfreq</code>	Measured line frequencies (P)
	<code>spinll</code>	Set up <code>slfreq</code> array (M)
	<code>spsm</code>	Enter spin system (M)
	<code>undospins</code>	Restore spin system as before last iterative run (M)

split **Split difference between two cursors (M)**

Syntax: `split`

Description: Repositions the left-hand cursor halfway between its original position and the position of the other cursor. This macro is very useful for finding the center of a powder pattern: place the two cursors on the horns of the pattern and then enter `split` to give the center.

See also: *Getting Started; UNITYplus Solid-State NMR Hardware Installation; UNITY INOVA Solids Hardware Installation*

Related: `delta` Difference of two frequency cursors (P)

spmin **Take minimum of two spectra in add/subtract experiment (C)**

Syntax: `spmin`

Description: Takes the minimum of two spectra, considered point-by-point in an absolute-value sense. For example, if the two corresponding values are -2 and $+3$, the `spmin` spectrum will have -2 ; if the two values are $+2$ and -3 , the `spmin` spectrum will have $+2$ at that point.

The function of `spmin` is to essentially select for common features within two spectra while eliminating features that are not common between them. In particular, if two CP/MAS spectra are obtained at different spin rates, the peaks stay in the same place (and hence the `spmin` spectrum also contains the same peaks), but the sidebands move. If spectrum 1 has baseline where spectrum 2 has sideband, and spectrum 2 has baseline where spectrum 1 has sideband, then the `spmin` spectrum will contain only baseline in these regions, eliminating the spinning sidebands.

Alternate: Minimum button in the Add/Subtract Menu.

See also: *User Guide: Liquids NMR*

Related:	<code>addi</code>	Start interactive add/subtract mode (C)
	<code>spadd</code>	Add current spectrum to add/subtract experiment (C)
	<code>spsub</code>	Subtract current spectrum from add/subtract experiment (C)

spsm **Enter spin system (M)**

Syntax: `spsm(spin_system)`

Description: Enables entry of the spin system for spin simulation and creates and initializes the appropriate parameters to describe the various chemical shifts and coupling constants. Chemical shifts can be entered for the X-nucleus, and the spectrum is calculated if that shift is in the window. Generally, however, it is not necessary to enter the X-nucleus chemical shift, and its value has no effect on the spectrum of the remainder of the spin system.

Arguments: `spin_system` is an alphanumeric string of upper-case letters for chemical shift and coupling constant parameters. Chemical shifts are stored in parameters A through Z, and the coupling constants are stored in the parameters starting with JAB and ending with JYZ. Different nucleus types are handled by using letters starting with A for the first type, X for the second, and M for the third. Once created, these parameters are entered and modified in the usual way (e.g., A=78.5 JAC=5.6). Entry of chemical shifts in ppm is entered by using `sfrq` (e.g., B=7.5*sfrq).

Examples: `spsm('AB')`
`spsm('A3B2')`
`spsm('AB2CMXY')`

See also: *User Guide: Liquids NMR*

Related: `sfrq` Transmitter frequency of observe nucleus (P)
`spins` Perform spin simulation calculation (C)

spsub **Subtract current spectrum from add/subtract experiment (C)**

Syntax: (1) `spsub<(multiplier<,shift>)>`
(2) `spsub('new')`
(3) `spsub('trace',index)`

Description: Performs non-interactive spectral subtraction. The last displayed or selected spectrum is subtracted from the current contents of the add/subtract experiment (`exp5`). A multi-element add/subtract experiment can be created using the 'new' keyword. Individual spectra in a multi-element add/subtract experiment can be subsequently subtracted from using the 'trace' keyword followed by an index number of the spectrum.

Arguments: `multiplier` is a value to multiply each spectrum being subtracted from the add/subtract experiment (`exp5`). The normal range of `multiplier` would be +1 to -1 but is actually unlimited. The default is 1.0.

`shift` is the number of data points to shift each spectrum. A positive value shifts the spectrum being added to a higher frequency, or to the left. A negative value shifts the spectrum to a lower frequency, or to the right. The default is 0.

'new' is a keyword to create a new spectrum in the add/subtract experiment.

'trace' is a keyword to select the spectrum given by the index number argument (`index`) and subtract it from the add/subtract experiment. The default is to subtract from the first spectrum in the add/subtract experiment.

`index` is the index number of the spectrum to be used as a target in a multi-element add/subtract experiment.

Examples: `spsub`
`spsub(.5,25)`
`spsub('new')`
`spsub('trace',2)`

Alternate: Subtract button in the Add/Subtract Menu.

See also: *User Guide: Liquids NMR*

Related:	<code>clradd</code>	Clear add/subtract experiment (C)
	<code>ds</code>	Display a spectrum (C)
	<code>jexp</code>	Join existing experiment (C)
	<code>spadd</code>	Add current spectrum to add/subtract experiment (C)
	<code>select</code>	Select a spectrum without displaying it (C)
	<code>spmin</code>	Take minimum of two spectra in add/subtract experiment (C)
	<code>sub</code>	Subtract current FID from add/subtract experiment (C)

sqcosine **Set up unshifted cosine-squared window function (M)**

Syntax: `sqcosine(<t1_inc><,t2_inc>)>`

Description: Sets up an unshifted cosine-squared window function in 1, 2, or 3 dimensions. The macro checks whether the data is 1D, 2D, and 3D.

Arguments: `t1_inc` is the number of t1 increments. The default is `ni`.
`t2_inc` is the number of t2 increments. The default is `ni2`.

See also: *Getting Started; User Guide: Liquids NMR*

Related:	<code>gaussian</code>	Set up unshifted Gaussian window function (M)
	<code>ni</code>	Number of increments in 1st indirectly detected dimension (P)
	<code>ni2</code>	Number of increments in 2nd indirectly detected dimension (P)
	<code>pi3ssbsq</code>	Set up pi/3 shifted sinebell-squared window function (M)
	<code>pi4ssbsq</code>	Set up pi/4 shifted sinebell-squared window function (M)
	<code>sq sinebell</code>	Set up unshifted sinebell-squared window function (M)

sq sinebell **Set up unshifted sinebell-squared window function (M)**

Syntax: `sq sinebell(<t1_inc><,t2_inc>)>`

Description: Sets up an unshifted sinebell-squared window function in 1, 2, or 3 dimensions. The macro checks whether the data is 1D, 2D, and 3D.

Arguments: `t1_inc` is the number of t1 increments. The default is `ni`.
`t2_inc` is the number of t2 increments. The default is `ni2`.

See also: *Getting Started; User Guide: Liquids NMR*

Related:	<code>gaussian</code>	Set up unshifted Gaussian window function (M)
	<code>ni</code>	Number of increments in 1st indirectly detected dimension (P)
	<code>ni2</code>	Number of increments in 2nd indirectly detected dimension (P)
	<code>pi3ssbsq</code>	Set up pi/3 shifted sinebell-squared window function (M)
	<code>pi4ssbsq</code>	Set up pi/4 shifted sinebell-squared window function (M)
	<code>sqcosine</code>	Set up unshifted cosine-squared window function (M)

srate **Spinning rate for magic angle spinning (P)**

Applicability: Systems with solids module.

Description: Set to the spinning speed for magic angle spinning (MAS). `srate` must be correct for the pulse sequence set up by `xpolar` to run TOSS or dipolar dephasing correctly. If `hsrotor='y'`, the measured spinning speed is reported in `srate` for systems that have rotor synchronization.

Values: 0 to 10^7 , in Hz.

See also: *User Guide: Solid-State NMR*

Related:	<code>hsrotor</code>	Display rotor speed for solids operation (P)
	<code>xpolar</code>	Set up parameters for XPOLAR pulse sequence (M)

sread **Read converted data into VNMR (C)**

Syntax: `sread(file<,template>)`

Description: Reads 32-bit data files into VNMR. For Bruker data files in the AMX and AM formats, each file must first be converted using the `convertbru` command before `sread` can read the data in the file into VNMR.

Arguments: `file` is the name of a file containing data converted using `convertbru`.
`template` is the full path of a parameter template file, but without appending the `.par` extension on the file name. The default is `bruker.par`. If no parameter template is specified and `bruker.par` cannot be found in the user or system `parlib` directory, `sread` aborts with an error message.

Examples: `sread('brudata.cv', '/vnmr/parlib/bruker')`

See also: *Getting Started*

Related: `convertbru` Convert Bruker data (M,U)

ss **Steady-state transients (P)**

Description: Sets the number of complete executions of the pulse sequence not accompanied by data collection prior to the acquisition of the real data (sometimes known as *dummy scans*). If `ss` is positive, `ss` steady-state transients are applied on the first increment only, and if `ss` is negative, `-ss` steady-state transients are applied at the start of each increment.

Values: 'n', -32768 to 32767

See also: *Getting Started; VNMR User Programming*

ss3d **f₃ solvent subtraction option (obsolete)**

Description: Obsolete because solvent subtraction processing options `zfs` and `lfs` are now selected in 3D by setting `ssorder` and `ssfilter`, the same as in 1D and 2D.

Related: `ssfilter` Full bandwidth of digital filter to yield a filtered FID (P)
`ssorder` Order of polynomial to fit digitally filtered FID (P)

ssecho **Set up solid-state echo pulse sequence (M)**

Applicability: Systems with a solids module. Not supplied with *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Syntax: `ssecho`

Description: Converts a standard two-pulse experiment to a ready-to-run solid-state NMR echo (SSECHO) pulse sequence.

Alternate: SSECHO button in the 1D Pulse Sequence Setup Secondary menu.

See also: *User Guide: Solid-State NMR*

ssecho1 **Set up parameters for SSECHO1 pulse sequence (M)**

Applicability: *UNITY INOVA* or *UNITY plus* system with a wideline solids module. Not supplied with *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Syntax: `ssecho1`

Description: Sets up a parameter set for the quadrupole echo pulse sequence SSECHO1.

See also: *User Guide: Solid-State NMR*

ssfilter Full bandwidth of digital filter to yield a filtered FID (P)

Description: Specifies the full bandwidth of the digital filter applied to the original FID to yield a filtered FID for solvent subtraction. If `ssfilter` does not exist in the current experiment, enter `addpar('ss')` to add it. The command `addpar('ss')` creates additional time-domain solvent subtraction parameters `ssfilter`, `sslsfrq`, `ssntaps`, and `ssorder`.

Values: 'n', 10 to `sw`/2, in steps of 0.1 Hz. The default is 100 Hz.

If `ssfilter` is set to a value and `ssorder` is set to some value, the zfs (zero-frequency) option of solvent subtraction is selected.

If `ssfilter` is set to 'n', ("Not Used"), both the lfs (low-frequency suppression) and zfs options are turned off.

See also: *Getting Started*

Related:	<code>addpar</code>	Add selected parameters to the current experiment (M)
	<code>ft</code>	Fourier transform 1D data (C)
	<code>parfidss</code>	Create parameters for time-domain solvent subtraction (M)
	<code>ssntaps</code>	Number of coefficients in the digital filter (P)
	<code>sslsfrq</code>	Center of solvent-subtracted region of spectrum (P)
	<code>ssorder</code>	Order of polynomial to fit digitally filtered FID (P)
	<code>sw</code>	Spectral width in directly detected dimension (P)
	<code>wft</code>	Weight and Fourier transform 1D data (C)

sslsfrq Center of solvent-suppressed region of spectrum (P)

Description: Specifies the location of the center of the solvent-suppressed region of the spectrum. If `sslsfrq` does not exist in the current experiment, enter `addpar('ss')` to add it. `addpar('ss')` also creates time-domain solvent subtraction parameters `ssfilter`, `ssntaps`, and `ssorder`.

Values: 'n' (or 0) specifies solvent suppresses a region centered about the transmitter frequency. This is the default

Non-zero value shifts the solvent-suppressed region by `sslsfrq` Hz. Multiple regions may be suppressed by arraying the value of `sslsfrq`. Up to 4 values are allowed.

See also: *Getting Started*

Related:	<code>addpar</code>	Add selected parameters to the current experiment (M)
	<code>parfidss</code>	Create parameters for time-domain solvent subtraction (M)
	<code>ssfilter</code>	Full bandwidth of digital filter to yield a filtered FID (P)
	<code>ssntaps</code>	Number of coefficients in the digital filter (P)
	<code>ssorder</code>	Order of polynomial to fit digitally filtered FID (P)

ssntaps Number of coefficients in digital filter (P)

Description: Specifies the number of taps (coefficients) to be used in the digital filter for solvent subtraction. If `ssntaps` does not exist in the current experiment, enter `addpar('ss')` to add it. `addpar('ss')` also creates time-domain solvent subtraction parameters `ssfilter`, `sslsfrq`, and `ssorder`.

Values: Integer from 1 to `np`/4. The default is 121. An odd number is usually best.

The more taps in a filter, the flatter the passband response and the steeper the transition from passband to stopband, giving a more rectangular filter.

For the lfs (low-frequency suppression) option, the default is suitable.

For the zfs (zero-frequency suppression) option, a value between 3 and 21 usually works better.

See also: *Getting Started*

Related:	addpar	Add selected parameters to the current experiment (M)
	ft	Fourier transform 1D data (C)
	ni	Number of increments in 1st indirectly detected dimension (P)
	np	Number of points (P)
	parfidss	Create parameters for time-domain solvent subtraction (M)
	ssfilter	Full bandwidth of digital filter to yield a filtered FID (P)
	sslsfrq	Center of solvent-suppressed region of spectrum (P)
	ssorder	Order of polynomial to fit digitally filtered FID (P)
	wft	Weight and Fourier transform 1D data (C)

ssorder **Order of polynomial to fit digitally filtered FID (P)**

Description: Specifies the order of the polynomial to fit the digitally filtered FID if the zfs (zero-frequency suppression) option is selected for solvent subtraction. **ssorder** is not used if the lfs (low-frequency suppression) option is selected. If **ssorder** does not exist in the current experiment, enter **addpar** (' ss ') to add it. **addpar** (' ss ') also creates time-domain solvent subtraction parameters **ssfilter**, **sslsfrq**, and **ssntaps**.

The solvent subtraction option (zfs or lfs) is selected as follows:

- If **ssorder** and **ssfilter** are both set to values, zfs is selected.
- If **ssorder**= ' n ' and **ssfilter** is set to a value, lfs is selected.
- If **ssorder**= ' n ' and **ssfilter**= ' n ' , zfs and lfs are both turned off.

Values: ' n ' , integer from 1 to 20. The default is ' n ' .

See also: *Getting Started*

Related:	addpar	Add selected parameters to the current experiment (M)
	parfidss	Create parameters for time-domain solvent subtraction (M)
	ssfilter	Full bandwidth of digital filter to yield a filtered FID (P)
	sslsfrq	Center of solvent-suppressed region of spectrum (P)
	ssntaps	Number of coefficients in the digital filter (P)
	wft	Weight and Fourier transform 1D data (C)

ssplan **Set slice parameters for target slice (M)**

Applicability: Systems with imaging capabilities.

Syntax: **ssplan**

Description: Used by the Calculate Target button of the slice planner menu to calculate and set the slice parameters **pss**, **psi**, **phi**, and **theta**. **ssplan** creates the string parameter **planlock** and assigns it the value ' ssplan ' . This prevents a user inadvertently performing a second planning operation without applying the **reset** command to restore the original parameters for the scout data.

See also: *User Guide: Imaging*

Related:	drawslice	Display target slices (M)
	plan	Display menu for planning a target scan (M)
	phi	Euler angle phi from magnet frame (P)
	psi	Euler angle psi from magnet frame (P)
	pss	Slice position (P)
	theta	Euler angle theta from magnet frame (P)

sslist Conjugate gradient list (P)

Applicability: Systems with imaging capabilities.

Description: Sets an array of strings that defines the names of gradient parameters used for slice or voxel selection. If the pulse performs no slice selection operation, the user may enter ' ' or 'n' for the value of `sslist` (e.g., `sslist=' ' , 'gss' , 'gss'`). The `nD`, `seqcon`, `plist`, `patlist`, `pwrlist`, `fliplist`, and `sslist` parameters configure a particular parameter set for an application sequence defined by the value of the `seqfil` parameter. The `plist`, `patlist`, `pwrlist`, `fliplist`, and `sslist` parameters provide information concerning the rf pulse and conjugate gradients used by the sequence.

See also: *User Guide: Imaging*

Related:	<code>fliplist</code>	Standard flip angle list (P)
	<code>nD</code>	Application dimension (P)
	<code>patlist</code>	Active pulse template parameter list (P)
	<code>plist</code>	Active pulse length parameter list (P)
	<code>pwrlist</code>	Active pulse power level parameter list (P)
	<code>seqcon</code>	Acquisition loop control (P)
	<code>seqfil</code>	Application object code name (P)

ssprep Calculate slice gradient and slice selection parameters (M)

Applicability: Systems with echo planar imaging (EPI) capabilities.

Syntax: `ssprep`

Description: Calculates the slice gradient parameter, `gss`, and the slice selection parameters, `tpwr1` and `tpwr2`, for use in the EPI experiment. Unlike `imprep`, readout and phase encode related parameters are not modified by `ssprep`.

See also: *User Guide: Imaging*

Related:	<code>gss</code>	Slice selection gradient strength (P)
	<code>imprep</code>	Calculate gradient and rf parameters for imaging (M)
	<code>tpwr1</code>	Intensity of an excitation pulse (P)
	<code>tpwr2</code>	Intensity of an inversion pulse (P)

stack Fix stacking mode for processing and plotting arrayed spectra (M)

Syntax: `stack(mode)`

Description: When processing and plotting arrayed 1D spectra, VNMR automatically determines if the *stacking mode* is horizontal, vertical or diagonal from the number of traces and the number of lines in the spectrum. If you do not want this automatic function (or it makes an undesirable decision), you can override it by placing the `stack` macro in the experiment startup macro or by calling `stack` before processing (or reprocessing) a spectrum. The macro `autostack` switches back to automatic determination of the stack mode by destroying the parameter `stackmode`.

Arguments: `mode` is one of the stacking modes 'horizontal', 'vertical', or 'diagonal'.

See also: *Getting Started*

Related:	<code>autostack</code>	Automatic stacking for processing and plotting arrays (M)
	<code>procarray</code>	Process arrayed 1D spectra (M)
	<code>plarray</code>	Plot arrayed 1D spectra (M)
	<code>stackmode</code>	Stacking control for processing (P)

stackmode **Stacking control for processing arrayed 1D spectra (P)**

Description: Controls whether stacking for processing arrayed 1D spectra is automatic or nonautomatic. The *automatic stacking mode* can be overridden by creating and setting `stackmode` in the startup macro or before calling `procplot` or `procarray`. The `autostack` macro switches back to automatic determination of the stack mode by destroying this parameter.

Values: 'horizontal', 'vertical', or 'diagonal'.

See also: *Getting Started*

Related: `autostack` Automatic stacking for processing and plotting arrays (M)
`procarray` Process arrayed 1D spectra (M)
`procplot` Automatically process FIDs (M)
`stack` Fix stacking mode for processing and plotting arrayed spectra (M)

status **Display status of sample changer (C,U)**

Applicability: Systems with an automatic sample changer.

Syntax: (From VNMR) `status<(directory<,config_file>)>`
 (From UNIX) `status directory <config_file>`

Description: Displays a status window with a summary of all experiments and a scrollable list of individual experiments. Individual experiments are selected by clicking anywhere on the experiment of interest. `status` updates as the state of an automation run changes. If an experiment finishes or a new experiment is added, the `status` display is updated.

Arguments: `directory` is the path to the directory where the done queue (`doneQ`) is stored. In the UNIX shell, a directory path is required. In VNMR, a directory path is optional. The default is the automation mode directory.

`config_file` is the name of a user-supplied file that customizes status for local use. Refer to the manual *VNMR User Programming* for details.

Examples: (From VNMR) `status`
 (From VNMR) `status('/home/vnmr1/AutoRun_621')`
 (From UNIX) `status /home/vnmr1/AutoRun_621 mystatus`

See also: *User Guide: Liquids NMR; VNMR User Programming*

Related: `autodir` Automation directory absolute path (P)
`autoname` Prefix for automation data file (P)
`enter` Enter sample information for automation run (C,U)

stdshm **Interactively create a method string for autoshimming (M)**

Syntax: `stdshm`

Description: Creates a `method` string to be used in adjusting the spinning controls `z1`, `z2`, `z3`, and `z4` when a sample is changed. If non-spin controls also need adjusting, further shimming operations are required.

The `method` string is constructed in answer to questions about the sample length, the time available for shimming, and the solvent T_1 or, in FID shimming, the T_1 of the sample (background FID shimming is not available on *GEMINI 2000*). In asking about sample height, `stdshm` assumes that `z3` and `z4` need adjusting only with short samples; therefore, select “sample height will vary” if `z3` and `z4` shimming is definitely wanted.

Try lock shimming first to see if it produces a satisfactory result. Lock shimming requires a much shorter shimming time than FID shimming and usually adjusts `z1` and `z2` just as well. If lock shimming is unsatisfactory, try FID

shimming. Again, when z3 and z4 adjustment is required, lock shimming is faster, but FID shimming is more effective. `stdshm` displays the estimated shimming time, permitting revision when the time is too long.

To shim after running `stdshm`, enter `method= 'std'` (for lock shimming) or `method= 'fidstd'` (for FID shimming). Then enter `shim` or set the `wshim` parameter to shim before the start of acquisition.

Note that the command `newshm` is much like `stdshm` but that `newshm` provides more flexibility in making `method` strings

See also: *Getting Started*

Related:	<code>dshim</code>	Display a shim method string (M)
	<code>method</code>	Autoshim method (P)
	<code>newshm</code>	Interactively create a shim method with options (M)
	<code>shim</code>	Submit an Autoshim experiment to acquisition (C)
	<code>wshim</code>	Conditions when shimming is performed (P)

steam **Set up volume localized spectroscopy sequence (M)**

Applicability: Systems with imaging capabilities.

Syntax: `steam`

Description: Sets up a sequence for volume localized spectroscopy that uses the stimulated echo technique.

See also: *User Guide: Imaging*

sth **Minimum intensity threshold (P)**

Description: Intensity threshold above which transitions are printed and included in the simulated spectrum. Transitions whose intensity falls below this threshold are omitted from the simulation.

Values: 0 to 1.00. A typical value is 0.05.

See also: *User Guide: Liquids NMR*

Related:	<code>spins</code>	Perform spin simulation calculation (C)
	<code>spsm</code>	Enter spin system (M)
	<code>th</code>	Threshold (P)

string **Create a string variable (C)**

Syntax: `string(variable)`

Description: Creates a string variable without a value.

Arguments: `variable` is the string variable to be created.

Examples: `string('strvar1')`

See also: *VNMR User Programming*

strtext **Starting point for LP data extension in np dimension (P)**

Description: Specifies inclusively the complex time-domain data point at which LP (linear prediction) data extension (alteration) is to begin in the `np` dimension. Enter `addpar('lp')` to create `strtext` and other `np` dimension LP parameters in the current experiment.

Values: 1 to `np/2`

See also: *Getting Started*

Related: `addpar` Add selected parameters to the current experiment (M)
`dglp` Display group of linear prediction parameters (M)
`lpalg` LP algorithm in `np` dimension (P)
`np` Number of data points (P)
`strtlp` Starting point for LP calculation in `np` dimension (P)

strtext1 Starting point for LP data extension in `ni` dimension (P)

Description: Specifies inclusively the complex time-domain data point at which LP (linear prediction) data extension (alteration) is to begin in the `ni` dimension. Enter `addpar('lp',1)` to create `strtext1` and other `ni` dimension LP parameters in the current experiment.

Values: 1 to `ni/2`

See also: *User Guide: Liquids NMR*

Related: `addpar` Add selected parameters to the current experiment (M)
`dglp` Display group of linear prediction parameters (M)
`lpalg1` LP algorithm in `ni` dimension (P)
`ni` Number of increments in 1st indirectly detected dimension (P)
`strtlp1` Starting point for LP calculation in `ni` dimension (P)

strtext2 Starting point for LP data extension in `ni2` dimension (P)

Description: Specifies inclusively the complex time-domain data point at which LP (linear prediction) data extension (alteration) is to begin in the `ni2` dimension. Enter `addpar('lp',2)` to create `strtext2` and other `ni2` dimension LP parameters in the current experiment.

Values: 1 to `ni2/2`

See also: *User Guide: Liquids NMR*

Related: `addpar` Add selected parameters to the current experiment (M)
`dglp` Display group of linear prediction parameters (M)
`lpalg2` LP algorithm in `ni2` dimension (P)
`ni2` Number of increments in 2nd indirectly detected dimension (P)
`strtlp2` Starting point for LP calculation in `ni2` dimension (P)

strtlp Starting point for LP calculation in `np` dimension (P)

Description: Specifies the first complex, time-domain data point to be used in calculating the complex linear prediction (LP) coefficients in the `np` dimension. If `lpopt='b'`, the `strtlp`-th complex time-domain data point and the ensuing $(2 * \text{lpfilt} - 1)$ data points are used in this calculation. If `lpopt='f'`, the `strtlp`-th complex time-domain data point and the preceding $(2 * \text{lpfilt} - 1)$ data points are used in this calculation. Enter `addpar('lp')` to create `strtlp` and other `np` dimension LP parameters in the current experiment.

See also: *Getting Started*

Related: `addpar` Add selected parameters to the current experiment (M)
`dglp` Display group of linear prediction parameters (M)
`lpalg` LP algorithm in `np` dimension (P)
`lpfilt` LP coefficients to calculate in `np` dimension (P)
`lpnupts` LP number of data points in `np` dimension (P)

`lpopt` LP algorithm data extension in `np` dimension (P)
`strtext` Starting point for LP data extension in `np` dimension (P)

`strtlp1` Starting point for LP calculation in `ni` dimension (P)

Description: Specifies the first complex, time-domain data point to be used in calculating the complex linear prediction (LP) coefficients in the `ni` dimension. It functions analogously to `strlp`. Enter `addpar('lp',1)` to create `strtlp1` and other `ni` dimension LP parameters in the current experiment.

See also: *User Guide: Liquids NMR*

Related: `addpar` Add selected parameters to the current experiment (M)
`dglp` Display group of linear prediction parameters (M)
`lpalg1` LP algorithm in `ni` dimension (P)
`lpfilt1` LP coefficients to calculate in `ni` dimension (P)
`lpnupts1` LP number of data points in `ni` dimension (P)
`lpopt1` LP algorithm data extension in `ni` dimension (P)
`strtext1` Starting point for LP data extension in `ni` dimension (P)

`strtlp2` Starting point for LP calculation in `ni2` dimension (P)

Description: Specifies the first complex, time-domain data point to be used in calculating complex linear prediction (LP) coefficients in the `ni2` dimension. `strtlp2` functions analogously to `strlp`. Enter `addpar('lp',2)` to create `strtlp2` and other `ni2` dimension LP parameters in the current experiment.

See also: *User Guide: Liquids NMR*

Related: `addpar` Add selected parameters to the current experiment (M)
`dglp` Display group of linear prediction parameters (M)
`lpalg2` LP algorithm in `ni2` dimension (P)
`lpfilt2` LP coefficients to calculate in `ni2` dimension (P)
`lpnupts2` LP number of data points in `ni2` dimension (P)
`lpopt2` LP algorithm data extension in `ni2` dimension (P)
`strtext2` Starting point for LP data extension in `ni2` dimension (P)

`su` Submit a setup experiment to acquisition (M)

Syntax: `su`

Description: Sets up the system hardware to match the current parameters but does not initiate data acquisition. Typical uses of `su` are to change the system frequency in preparation for probe tuning, to change the sample temperature in advance of beginning an experiment (or after a variable temperature experiment is run), and to turn the decoupler on or off. If `load='y'`, `su` can be used to set shim values. `su` also sets lock parameters (`lockpower`, `lockgain`, `lockphase`) and the field offset parameter (`z0`).

`su` does *not* delete any existing data in the current experiment (only `go`, `ga`, and `au` do that). Everything that `su` does is also done by `go`, `ga`, and `au`.

On ^{UNITY}*INOVA* systems, shim DAC values are automatically loaded when the acquisition system boots up; if the acquisition system has been recently rebooted, `su` must be entered before `acqi` or `qtune` can be run.

See also: *Getting Started; User Guide: Liquids NMR*

Related: `acqi` Interactive acquisition display process (C)
`au` Submit experiment to acquisition and process data (C)
`change` Submit a change sample experiment to acquisition (M)

<code>ga</code>	Submit experiment to acquisition and FT the result (C)
<code>go</code>	Submit experiment to acquisition (C)
<code>load</code>	Load status of displayed shims (P)
<code>lock</code>	Submit an Autolock experiment to acquisition (C)
<code>lockgain</code>	Lock gain (P)
<code>lockphase</code>	Lock phase (P)
<code>lockpower</code>	Lock power (P)
<code>qtune</code>	Tune probe using swept-tune graphical tool (C)
<code>sample</code>	Submit change sample, autoshim experiment to acquisition (M)
<code>shim</code>	Submit an Autoshim experiment to acquisition (C)
<code>spin</code>	Submit a spin setup experiment to acquisition (C)
<code>z0</code>	Z0 field position (P)

sub **Subtract current FID from add/subtract experiment (C)**

Syntax: (1) `sub<(multiplier<, 'new'>)>`
 (2) `sub('new')`
 (3) `sub('trace', index)`

Description: Subtracts the last displayed or selected FID from the current contents of the add/subtract experiment (`exp5`). `lsfid` and `phfid` can be used to shift or phase rotate the selected FID before it is subtracted from the data in add/subtract experiment. A multi-FID add/subtract experiment can be created by using the 'new' keyword. Individual FIDs in a multi-FID add/subtract experiment can subsequently be subtracted by using the 'trace' keyword followed by the index number of the FID.

Arguments: `multiplier` is a value that the FID is to be multiplied by before being subtracted from the add/subtract experiment (`exp5`). The default is 1.0.
 'new' is a keyword to create a new FID element in an add/subtract experiment.
 'trace' is a keyword to use the next argument (`index`) as the number of the FID to subtract from in an add/subtract experiment. The default is to subtract from the first FID in a multi-FID add/subtract experiment.
`index` is the index number of the FID to be used as a target in a multi-FID add/subtract experiment.

Examples: `sub`
`sub(0.75)`
`sub('new')`
`sub('trace', 2)`

See also: *User Guide: Liquids NMR*

Related: `add` Add current FID to add/subtract experiment (C)
`clradd` Clear add/subtract experiment (C)
`lsfid` Number of complex points to left-shift n_1 interferogram (P)
`phfid` Zero-order phasing constant for n_p FID (P)
`select` Select a spectrum without displaying it (C)
`spsub` Subtract current spectra from add/subtract experiment (P)

substr **Select a substring from a string (C)**

Syntax: (1) `substr(string, word_number) : substring`
 (2) `substr(string, index, length) : substring`

Description: Returns a substring from a string based on the number of a word in the string (syntax 1) or on the starting character and length of the substring (syntax 2).

Arguments: `string` is the string or a string variable.

`word_number` is the number of the word to be selected. A *word* is defined here as any string of characters separated by spaces or tabs. For example, if `string` is 'There are 10 samples to run' and `word_number` is 4, the substring 'samples' is returned (see first example below).

`substring` returns the substring from `string`.

`index` is the character to start from, with the first character considered 1.

`length` is the length of substring in characters or spaces. For example, if `string` is 'abcdefg', `index` is 2, and `length` is 3, the substring 'bcd' is returned (see second example below)

Examples: `substr('There are 10 samples to run',4):sa`
`substr('abcdefg',2,3):sa`

See also: *VNMR User Programming*

Related: `length` Determine length of a string (C)
`string` Create a string variable (C)

suselfrq **Select peak, continue selective excitation experiment (M)**

Syntax: `suselfrq`

Description: Sets up selective frequency pulse, power, and shape and continue with the selective excitation experiment. Used by `NOESY1D`, and `TOCSY1D`.

Related: `NOESY1D` Change parameters for NOESY1D experiment (M)
`setselinv` Set up selective inversion (M)
`setselfrqc` Select selective frequency and width (M)
`TOCSY1D` Change parameters for TOCSY1D experiment (M)

svdat **Save data (C)**

Syntax: `svdat(file<,'f'|'m'|'i'|'b'>)`

Description: Outputs current data from the current experiment to a file. Integer data is scaled when it is written.

Note that `svdat` is also known and used as `svsdfd`; however, that name is in the process of being obsoleted.

Arguments: `file` is the name of the data file. The file is created in the current directory VNMR is in unless a full directory path is given. If a file of the same name already exists, the user will be queried to overwrite the file. If a fully qualified filename is not given, the file will be created in VNMR's current directory.

'f'|'m'|'i'|'b' defines how the data is to be written out: 'f' is 32-bit floating point, 'm' or 'i' is 16-bit integer scaled to 12 bits, and 'b' is 8-bit byte integer. The default is 'f'.

Floating point data is not scaled when written.

Integer data is scaled when written. A data value x is scaled as $ax+b$ where:

$$a = (vs*grays1*numgray)/64.0$$

$$b = numgray*(0.5-(grays1*grayctr/64.0))$$

where `numgray` (see below) has a default of 4096 for 'm' and 'i' formats and a default of 256 for the 'b' format, `grays1` has a default of 1, and `grayctr` has a default of 32.0.

To scale 16-bit integer data other than 12-bits, the global parameter `numgray` can be created using `create(numgray,real,global)` and set to the value 2^n , where n is the number of bits desired. For example, to scale to 15-bits, set `numgray=32768`.

The display parameters `graysl` and `grayctr` are used by the macros `svib` and `svsis` to save data files for ImageBrowser.

Examples: `svdat (rathead, 'b')`

See also: *User Guide: Imaging*

Related:	<code>browser</code>	Start ImageBrowser (U)
	<code>create</code>	Create new parameter in parameter tree (C)
	<code>fdfgluer</code>	Make FDF file from header and data parts (C)
	<code>grayctr</code>	Gray level window adjustment (P)
	<code>graysl</code>	Gray level slope (contrast) adjustment (P)
	<code>svib</code>	Generate and save images as ImageBrowser FDF files.(M)
	<code>svsis</code>	Generate and save images as FDF files (M)

svdef Copy .def files with FID (M)

Applicability: *GLIDE*

Syntax: `svdef (def_file, FID_file)`

Description: Duplicates .def file with the FID. Called by `AutoLIST`.

Arguments: `def_file` is either 'acquire', 'process', or 'plot'.
`FID_file` is the full name of the FID file.

Related: `AutoLIST` Run chained experiments (M)

svf Save FIDs in current experiment (M)

Syntax: `svf<(file<, 'nolog'><, 'arch'>)>`

Description: Saves parameters, text, and FID data in the current experiment to a file. No data is removed from the current experiment; `svf` merely saves a copy of the data in a different file. You can enter `rt` to retrieve the complete data set, or enter `rtp` to retrieve parameters only.

Arguments: `file` is the name of the file, with the suffix `.fid` added, to be created to save the data. The default is the system prompts for a file name. You are warned if you attempt to overwrite a file that already exists. In fact, if data has been acquired with the `file` parameter set, the data does not need to be saved. It is already stored in a named file.

'nolog' is a keyword to not save the log file with the data. The default is to save the log file.

'arch' is a keyword to assume that the data goes to a database and appends to the (or creates a) doneQ file with information that can be used by the command `status`.

Examples: `svf`
`svf ('/home/vnmr1/mydatafile')`

See also: *Getting Started*

Related:	<code>file</code>	File name (P)
	<code>rt</code>	Retrieve FID (M)
	<code>rtp</code>	Retrieve parameters (M)
	<code>status</code>	Display status of all experiments (C)

svfdf Save FID data in FDF format (M)

Syntax: `svfdf (directory)`

Description: Saves raw data from the FID file of the current experiment as an FDF (Flexible Data Format) file. Data is saved in multiple files, with one trace per file. The files are named `fid0001.fdf`, `fid0002.fdf`, etc. The `procp` file from the current experiment is also saved in the same directory.

The FDF file format is described in the manual *VNMR User Programming*. Note that the data is complex (FDF type="complex"), and the FDF ordinate = {"intensity", "intensity"}, indicating that each point consists of a pair of intensities. The FDF headers also contain the following special fields:

- `nfile` gives the sequential number of this file in the series.
- `ct` is the value of the VNMR `ct` parameter. The data should be divided by `ct` to give the average signal intensity for one scan.
- `scale` gives the power of two scaling factor for the data. The data should be multiplied by 2^{scale} to give the true values.

Arguments: `directory_name` is the directory in which to store the files. The extension `.dat` is appended to the given name.

Examples: `svfdf(curexp+' /raw')`

See also: *VNMR User Programming*

Related: `ct` Completed transients (P)
`svib` Save image data in FDF format (M)

svib Generate and save images as ImageBrowser FDF files (M)

Applicability: Systems with imaging capabilities.

Syntax: `svib(directory<, 'f' | 'm' | 'i' | 'o'>)`

Description: Generates images from the current experiment and saves them into the specified directory as FDF (Flexible Data Format) files. `svib` can save a single image, or a number of images in the case of multislice experiments.

The resulting FDF image files are composed of two parts: a text header, followed by the binary image data.

`svib` uses a the command `svdat` to dump the transformed data out to the data file. After dumping the headers out, a UNIX shell command `fdfgluer` is called to glue the headers to the data. `svdat` dumps the data so that the (0,0) coordinates are the first data point in the file.

Note that modifications to `svib` should be made in the user's `macplib` and that the output values of the direction cosines may not be correct.

Arguments: `directory` is the name of a directory that is made in the current working VNMR directory. The `.dat` extension is appended to the name. Image files are created in this directory as `image0001.fdf`, `image0002.fdf`, and so on. A `procp` file is also saved into this directory.

'f', 'm', 'i', and 'o' are keywords that define the type of image data:

- 'f' outputs the data in floating point format. This is the default.
- 'm' or 'i' outputs the data as 12-bit integer values in 16-bit words.
- 'b' outputs the data in 8-bit integer bytes.

Examples: `svib('rat.images')`

See also: *User Guide: Imaging*

Related: `dmi` Display multiple images (M)
`fdfgluer` Make FDF file from header and data parts (U)

`svdat` Save data (C)
`svimg` Generate and save images as FDF files (M)

svimg Generate and save images as FDF files (M)

Applicability: Systems with imaging capabilities. This command will be replaced by `svib` in future versions of VNMR and will be eventually obsolete.

Syntax: `svimg(directory<, 'f' | 'm'>)`

Description: Generates images from the current experiment and saves them into the specified directory as Flexible Data Format (FDF) files. `svimg` can save a single image, or a number of images in the case of multislice experiments.

`svimg` only saves images with the new imaging parameters that support oblique imaging. Unlike `svsis`, `svimg` does not need the name of the sequence. It formats the header according to the following parameters.

<code>seqcon</code>	Sequence loop control flag
<code>nD</code>	Data dimension assumed to be 2
<code>tn, dm</code>	Transmitter nucleus (string)
<code>sfrq, dfrq</code>	Spectrometer frequency (MHz)
<code>lro</code>	FOV size for read out axis (cm)
<code>lpe</code>	FOV size for phase encode axis (cm)
<code>pro</code>	Image center position on the read out axis (cm)
<code>ppe</code>	Image center position on 2D phase encode axis (cm)
<code>thk</code>	Slice thickness (mm)
<code>pss</code>	Slice position (cm)
<code>psi, phi, theta</code>	Euler angles determining direction

`svimg` uses the command `svsdfd` to dump the transformed data out to the data file. After dumping the headers out, the UNIX shell command `fdfgluer` is called to glue the headers to the data. `svsdfd` dumps the data in such a way that the (0,0) coordinates are the first data point in the file.

Note that modifications to the macro should be made in the user's `mac.lib`, and that the output values of the direction cosines may be incorrect.

Arguments: `directory` is the directory name desired. The specified directory is made in the user's data directory and is appended with the suffix `.dat`. Image files are created under this directory as `image0001.fdf`, `image0002.fdf`, etc. A `propar` file is also saved into this directory.

'f' | 'm' defines the type of image data. 'f' outputs the data in floating point format. 'm' outputs the data in 12-bit integer values in 16-bit words. The default is 'f'. ImageBrowser currently only accepts data in floating point values.

See also: *User Guide: Imaging*

Related:	<code>dfrq</code>	Transmitter frequency of first decoupler (P)
	<code>dm</code>	Decoupler mode for first decoupler (P)
	<code>fdfgluer</code>	Make FDF file from header and data parts (C)
	<code>lpe</code>	Field of view size for phase encode axis in cm (P)
	<code>lro</code>	Field of view size for readout axis in cm (P)
	<code>nD</code>	Application dimension (P)
	<code>phi</code>	Euler angle determining direction (P)
	<code>psi</code>	Euler angle determining direction (P)

<code>pss</code>	Slice position (P)
<code>ppe</code>	Position of image center on 2D phase encode axis (P)
<code>pro</code>	Position of image center on readout axis (P)
<code>seqcon</code>	Acquisition loop control (P)
<code>sfrq</code>	Transmitter frequency of observe nucleus (P)
<code>svsis</code>	Generate and save Varian images as FDF files (M)
<code>theta</code>	Euler angle determining direction (P)
<code>thk</code>	Slice thickness (P)
<code>tn</code>	Nucleus for observe transmitter (P)

svp Save parameters from current experiment (M)

Syntax: `svp(file)`

Description: Saves parameters from current experiment to a file. The parameter set can be retrieved with the `rtp` and `rt` macros. `svp` reflects any changes made in parameters up to the moment of entering `svp`, including acquisition parameters (unlike macro `svf`).

Arguments: `file` is the name of the file, with the suffix `.par` added, to be created to save the parameters. The default is the system prompts for a file name. You are warned if you attempt to overwrite a parameter set that already exists.

Examples: `svp('/vnmr/stdpar/P31')`
`svp('/usr/george/testdata')`

See also: *Getting Started*

Related: `rt` Retrieve FID (M)
`rtp` Retrieve parameters (M)
`svf` Save FIDs in current experiment (M)

svphf Save current VNMR phasefile (C)

Applicability: Systems with imaging capabilities.

Syntax: `svphf(file)`

Description: Copies current experiment phasefile (`curexp+ '/datdir/phasefile'`) to `planes` directory of current experiment (`curexp+ '/planes/file'`, where `file` is the file name given in the argument). The current phasefile is the current processed data set after apodization, Fourier transformation, vertical scaling, and phasing or absolute-value calculation, but before the contrast windowing controlled by the `grayctr` and `graysl` parameters. No parameters of any kind are stored with the phasefile. `svphf` creates the `planes` directory if it does not already exist.

Arguments: `file` is the name to be given to the phasefile when copied to the `planes` directory. Use only a relative path for `file`, not an absolute path.

Examples: `svphf('elsa')`

See also: *User Guide: Imaging*

Related: `curexp` Current experiment directory (P)
`grayctr` Gray level window adjustment (P)
`graysl` Gray level slope (contrast) adjustment (P)
`imcalc` Calculate 2D phasefiles (M,U)
`makephf` Transform and save images as phasefiles (M)
`rtphf` Return stored phasefile to the current VNMR phasefile (C)

svs Save shim coil settings (C)

Syntax: `svs(file)<:status>`

Description: Saves all shim coil settings except Z0 to a file. If `svs` cannot store the shim file, it displays the directories it tried to use.

Arguments: `file` is the name of a file for saving the shim coil settings. If the file name is an absolute path, `svs` uses it with no modifications. Otherwise, `svs` tries to go into up to three different directories, as follows:

- First, it looks for a `shims` subdirectory in your VNMR user directory. If that exists, the settings are stored there.
- Next, if the `shims` subdirectory does not exist, it then looks for the global parameter `shimspath`. If `shimspath` is present, it is expected to contain a directory name. If this directory exists and a new file entry can be created in the directory, the file is saved there.
- Finally, if this does not work, the file is saved in the `shims` subdirectory of the VNMR system directory.

`status` is a return variable with one of the following values after `svs` finishes:

- 0 indicates `svs` failed to store shim file.
- 1 indicates `svs` stored the shim file, either as an absolute path or in the `shims` subdirectory of the VNMR user directory.
- 2 indicates `svs` stored the file using the global parameter `shimspath`.
- 3 indicates `svs` stored the file in `shims` subdirectory of the VNMR system directory.

Examples: `svs('acetone')`
`svs('bb10mm'):r1`

See also: *Getting Started*

Related: `rts` Retrieve shim coil settings (C)
`shimspath` Path to user's `shims` directory (P)

svs Spin simulation vertical scale (P)

Description: Vertical scale for simulated spectrum.

Values: 0 to 1e10. A typical value is 200.

See also: *User Guide: Liquids NMR*

Related: `spins` Perform spin simulation calculation (C)
`spsm` Enter spin system (M)

svsis Generate and save images as FDF files (M)

Applicability: Systems with imaging capabilities.

Syntax: `svsis(directory<,'f'|'m'>)`

Description: Generates images from the current experiment and saves them into the specified directory as Flexible Data Format (FDF) files. `svsis` saves one image, or a number of images in the case of multislice experiments.

`svsis` only saves images from the standard SISCO imaging sequences: `image`, `shorte`, `stecho`, `multiecho`, `csi2D`, and `ssfp`. However, `svsis` can be easily modified to produce images from user sequences, provided the sequences use standard SISCO parameters, slice select pulse shapes, and generate data in the same manner as the standard SISCO sequences.

To modify `svsis` for a user sequence, add a line similar to the following in the “Valid Sequences” section:

```
$k=$k+1 $seqfil[$k]='t1image' $seq[$k]='ncsnn'
      $thk[$k]='image'
```

The new sequence name is `t1image`. Its reconstruction properties are given by `$seq`, whose values are similar to the parameter `seqcon`. The string characters for `seqcon` are defined as follows:

First character:	multiecho looping
Second character:	multislice looping
Third character:	2D phase encode loop
Fourth character:	3D phase encode loop
Fifth character:	4D phase encode loop

The values of each character are 'n' for a null loop, 's' for a standard loop, or 'c' for a compressed loop.

In this case, 'ncsnn' is a standard 2D image with compressed multislice. The `$thk` value is the slice thickness type, as defined by the type of acquisition, which in this case is the standard image sequence.

`svsis` uses the command `svsdfd` to dump the transformed data out to the data file. After dumping the headers out, the UNIX shell command `fdfgluer` is called to glue the headers to the data. `svsdfd` dumps the data in such a way that the (0,0) coordinates are the first data point in the file.

More detailed modifications can be made to `svsis` but it is left to the user to make these adjustments. Modifications to the macro should be made in the user's `maclib`.

Arguments: `directory` is the directory name desired. The specified directory is made in the user's data directory and is appended with the suffix `.dat`. Image files are created under this directory as `image0001.fdf`, `image0002.fdf`, etc. A `propar` file is also saved into this directory.

'f' | 'm' defines the type of image data. 'f' outputs the data in floating point format. 'm' outputs the data in 12-bit integer values in 16-bit words. The default is 'f'. ImageBrowser currently only accepts data in floating point values.

See also: *User Guide: Imaging*

Related: `seqcon` Acquisition loop control (P)
`svimg` Generate and save images as FDF files (M)

svtmp Move experiment data into experiment subfile (M)

Syntax: `svtmp<(file)>`

Description: Moves the experiment data (parameters, FID, and transformed spectrum) from current experiment into a subdirectory inside `curexp+ '/subexp'`. Unlike the macro `cptmp`, the experiment data is no longer accessible in the current experiment; only a copy of the parameters is still present.

Arguments: `file` is the name of the subfile that receives the experiment data. The default name is either the transmitter nucleus (if `seqfil='s2pul'`) or the pulse sequence name.

Examples: `svtmp`
`svtmp('cosy')`

See also: *Getting Started*

Related: `cptmp` Copy experiment data into experiment subfile (M)
`curexp` Current experiment directory (P)
`rttmp` Retrieve experiment data from experiment subfile (M)
`seqfil` Pulse sequence name (P)

sw Spectral width in directly detected dimension (P)

Description: Sets the total width of the spectrum to be acquired, from one end to the other. All spectra are acquired using quadrature detection. The spectral width determines the sampling rate for data, which occurs at a rate of $2 * sw$ points per second (actually sw pairs of complex points per second). Note that the sampling rate itself is not entered, either directly or as its inverse (known on some systems as the *dwelt time*).

The sampling rate is internally constrained to a multiple of a timebase that is set based on the type of acquisition controller board in the system (see the description of the `acquire` statement for a description of these boards):

- 12.5 ns on systems with a Data Acquisition Controller board.
- 25 ns on systems with a Pulse Sequence Controller board or an Acquisition Controller board.
- 0.1 μ s on a *GEMINI 2000* system and on systems with an Output board.

If a value of `sw` is entered whose inverse is not an even multiple of the time base listed above, `sw` is automatically adjusted to a slightly different value to give an acceptable sampling rate.

A value of `sw` greater than the value of the `maxsw_loband` parameter forces `dp= 'y'`.

To enter a value in ppm, append the character `p` (e.g., `sw=200p`).

If a DSP facility is present in the system (i.e., `dsp= 'i'` or `dsp= 'r'`) and oversampling in the experiment has not been turned off by setting `oversamp= 'n'`, then the oversampling factor will be recalculated.

Values: Number, in Hz. The range possible is based on the system:

On *UNITYINOVA*: 100 Hz to 500 kHz.

On *MERCURY-VX*, *MERCURY*, *GEMINI 2000* broadband, *UNITYplus*, *UNITY*, and *VXR-S*: 100 Hz to 100 kHz.

On *GEMINI 2000* $^1\text{H}/^{13}\text{C}$: 100 Hz to 23 kHz.

On *UNITYINOVA* and *UNITYplus* with solids: up to 5 MHz.

On *UNITY* and *VXR-S* with solids: up to 2 MHz.

On *UNITYplus*, *UNITY*, *VXR-S* with 200-kHz option: 100 Hz to 200 kHz.

See also: *Getting Started*

Related: `dp` Double precision (P)
`dsp` Type of DSP for data acquisition (P)
`maxsw_loband` Maximum spectral width of input board (P)
`oversamp` Oversampling factor for acquisition (P)
`sw1` Spectral width in 1st indirectly detected dimension (P)
`sw2` Spectral width in 2nd indirectly detected dimension (P)
`sw3` Spectral width in 3rd indirectly detected dimension (P)

sw1 Spectral width in 1st indirectly detected dimension (P)

Description: Analogous to the **sw** parameter except that **sw1** applies to the first indirectly detected dimension of a multidimensional data set. The increment of the variable evolution time **d2** is automatically calculated from **sw1**. The number of increments for this dimension is set by **ni**. To create **sw1** in the current experiment, as well as **ni** and **phase**, enter **addpar** (' 2d ').

See also: *User Guide: Liquids NMR*

Related: **addpar** Add selected parameters to the current experiment (M)
d2 Incremented delay in 1st indirectly detected dimension (P)
ni Number of increments in 1st indirectly detected dimension (P)
phase Phase selection (P)
sw Spectral width in directly detected dimension (P)
sw2 Spectral width in 2nd indirectly detected dimension (P)
sw3 Spectral width in 3rd indirectly detected dimension (P)

sw2 Spectral width in 2nd indirectly detected dimension (P)

Description: Analogous to the **sw** parameter except that **sw2** applies to the second indirectly detected dimension of a multidimensional data set. The increment of the variable evolution time **d3** is automatically calculated from **sw2**. The number of increments for this dimension is set by **ni2**. To create **sw2** in the current experiment, as well as **d3**, **ni2**, and **phase2**, enter **addpar** (' 3d ').

See also: *User Guide: Liquids NMR*

Related: **addpar** Add selected parameters to the current experiment (M)
d3 Incremented delay for 2nd indirectly detected dimension (P)
ni2 Number of increments in 2nd indirectly detected dimension (P)
phase2 Phase selection for 3D acquisition (P)
sw Spectral width in directly detected dimension (P)
sw1 Spectral width in 2nd indirectly detected dimension (P)
sw3 Spectral width in 3rd indirectly detected dimension (P)

sw3 Spectral width in 3rd indirectly detected dimension (P)

Description: Analogous to the **sw** parameter except that **sw3** applies to the third indirectly detected dimension of a multidimensional data set. The increment of the variable evolution time **d4** is automatically calculated from **sw3**. The number of increments for this dimension is set by **ni3**. To create **sw3** in the current experiment, as well as **d4**, **ni3**, and **phase3**, enter **addpar** (' 4d ').

See also: *User Guide: Liquids NMR*

Related: **addpar** Add selected parameters to the current experiment (M)
d4 Incremented delay for 3rd indirectly detected dimension (P)
ni3 Number of increments in 3rd indirectly detected dimension (P)
par4d Create 4D acquisition parameters (C)
phase3 Phase selection for 4D acquisition (P)
sw Spectral width in directly detected dimension (P)
sw1 Spectral width in 1st indirectly detected dimension (P)
sw2 Spectral width in 2nd indirectly detected dimension (P)

syn Number of frequency synthesizers (obsolete)

Description: This parameter is no longer part of VNMR.

sysgcoil **System gradient coil (P)**

Description: Specially reserved string parameter that specifies which physical gradient set is currently installed, and allows convenient updating of important gradient characteristics when one gradient set is interchanged for another. The value to `sysgcoil` is assigned to the parameter `gcoil` when joining experiments or retrieving parameter sets.

This parameter is set in the CONFIG window (opened by entering `config`) to the name of the gradient set in use. Once set, it is then available to all experiments and to all users.

See also: *VNMR and Solaris Software Installation; User Guide: Imaging*

Related:	<code>boresize</code>	Magnet bore size (P)
	<code>config</code>	Display current configuration and possibly change it (M)
	<code>createtable</code>	Generate new gradient calibration file (M)
	<code>gcoil</code>	Current gradient coil (P)
	<code>gmax</code>	Maximum gradient strength (P)
	<code>setgcoil</code>	Assign sysgcoil configuration parameter (M)
	<code>trise</code>	Gradient rise time (P)

system **System type (P)**

Description: A global parameter that sets the basic type of system: spectrometer or data station. The value is set using the System Type label in the CONFIG window (opened from `config`).

Values: 'spectrometer' is a spectrometer system (Spectrometer choice in CONFIG window).

'datastation' is a system used as a data station (Data Station choice in CONFIG window). Acquisition is not allowed in this setting.

See also: *VNMR and Solaris Software Installation*

Related:	<code>config</code>	Display current configuration and possibly change it (M)
	<code>Console</code>	System console type (P)

systemdir **VNMR system directory (P)**

Description: Contains path to VNMR system directory, typically `/vnmr`. The UNIX environmental variable `vnmrssystem` initializes `systemdir` at bootup.

See also: *Getting Started*

t1 T_1 exponential analysis (M)Syntax: `t1`

Description: Processes data obtained using an array of values of the parameter `d2` for a T_1 experiment. It runs `expfit`, which does an exponential curve fitting that determines the value of T_1 . The output is matched to the equation:

$$M(t) = (M(0) - M_0) * \exp(-t/T_1) + M_0$$

where M_0 is the equilibrium Z magnetization and $M(0)$ is the magnetization at time zero (e.g., immediately after the 180° pulse for an inversion recovery T_1 experiment). Notice that this equation will fit inversion recovery data (for which $M(0)$ is approximately equal to $-M_0$) or saturation recovery data (for which $M(0)$ is 0).

The required input is the file `fp.out` from `fp` and the values of the arrayed parameter. The T_1 analysis is done for all the peaks listed in `fp.out`. Peaks are selected for analysis by entering `fp(index1, index2, ...)` before running the analysis. The output file is the `analyze.list` in the current experiment. The file `analyze.out` is used by `expl` to display the results. The output of the analysis program shows T_1 and its standard deviation, but does not explicitly show $M(0)$, M_0 , or their standard deviations. The $M(0)$ and M_0 values can be found in “raw” form in `analyze.out` in the current experiment, but their standard deviations are not part of the program output.

See also: *User Guide: Liquids NMR*

Related:	<code>d2</code>	Incremented delay in 1st indirectly detected dimension (P)
	<code>expfit</code>	Make least squares fit to polynomial or exponential curve (C)
	<code>fp</code>	Find peak heights (C)
	<code>t1s</code>	T_1 exponential analysis with short output table (M)
	<code>t2</code>	T_2 exponential analysis (M)
	<code>t2s</code>	T_2 exponential analysis with short output fable (M)

t1image Fit arrayed imaging data to T_1 exponential data (M)

Applicability: Systems with imaging capabilities.

Syntax: `t1image`

Description: Does preprocessing required for fitting arrayed imaging data to T_1 data using the `imfit` program. The user is prompted for the base phasefile names and the lower limit noise threshold. `t1image` then transforms and saves all of the images, and calls `imfit` to complete the fitting process.

See also: *User Guide: Imaging*

Related:	<code>imfit</code>	Fit arrayed imaging data to T_1 or T_2 exponential data (M,U)
	<code>t2image</code>	Fit arrayed imaging data to T_2 exponential data (M)

t1s T_1 exponential analysis with short output table (M)Syntax: `t1s`

Description: Performs the same analysis as `t1` but produces a short output table showing only a summary of the measured relaxation times.

See also: *User Guide: Liquids NMR*

Related: **t1** T_1 exponential analysis (M)

t2 T_2 exponential analysis (M)

Syntax: **t2**

Description: Processes data obtained using an array of values for the base time parameter **bt** for a T_2 experiment. It runs **expfit**, which does an exponential curve fitting that determines the value of T_2 . The output is matched to the equation:

$$M(t) = (M(0) - M(\text{inf})) * \exp(-t/T_2) + M(\text{inf})$$

where $M(0)$ is the magnetization at time zero (i.e., the full magnetization excited by the observe pulse) and $M(\text{inf})$ is the xy-magnetization at infinite time (zero unless the peak is sitting on an offset baseline).

The required input is the file **fp.out** from **fp** and the values of the arrayed parameter. The T_2 analysis is done for all the peaks listed in **fp.out**. Peaks are selected for analysis by entering **fp(index1, index2, ...)** before running the analysis. The output file is the file **analyze.list** in the current experiment. The file **analyze.out** is used by **exp1** to display the results. The output of the analysis program shows T_2 and its standard deviation, but does not explicitly show $M(0)$, $M(\text{inf})$, or their standard deviations. The $M(0)$ and $M(\text{inf})$ values can be found in “raw” form in **analyze.out** in the current experiment, but their standard deviations are not part of the program output.

See also: *User Guide: Liquids NMR*

Related: **expfit** Make least squares fit to polynomial or exponential curve (C)
fp Find peak heights (C)
t1 T_1 exponential analysis (M)
t1s T_1 exponential analysis with short output table (M)
t2s T_2 exponential analysis with short output fable (M)

t2image Fit arrayed imaging data to T_2 exponential data (M)

Applicability: Systems with imaging capabilities.

Syntax: **t1image**

Description: Does preprocessing required for fitting arrayed imaging data to T_2 data using the **imfit** program. The user is prompted for the base phasefile names and the lower limit noise threshold. **t2image** then transforms and saves all of the images, and calls **imfit** to complete the fitting process.

See also: *User Guide: Imaging*

Related: **imfit** Fit arrayed imaging data to T_1 or T_2 exponential data (M,U)
t1image Fit arrayed imaging data to T_1 exponential data (M)

t2s T_2 exponential analysis with short output table (M)

Syntax: **t2s**

Description: Performs the same analysis as **t2** but produces a short output table showing only a summary of the measured relaxation times.

See also: *User Guide: Liquids NMR*

Related: **t2** T_2 exponential analysis (M)

tabc **Convert data in table order to linear order (M)**

Syntax: `tabc <(dimension) >`

Description: Converts arbitrarily ordered data obtained under control of an external AP table to linear monotonic order, suitable for processing in VNMR. The data must have been acquired according to a table in the `tablib` directory.

Imaging and other 2D experiments are normally acquired so that the order of the incremented acquisition parameter, such as the phase-encode gradient, is linear and monotonic. For a standard imaging experiment, this linear order means that the phase-encode gradient progresses from a starting negative value monotonically up through zero to a positive value (e.g., -64, -63, -62, ..., -1, 0, 1, ..., 62, 63). The `ft2d` program assumes this structure in its operation.

Data from table-driven 2D pulse sequences is used by entering `tabc` *only once* before normal 2D processing and/or parameter storage. In this situation, `tabc` takes no arguments and is executed by entering `tabc` in the VNMR command window. A simple check is done by `tabc` to prevent it from being executed more than once on the same data set.

2D data is expected to be in the standard VNMR format, but if the 2D data is in the compressed format, setting `dimension` to 1 converts the data. `tabc` supports all 2D data types recognized by VNMR: arrayed, compressed multislice, and arrayed compressed multislice,

3D data is expected to be in the compressed/standard format, in which there are `ni` standard 2D planes of data (the third dimension), each consisting of `nf` compressed FIDs (the second dimension). Setting `dimension` to 3 reorders 3D data acquired with an external table.

`tabc` reads the file `fid` in the `acqfil` subdirectory of the current experiment. Before the data is reordered, this file is written to the file `fid.orig` in the same `acqfil` directory. If for any reason `tabc` fails or results in an unpredictable or undesired transformation, the original raw data can be recovered by moving `fid.orig` back to `fid`. To gain more disk space, you can delete `fid.orig` after you are satisfied that conversion is successful.

Use `tabc` on saved data that has been loaded into a VNMR experiment or on data in an experiment that has just been acquired but not yet saved. In the first case, converted data must be resaved for the saved data set to reflect conversion.

`tabc` requires that data must have the same number of “traces” as the table elements. It does not support any of the advanced features of table expansion (e.g., the entire table must be explicitly listed in the table file), and expects to find only one table in a file; whether the table is `t1` or `t60` is unimportant.

Arguments: `dimension` specifies the type of data to be converted: 1 for 2D compressed data, 2 for 2D standard data, or 3 for 3D compressed/standard data. The default is 2.

Examples: `tabc`
`tabc(1)`
`tabc(3)`

See also: *User Guide: Imaging*

Related:	<code>flashc</code>	Convert compressed 2D data to standard 2D format (C)
	<code>ft2d</code>	Fourier transform 2D data (C)
	<code>ni</code>	Number of increments in 1st indirectly detected dimension (P)
	<code>nf</code>	Number of FIDs (P)

tan Find tangent value of an angle (C)

Syntax: `tan (angle) <:n>`

Description: Finds the tangent of an angle.

Arguments: `angle` is an angle, in radians.

`n` is the return value giving the tangent of `angle`. The default is to display the tangent value in the status window.

Examples: `tan (.5)`
`tan (val) :tan_val`

See also: *VNMR User Programming*

Related:	<code>arccos</code>	Calculate arc cosine of real number (M)
	<code>arcsin</code>	Calculate arc sine of real number (M)
	<code>arctan</code>	Calculate arc tangent of real number (M)
	<code>atan</code>	Find arc tangent value of a number (C)
	<code>cos</code>	Find cosine value of an angle (C)
	<code>exp</code>	Find exponential value of a number (C)
	<code>ln</code>	Find natural logarithm of a number (C)
	<code>sin</code>	Find sine value of an angle (C)

tape Read tapes from VXR-style system (M,U)

Syntax: (From VNMR) `tape (<-d device,><type,>option
 <,file1,file2,...>)`
 (From UNIX) `tape <-d device> <type> <option>
 <file1> <file2>...`

Description: Displays the contents of a VXR-style (Gemini, VXR-4000, or XL) 9-track tape for use with VNMR or reads one or several files from the tape into the current directory. Note that the *write* option is not supported (i.e., VNMR only *reads* tapes in a VXR-style format and does not write to a tape).

Arguments: `device` is the tape drive device name. The default value is `/dev/rst8`. For Solaris 2.x, `device` should be `/dev/rmt/0b`. For AIX systems, `device` should be `/dev/rmt0`. If the default value is not set properly or another device name is wanted, be sure to type `-d` and a space before the device name you want to input.

`type` is the type of tape to be accessed. `'-q'` or `'-s'` select the 1/4-inch tape unit ("streaming" or cartridge tape); this is the default. `'-9'`, `'-h'`, or `'-n'` select the 1/2-inch tape unit (open reel tape drive).

`option` is one of the following:

- `'help'` is a keyword to display help on the use of the system.
- `'cat'` is a keyword to display a catalog of files on tape.
- `'read'` is a keyword to read one or more files. This option requires that the files be listed as the next argument.
- `'rewind'` is a keyword to rewind tape (1/2-inch tape only).
- `'quit'` is a keyword to release the tape drive (1/2-inch tape only).

`file1`, `file2`, ... are the names of one or more files to be read.

Wildcard characters (`*` and `?`) can be used.

Examples: `tape ('cat')`
`tape ('-h', 'read', 'mydata')`
`tape -h read mydata`
`tape -d /dev/rmt/01b read mydata`

See also: *Getting Started*

Related: `decomp` Decompose a VXR-style directory (C)
`vxr_unix` Convert VXR-style text files to UNIX format (M,U)

tape Control tape options of files program (P)

Description: Defines device that `files` program accesses when it is instructed to read or write to a tape. The parameter `tape` is in the user's global parameter tree.

Values: Name of a device. The default device is `/dev/rst8`. If `tape` does not exist or is set to the null string (two single quotes with no space between), `files` uses its default device value. Notice that different computers define tape drives differently. For VnmrSGI, `tape= '/dev/tapens'` is appropriate. For Solaris, `tape= '/dev/rmt/0mb'`.

See also: *Getting Started*

Related: `files` Interactively handle files (C)

tbox Draw a tilted box (C)

Applicability: Systems with imaging capabilities.

Syntax: (1) `tbox(<'keywords'>angle, xcenter, ycenter, hlen, vlen)`
 (2) `tbox(<'keywords'>angle, xcenter, ycenter, hlen, vlen, vspace, nboxes)`

Description: Draws a tilted box centered at `xcenter, ycenter` (as indicated by a small diamond) (syntax 1) or produces an aligned array of `nboxes` tilted boxes centered at `xcenter, ycenter` (syntax 2) and separated by `vspace`.

Arguments: `'keywords'` identifies the output device (`'graphics' | 'plotter'`), drawing mode (`'xor' | 'normal'`), and drawing capability (`'newovly' | 'ovly' | 'ovlyC'`).

- `'graphics' | 'plotter'` is a keyword selecting the output device. The default is `'plotter'`. The output selected is passed to subsequent `pen`, `move`, or `draw` commands and remains active until a different mode is specified.
- `'xor', 'normal'` is a keyword for the drawing mode when using the `'graphics'` output device. The default is `'normal'`. In the `'xor'` mode, if a line is drawn such that one or more points of the line are in common with a previous `'xor'` line, the common points are erased. In the `normal` mode, the common points remain. The mode selected is passed to subsequent `pen`, `move`, and `draw` commands and remains active until a different mode is specified.
- `'newovly', 'ovly'` and `'ovlyC'` are keywords that specify an interactive drawing capability that is slightly slower than the `'xor'` mode but more consistent in color. `'newovly'` clears any previous draws, boxes, and writes made with the `'ovly'` modes and draws the figure. `'ovly'` draws without clearing so that multi-segment figures can be created. `'ovlyC'` clears without drawing.

`angle` is the tilt angle, in radians, of a box.

`xcenter, ycenter` are coordinates on the screen, in mm, specifying the point at which a box is centered.

`hlen` is the horizontal coordinate on the screen, in mm.

`vlen` is the vertical coordinate, on the screen, in mm.

`vspace` controls the separation or overlap of boxes.

`nboxes` is the number of boxes.

Examples: `tbox('plotter', 20, 100, 40, 150)`

See also: *Getting Started*

Related: [box](#) Draw a box on a plotter or graphics display (C)

tccapply Apply table conversion reformatting to data (C)

Applicability: Systems with imaging capabilities.

Syntax: `tccapply<(file)>`

Description: Rearranges the spectra in a 2D data set that resides in the current data file. You must apply [ft1d](#) to the data before you can use `tccapply`. Using values from an AP table, `tccapply` arranges the spectra corresponding to the value in the AP table from low value to high value. The values might have already been read in by the [tcopen](#) command.

Arguments: `file` specifies the name of the file containing the AP table to be read. The file must be in `$vnmruser/tablib`.

Examples: `tccapply('petable')`

See also: *User Guide: Imaging*

Related: [tabc](#) Close table conversion file (C)
[ft1d](#) Fourier transform along f_2 dimension (C)
[ft2d](#) Fourier transform along f_2 dimension (C)
[tcclose](#) Close table conversion file (C)
[tcopen](#) Open table convert file (C)

tcclose Close table conversion file (C)

Applicability: Systems with imaging capabilities.

Syntax: `tcclose`

Description: Removes a table conversion file and frees the memory used to store the sorted table indices read in with the [tcopen](#) command.

See also: *User Guide: Imaging*

Related: [tccapply](#) Apply table conversion reformatting to data (C)
[tcopen](#) Open table convert file (C)

tcl Send Tcl script to Tcl version of dg window (C)

Syntax: `tcl(script)`

Description: Sends a Tcl (Tool Command Language) script to the Tcl version of the [dg](#) window. If this window is not active, this command does nothing.

Arguments: `script` is any legal Tcl script.

See also: *VNMR User Programming*

Related: [dg](#) Display group of acquisition/processing parameters (C)

tcopen Open table conversion file (C)

Applicability: Systems with imaging capabilities.

Syntax: `tcopen<(file)>`

T

Description: Explicitly reads, sorts, and stores in memory, a table conversion file. `tcopen` uses the file when `tcapply` is called.

Arguments: `file` specifies the file to be read; it must be in `$vnmruser/tablib`.

Examples: `tcopen('petable')`

See also: *User Guide: Imaging*

Related: `tcapply` Apply table conversion reformatting to data (C)
`tcclose` Close table convert file (C)

te Echo time (P)

Applicability: Systems with imaging capabilities.

Description: Echo time for imaging and some localized spectroscopy experiments.

In gradient and spin echo imaging sequences, `te` is usually defined as the time measured from the middle of the initial rf excitation pulse to the center of the resulting echo.

In multiecho sequences, `te` may also define the time duration between successive echoes, normally a constant interval. Multiecho sequences with variable echo times are also possible, in which case the `te` period between successive echoes may take on a range of values represented by a `te` array.

Some more unusual pulse sequences, such as stimulated echo, RARE and Fast Spin Echo, may use `te` in ways somewhat different from the normal standards.

See also: *User Guide: Imaging*

Related: `ne` Number of echoes to be acquired (P)

techron Set up parameters for gradient amplifier tests (M)

Applicability: Systems with imaging capabilities.

Syntax: `techron`

Description: Recalls parameters sets for gradient amplifier tests during microimaging installation.

See also: *Microimaging Module Installation*

temp Open the Temperature Control window (C)

Applicability: Systems with a variable temperature (VT) controller.

Syntax: `temp`

Description: Opens the Temperature Control window, which has the following capabilities:

- Turn temperature control off.
- Set temperature control on at a specified temperature in degrees C.
- Enable temperature control from within an experiment using the `temp` parameter and the `su`, `go`, `ga`, or `au` macros. This mode is the default.
- Alternatively, turn off experiment control of the temperature and allow only the Temperature Control window (and `sethw`) to set the temperature. This mode has the advantage that, often times, `temp` is different between experiments. Joining a different experiment and entering `go` can unexpectedly change the temperature. This mode prevents this problem.
- Resetting the temperature controller when the temperature cable is reconnected to a probe.

See also: *Getting Started; User Guide: Liquids NMR*

Related:	<code>acqi</code>	Interactive acquisition display process (C)
	<code>au</code>	Submit experiment to acquisition and process data (M)
	<code>ga</code>	Submit experiment to acquisition and FT the result (M)
	<code>go</code>	Submit experiment to acquisition (M)
	<code>sethw</code>	Set values for hardware in acquisition system (C)
	<code>su</code>	Submit a setup experiment to acquisition (M)
	<code>temp</code>	Sample temperature (P)
	<code>tin</code>	Temperature interlock (P)

`temp` **Sample temperature (P)**

Applicability: Systems with a variable temperature (VT) module.

Description: Sets the temperature of sample.

Values: 'n' or -150 to +200, in steps of 0.1°C. 'n' instructs the acquisition system not to change the VT controller and to ignore temperature regulation throughout the course of the experiment.

See also: *Getting Started; User Guide: Liquids NMR*

Related:	<code>temp</code>	Open the Temperature Control window (C)
	<code>tempcal</code>	Temperature calculation (C)
	<code>tin</code>	Temperature interlock (P)
	<code>vtc</code>	Variable temperature cutoff point (P)

`tempcal` **Temperature calculation (C)**

Applicability: Systems with a variable temperature (VT) module.

Syntax: `tempcal(solvent)<:temperature>`

Description: For exact determination of sample temperature when using the VT unit, a temperature calibration curve must be made for each probe used. All data, such as gas flow, must be noted. Use samples of ethylene glycol for high-temperature calibration, and use samples of methanol for low-temperature calibration. To make the calculation:

- Bring the sample to the desired temperature and allow sufficient time for equilibration, then obtain a spectrum.
- Next, align two cursors on the two resonances in the spectrum, then enter `tempcal('e')` for ethylene glycol, or enter `tempcal('m')` for methanol. The temperature is calculated based on the difference frequency between the cursors.

Arguments: `solvent` is the sample solvent: 'glycol', 'e', or 'g' for ethylene glycol, or 'methanol' or 'm' for methanol.

`temperature` returns the calculated value of the sample temperature. The default is the system displays the value.

Examples: `tempcal('glycol')`
`tempcal('m'):temp`

See also: *User Guide: Liquids NMR*

`tep` **Post-acquisition delay in EPI experiments (P)**

Applicability: Systems with echo planar imaging (EPI) capabilities.

Description: Delay used in the EPI sequence to adjust the beginning of data acquisition. This correction is necessary to allow for the finite (propagation) delay of gradient pulses. This allows the user to center the EPI echoes in the acquisition window.

Values: Number, in μs . Typically 0 to 50 μs , depending on the gradient hardware.

See also: *User Guide: Imaging*

Related: `episet` Set up parameters for EPI experiment (M)

`testct` Check ct for resuming signal-to-noise testing (M)

Syntax: `testct`

Description: Used by the `testsn` macro to decide when to resume testing of signal-to-noise. See the description of `testsn` for details.

See also: *User Guide: Liquids NMR*

Related: `ct` Completed transients (P)
`testsn` Test signal-to-noise of a spectrum (M)

`testsn` Test signal-to-noise of a spectrum (M)

Syntax: `testsn`

Description: Part of the automatic periodic signal-to-noise testing that occurs during various automated acquisitions, most notably `c13`. Transforms the data using `fn=16000`, and then baseline corrects, setting the left-most 10% of the spectrum and the right-most 2% as baseline. After the baseline correction, `testsn` uses `getsn` to calculate the signal-to-noise.

- If signal-to-noise exceeds the desired goal in parameter `sn` (found in the standard carbon parameter set `/vnmr/stdpar/c13`), `testsn` aborts the experiment using the command `halt`, which initiates processing according to the `wexp` parameter.
- If signal-to-noise is not reached, `testsn` estimates the signal-to-noise ratio at the end of the experiment. If signal-to-noise target will not be reached by then, it cancels subsequent signal-to-noise testing, but allows the experiment to proceed.
- If the signal-to-noise target will be reached before the end of the experiment, it saves the estimated number of transients required to reach the goal in the parameter `r7` (using a conservative estimate), and then sets the processing at future blocks to be only `testct`, which simply tests if `ct` is greater than `r7`, and, if so, resumes testing of signal-to-noise with `testsn`.

See also: *User Guide: Liquids NMR*

Related: `c13` Automated carbon acquisition (M)
`fn` Fourier number in directly detected dimension (P)
`getsn` Get signal-to-noise estimate of a spectrum (M)
`halt` Abort acquisition with no error (C)
`r1-r7` Real parameter storage for macros (P)
`sn` Signal-to-noise ratio (P)
`testct` Check ct for resuming signal-to-noise testing (M)
`wexp` Specify action when experiment completes (C)

`text` Display text or set new text for current experiment (C)

Syntax: `text<(text_string)><:string_variable>`

Description: Associated with each experiment is a text file, consisting of a block of text, that can be used to describe the sample and experiment. `text` allows displaying the text file and changing the text file for the current experiment. A UNIX text editor, such as `vi`, or the macro `textvi` can also be used to edit the text file of the current experiment.

Arguments: `text_string` is a string of text that replaces the existing text file. The default is to display the text file in the current experiment. The characters `\` or `\n` can be used in the string to denote a new line, and the characters `\t` can be used to denote a tab (see example below).

`string_variable` returns the text in `text_string` as a string variable. Thus, for example, the `text:n1` and `text(n1+'cosy experiment')` commands, where `n1` is a string, can be used in a macro to add a “cosy experiment” to the text. An equivalent operation using the `atext` command would be `atext('cosy experiment')`.

Examples: `text('Sample 101\tCDC13\\13 February')`

See also: *Getting Started*

Related:

<code>atext</code>	Append string to the current experiment text (M)
<code>ctext</code>	Clear the text of the current experiment (C)
<code>curexp</code>	Current experiment directory (P)
<code>dtext</code>	Display a text file in the graphics window (C)
<code>puttxt</code>	Put text file into another file (C)
<code>textvi</code>	Edit text file of current experiment (M)
<code>vnmrprint</code>	Print text files (U)

textis **Return the current text display status (C)**

Syntax: (1) `textis(command):$yes_no`
 (2) `textis:$display_command`

Description: Determines if a command given by the user currently controls the text window (syntax 1) or returns the name of the command currently controlling the text window (syntax 2).

Arguments: `command` is the name of a command that potentially may be controlling the text window.

`$yes_no` returns 1 if `command` controls the text window, or 0 if it does not.

`$display_command` returns the name of the command currently controlling the text window.

Examples: `textis:$display`
`if ($display = 'dg') then . . . endif`

See also: *VNMR User Programming*

Related: `graphis` Return the current graphics display status (C)

textvi **Edit text file of current experiment (M)**

Syntax: `textvi`

Description: Edits the text file of the current experiment using the UNIX text editor `vi`. `textvi` is equivalent to the command `vi(curexp+' /text')`.

See also: *Getting Started*

Related:

<code>edit</code>	Edit a file with user-selectable editor (M)
<code>text</code>	Display text or set new text for current experiment (C)
<code>vi</code>	Edit text file with <code>vi</code> editor (M)

- th** **Threshold (P)**
- Description: Sets threshold for printout of peak frequencies so that peaks greater than `th` on the plot appear on any peak listings. `th` is always bipolar (i.e., negative peaks greater in magnitude than `th` also appear in peak listings).
- Values: 0 to 1e9, in mm.
- See also: *User Guide: Liquids NMR*
- Related: `thadj` Adjust threshold for peak printout (M)
-
- th2d** **Threshold for integrating peaks in 2D spectra (P)**
- Description: Used by `l12d` when determining the bounds of a peak and calculating its volume. To create the 2D peak picking parameters `th2d` and `xdiag` in the current experiment, enter `addpar('l12d')`.
- Values: From 0.0 to 1.0. If `th2d=1.0`, `l12d` integrates all points in the peak that are above the current threshold for the spectrum (i.e., the portion of the peak that can be seen in a contour plot of the spectrum). A smaller value causes `l12d` to integrate a larger area when determining the volume of a peak. If `th2d=0.5`, for example, `l12d` integrates all points in a peak that are above 0.5 times the current threshold.
- See also: *User Guide: Liquids NMR*
- Related: `addpar` Add selected parameters to the current experiment (M)
`l12d` Automatic and interactive 2D peak picking (C)
`xdiag` Threshold for excluding diagonal peaks when peak picking (P)
-
- thadj** **Adjust threshold for peak printout (M)**
- Syntax: `thadj<(max_peaks<,noise_mult<,llarg1<,llarg2>>>>>`
- Description: Adjusts the threshold `th` so that no more than a specified maximum number of peaks are found in a subsequent line listing (see `nll`) and so that `th` is at least a specified noise multiplier times the root-mean-square noise level.
- Arguments: `max_peaks` is the maximum number of peaks in the displayed spectral range. The default is `wc/4` (i.e., the threshold is adjusted such that `ppf` will produce a “reasonable” number of lines with any width of plot).
- `noise_mult` is a noise multiplier used to calculate the minimum value for `th` from the size of the root-mean-square noise.
- `llarg1` is the `noise_mult` argument (the default is 3) to the `nll` command used inside this macro
- `llarg2` is the keyword argument ('pos', 'neg', 'all'; the default is 'all'.) to the `nll` command used inside this macro.
- Examples: `thadj`
`thadj(50)`
`thadj(200,4)`
`thadj(200,4,2)`
`thadj(200,4,2,'pos')`
- See also: *Getting Started*
- Related: `nll` Find line frequencies and intensities (C)
`ppf` Plot teak frequencies over spectrum (M)
`th` Threshold (P)
`vsadj` Automatic vertical scale adjustment (M)
`vsadj2` Automatic vertical scale adjustment by powers of two (M)

<code>vsadjc</code>	Automatic vertical scale adjustment for ^{13}C spectra (M)
<code>vsadjh</code>	Automatic vertical scale adjustment for ^1H spectra (M)
<code>wc</code>	Width of chart (P)

theta Euler angle theta from magnet frame (P)

Applicability: Systems with imaging capabilities.

Description: Euler angle theta from magnet frame.

Values: -90 to $+90$, in degrees.

See also: *User Guide: Imaging*

Related:	<code>phi</code>	Euler angle phi from magnet frame (P)
	<code>psi</code>	Euler angle psi from magnet frame (P)

thk Slice thickness (P)

Applicability: Systems with imaging capabilities.

Description: Returns the slice thickness, in mm.

See also: *User Guide: Imaging*

ti Inversion recovery time (P)

Applicability: Systems with imaging capabilities.

Description: Specifies the recovery time following an inversion prepulse in inversion recovery experiments. The value of `ti` generally has a strong impact on image contrast, which depends on the T_1 relaxation time of the sample in different regions of the image.

See also: *User Guide: Imaging*

Related:	<code>ir</code>	Inversion recovery mode (P)
	<code>pi</code>	Width of an inversion pulse (P)
	<code>pipat</code>	Shape of an inversion pulse (P)
	<code>tpwri</code>	Intensity of inversion pulse (P)

ticks Number of trigger pulses (P)

Applicability: Systems with imaging capabilities.

Description: Sets the number of trigger pulses the system waits before acquisition begins. This parameter is found in some Varian pulse sequences that feature gating. `ticks` controls an external gating signal received through an external TTL input. If `ticks=0`, the system ignores trigger pulses and runs in the nontriggered mode. The pre- and post-trigger delays `rcvry` and `hold` remain active in the nontriggered mode.

Values: Integers from 0 to 100.

See also: *User Guide: Imaging*

Related:	<code>hold</code>	Post-trigger delay (P)
	<code>rcvry</code>	Pre-trigger delay (P)

time Display experiment time or recalculate number of transients (M)

Syntax: `time(<<hours>, >minutes>)`

Description: Estimates the acquisition time or recalculates the number of transients so that the total acquisition time is approximately the requested time. The parameters

T

looked at when calculating the time per transient are `d1`, `d2`, `d3`, `at`, `ni`, `sw1`, `ni2`, and `sw2`.

Arguments: `hours` and `minutes` are numbers making up a time to be used by the system to recalculate the parameter `nt` so that the total acquisition time is approximately the time requested; the default (no arguments) is for the system to estimate the acquisition time for a 1D, 2D, or 3D experiment using the parameters in the current experiment.

Examples: `time`
`time(2,45)`

Alternate: Show Time button in the Acquire menu.

See also: *Getting Started*

Related:

<code>at</code>	Acquisition time (P)
<code>d1</code>	First delay (P)
<code>d2</code>	Incremented delay in 1st indirectly detected dimension (P)
<code>d3</code>	Incremented delay in 2nd indirectly detected dimension (P)
<code>exptime</code>	Display experiment time (C)
<code>ni</code>	Number of increments in 1st indirectly detected dimension (P)
<code>ni2</code>	Number of increments in 2nd indirectly detected dimension (P)
<code>nt</code>	Number of transients (P)
<code>sw1</code>	Spectral width in 1st indirectly detected dimension (P)
<code>sw2</code>	Spectral width in 2nd indirectly detected dimension (P)

tin **Temperature interlock (P)**

Description: Controls error handling based on temperature regulation. If temperature regulation is lost, `tin` can be used to select whether an error is generated and acquisition is halted or whether a warning is generated and acquisition continues. In both cases, the lost regulation will cause `werr` processing to occur, thus providing a user-selectable mechanism to respond to VT failure.

Values: 'n' turns off the temperature interlock feature

'w' indicates the variable temperature regulation light is monitored during the course of the experiment and, if it starts to flash (regulation lost), a warning is generated; however, acquisition is not stopped.

'y' indicates the variable temperature regulation light is monitored during the course of the experiment and, if it starts to flash (regulation lost), the current data acquisition is stopped. The acquisition will not resume automatically if regulation is regained.

See also: *User Guide: Liquids NMR*

Related:

<code>in</code>	Lock and spin interlock (P)
<code>werr</code>	When error (P)

title **Plot a title on a plotter (M)**

Applicability: Systems with imaging capabilities.

Syntax: `title(string)`

Description: Plots a string provided by the user on the plotter.

Arguments: `string` is a string of characters.

Examples: `title('15 June Image')`

See also: *User Guide: Imaging*

- t1t** **First-order baseline correction (P)**
- Description: When spectral display is active, the command `dc` turns on a linear drift correction (baseline correction). The result of this operation includes calculating a first-order baseline correction parameter `t1t`. The calculation is made by averaging of a small number of points at either end of the display and drawing a straight line baseline between them.
- See also: *Getting Started*
- Related: `cdc` Cancel drift correction (C)
`dc` Calculate spectral drift correction (C)
`lv1` Zero-order baseline correction (P)
- tmove** **Left-shift FID to time-domain cursor (M)**
- Syntax: `tmove`
- Description: Provides an alternative method of left shifting time-domain data. To use this method, position the right time cursor at the place that should be the start of the FID, then enter `tmove`. This adjusts `lsfid` to left-shift the FID.
- See also: *Getting Started*
- Related: `lsfid` Number of complex points to left-shift *np* FID (P)
- tmsref** **Reference 1D proton or carbon spectrum to TMS (M)**
- Syntax: `tmsref:tms_found`
- Description: Tries to locate a TMS line. If found, `tmsref` re-references the spectrum to the TMS line and returns a 1 to the calling macro; if not found, `tmsref` returns 0 and the referencing is left as it was. In the case of other signals (e.g., from silicon grease) immediately to the left of the TMS line (even if they are higher than the reference line), `tmsref` tries avoiding those by taking the rightmost line in that area, as long as it is at least 10% of the main Si-CH₃ signal. Large signals within 0.6 ppm for ¹H (or 6 ppm for ¹³C) to the right of TMS may lead to misreferencing.
- Arguments: `tms_found` returns 1 if a TMS line was located or returns 0 if not.
- See also: *Getting Started*
- Related: `c13` Automated carbon acquisition (M)
`h1` Automated proton acquisition (M)
- tn** **Nucleus for observe transmitter (P)**
- Description: Changing the value of `tn` causes a macro (`_tn`) to be executed that extracts values for `sfrq` and `tof` from lookup tables. The tables, stored in the directory `/vnmr/nuctables`, are coded by atomic weights.
- Values: In the lookup tables, typically given by 'H1', 'C13', 'P31', etc. The value `tn='lk'` sets the deuterium frequency, and also holds the lock current and switches the relay in the automated deuterium gradient shimming module, if present, so that deuterium signal may be observed without disturbing lock. The frequency is the same as `tn='H2'`. The relay is available only on UNITY^{INNOVA}, MERCURY-VX, MERCURY, and UNITY^{plus} systems.
- Alternate: Nucleus Selection Menu
- See also: *Getting Started*
- Related: `dn` Nucleus for first decoupler (P)
`dn2` Nucleus for second decoupler (P)

<code>dn3</code>	Nucleus for third decoupler (P)
<code>sfrq</code>	Transmitter frequency of observe nucleus (P)
<code>tof</code>	Frequency offset for observe transmitter (P)

- tncosyps** **Set up parameters for TNCOSYPS pulse sequence (M)**
- Applicability: Sequence is not supplied with *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.
- Syntax: `tncosyps`
- Description: Sets up a homonuclear correlation experiment (phase-sensitive version) with water suppression.
- See also: *User Guide: Liquids NMR*
- tndqcosy** **Set up parameters for TNDQCOSY pulse sequence (M)**
- Applicability: Systems with a linear amplifier on the observe channel and a T/R switch. Sequence is not supplied with *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.
- Syntax: `tndqcosy`
- Description: Sets up a 2D J-correlation experiment with water suppression.
- See also: *User Guide: Liquids NMR*
- tnmqcosy** **Set up parameters for TNMQCOSY pulse sequence (M)**
- Applicability: Systems with hardware digital phaseshifter for transmitting with direct-synthesis rf; otherwise, software small-angle phaseshifter for transmitting with the old-style rf is used. Sequence not supplied with *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.
- Syntax: `tnmqcosy`
- Description: Sets up a multiple-quantum filtered COSY experiment with water suppression.
- See also: *User Guide: Liquids NMR*
- tnnoesy** **Set up parameters for TNNOESY pulse sequence (M)**
- Applicability: Systems with a linear amplifier on the observe channel and a T/R switch. Sequence is not supplied with *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.
- Syntax: `tnnoesy`
- Description: Sets up a 2D cross-relaxation experiment with water suppression.
- See also: *User Guide: Liquids NMR*
- tnroesy** **Set up parameters for TNROESY pulse sequence (M)**
- Applicability: Sequence is not supplied with *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.
- Syntax: `tnroesy`
- Description: Sets up a rotating-frame NOE experiment with water suppression.
- See also: *User Guide: Liquids NMR*
- tntocsy** **Set up parameters for TNTOCOSY pulse sequence (M)**
- Applicability: Systems with T/R switch, computer-controlled attenuators, and linear amplifiers on observe channel. Sequence not supplied with *MERCURY-VX*, *MERCURY*, and *GEMINI 2000*.

Syntax: `tntocsy`

Description: Sets up a total-correlation spectroscopy experiment (HOHAHA) with water suppression.

See also: *User Guide: Liquids NMR*

tnuc Retrieve nucleus table parameters for transmitter (obsolete)

Description: A command no longer in VNMR. Use `setfrq` as the effective replacement.

Related: `setfrq` Set frequencies of rf channels in system (C)

TOCSY Change parameters for TOCSY experiment (M)

Syntax: `TOCSY<('GLIDE')>`

Description: Converts the current parameter set to a TOCSY experiment.

Arguments: 'GLIDE' is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding proton spectrum for the experiment.

Related: `tocsy` Set up parameters for TOCSY experiment (M)

tocsy Set up parameters for TOCSY pulse sequence (M)

Applicability: Any system with linear amplifiers on the observe channel. Sequence is not supplied with the *GEMINI 2000*.

Syntax: `tocsy`

Description: Sets up a total-correlation (TOCSY) experiment, also known as the Homonuclear Hartmann-Hahn (HOHAHA) experiment.

Alternate: TOCSY button on the 2D Pulse Sequence Setup Secondary Menu.

See also: *User Guide: Liquids NMR*

Related: `ft1dac` Combined arrayed 2D FID matrices (M)
`ft2dac` Combined arrayed 2D FID matrices (M)
`wft1dac` Combined arrayed 2D FID matrices (M)
`wft2dac` Combined arrayed 2D FID matrices (M)

TOCSY1D Change parameters for TOCSY1D experiment (M)

Syntax: `TOCSY<('GLIDE')>`

Description: Converts the current parameter set to a TOCSY1D (also known as DPGSE-noe) experiment. A 1D proton spectrum is displayed with `ds_sel frq` menu to do peak selection.

Arguments: 'GLIDE' is a keyword used only in a *GLIDE* run to ensure that the starting parameter set is the corresponding carbon spectrum for the experiment.

Related: `NOESY1D` Change parameters for NOESY1D experiment (M)

tof Frequency offset for observe transmitter (P)

Description: Controls the exact positioning of the transmitter. As the value assigned to `tof` increases, the transmitter moves to a higher frequency (toward the left side of the spectrum). The minimum step size of `tof` is determined by the type of rf hardware in the spectrometer. The limit is specified using the Step Size label in the CONFIG window (opened from `config`, implicitly set for *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems). Systems with broadband style rf

(`rftype='b'`) generally have 100-Hz resolution; all other systems have 0.1 Hz resolution.

Values: Approximate, depends on frequency. On *GEMINI 2000* $^1\text{H}/^{13}\text{C}$ systems: -50000 to 50000, in Hz (^1H has 0.0795... Hz step size, ^{13}C has 19.... Hz step size); on other systems: -100000 to 100000, in Hz.

See also: *Getting Started*

Related: `config` Determine current configuration and possibly change it (M)
`dof` Frequency offset for first decoupler (P)
`dof2` Frequency offset for second decoupler (P)
`dof3` Frequency offset for third decoupler (P)
`rftype` Type of rf generation (P)

tpe Duration of the phase encoding gradient pulse (P)

Applicability: Systems with imaging capabilities.

Description: Sets the length of the phase encoding gradient period in imaging and CSI experiments. The spectral width in the indirect dimension (`sw1`) is determined from `tpe` as `sw1=1/tpe`. `tpe` may be recomputed within the pulse sequence to provide optimum performance, such as minimum echo time, or scaled to match the required timing for slice refocusing and readout dephasing.

See also: *User Guide: Imaging*

Related: `gpe` Phase encoding gradient increment in DAC units (P)
`nv` Number of 2D phase encode steps to be acquired (P)
`sw1` Spectral width in 1st indirectly detected dimension (P)
`tpe2, tpe3` Duration of second and third phase encoding gradient periods (P)

tpe2, tpe3 Duration of second and third phase encoding gradient periods (P)

Applicability: Systems with imaging capabilities.

Description: Sets the lengths of the phase encoding gradient periods that control second spatial and third spatial dimensions in nD imaging and CSI experiments. For example, 3D volume imaging sequence have two independent phase encode axes, controlled by `tpe` and `tpe2`. It is common to have a single phase encoding time block, in which two independent phase encode gradients share the same time period. In this case, `tpe` and `tpe2` would be equal.

See also: *User Guide: Imaging*

Related: `sw2` Spectral width in 2nd indirectly detected dimension (P)
`tpe` Duration of the phase encoding gradient pulse (P)

tpwr Observe transmitter power level with linear amplifiers (P)

Applicability: Systems with a linear amplifier on the observe channel.

Description: Controls transmitter power. The value of the attenuator upper safety limit is set using the Upper Limit label in the CONFIG window (opened from `config`). Depending on hardware adjustments, the system may saturate at a given value of `tpwr` (i.e., values above a certain value may give equal output).

Values: On *MERCURY* systems, the range is 0 to 63, in dB, 1-dB steps.
 On *GEMINI 2000* systems: 0 to 63.5 (63.5 is maximum power), in units of dB, 0.5-dB steps.

On systems other than *GEMINI 2000* with 63-dB attenuator installed: 0 to 63 (63 is maximum power), in units of dB. About 55 to 60 is normal. Lower values (e.g., 49) might be used for water suppression experiments like 1-3-3-1.

On systems other than *GEMINI 2000* with 79-dB attenuator installed: -16 to 63 (63 is maximum power), in units of dB.

CAUTION: Continuous power greater than 2 watts in a switchable probe will damage the probe. Always carefully calibrate power to avoid exceeding 2 watts. The maximum value for `tpwr` on a 200-MHz, 300-MHz, or 400-MHz system with a linear amplifier on the decoupler channel has been set to 49, corresponding to about 2 watts of power. Before using `tpwr=49` for continuous decoupling, ensure safe operation by measuring the output power. This should be done during system installation and checked periodically by the user.

See also: *Getting Started*

Related:	<code>cattn</code>	Coarse attenuator (P)
	<code>config</code>	Determine current configuration and possibly change it (M)
	<code>dpwr</code>	Power level for first decoupler with linear amplifiers (P)
	<code>dpwr2</code>	Power level for second decoupler (P)
	<code>dpwr3</code>	Power level for third decoupler (P)
	<code>dpwrf</code>	First decoupler fine power (P)
	<code>fattn</code>	Fine attenuator (P)
	<code>tpwrf</code>	Observe transmitter fine power (P)

`tpwr1` Intensity of an excitation pulse (P)

Applicability: Systems with imaging capabilities.

Description: Specifies the peak power, in dB, of transmitter pulses corresponding to `p1`.

See also: *User Guide: Imaging*

Related:	<code>p1</code>	First pulse width (P)
	<code>tpwr</code>	Observe transmitter power level with linear amplifiers (P)

`tpwr2` Intensity of an excitation pulse (P)

Applicability: Systems with imaging capabilities.

Description: Specifies the peak power, in dB, of transmitter pulses corresponding to `p2`.

See also: *User Guide: Imaging*

Related:	<code>p2</code>	Second pulse width (P)
	<code>tpwr</code>	Observe transmitter power level with linear amplifiers (P)

`tpwrcal` Calibrate power levels of 90° and 180° pulse (M)

Applicability: Systems with imaging capabilities.

Syntax: `tpwrcal(start_tpwr, end_tpwr)`

Description: Sets up paired arrays of form `tpwr1`, `tpwr2`. The parameter `array` is set as `array= '(tpwr1, tpwr2)'`. This macro is especially useful for calibrating the 90° and 180° power levels for a slice.

Arguments: `start_tpwr` is the starting value for the `tpwr` part of the arrayed pairs. The starting value for `tpwr1` is 6 less than `start_tpwr`.

`end_tpwr` is the ending value for the `tpwr` part of the arrayed pairs. The ending value for `tpwr1` is 6 less than `end_tpwr`.

Examples: `tpwrcal(30, 45)`

See also: *User Guide: Imaging*

Related: `array` Parameter order and precedence (P)
`tpwr` Observe transmitter power level with linear amplifiers (P)
`tpwr1` Intensity of excitation pulse (P)

tpwrf Observe transmitter fine power (P)

Applicability: Systems with a fine attenuator on the observe transmitter channel.

Description: Controls the transmitter fine attenuator. Systems with this attenuator are designated using the Fine Attenuator label in the CONFIG window (opened from `config`). The fine attenuator is linear and spans 60 dB (^{UNITY}INOVA or UNITYplus system) or 6 dB (other systems). If `tpwrf` is not present, enter `create('tpwrf','integer') setlimit('tpwrf',4095,0,1)` to create it.

Values: 0 to 4095, where 4095 is maximum power. If `tpwrf` does not exist in the parameter table, a value of 4095 is assumed.

See also: *Getting Started*

Related: `config` Determine current configuration and possibly change it (M)
`dpwr` Power level for first decoupler with linear amplifiers (P)
`dpwrf` First decoupler fine power (P)
`fattn` Fine attenuator (P)
`tpwr` Observe transmitter power level with linear amplifier (P)

tpwri Intensity of inversion pulse (P)

Applicability: Systems with imaging capabilities.

Description: Specifies the peak power of transmitter pulses corresponding to `pi`.

Values: Number, in dB.

See also: *User Guide: Imaging*

Related: `ir` Inversion recovery mode (P)
`pi` Width of an inversion pulse in microseconds (P)
`tpwr` Observe transmitter power level with linear amplifiers (P)
`tpwr1` Intensity of an excitation pulse (P)

tpwrm Observe transmitter linear modulator power (P)

Applicability: ^{UNITY}INOVA and UNITYplus systems.

Description: Controls the power level on the observe transmitter linear modulator.

Values: 0 to 4095, where 4095 is maximum power. If `tpwrm` does not exist in the parameter table, a value of 4095 is assumed.

See also: *Getting Started*

Related: `config` Determine current configuration and possibly change it (M)
`dpwrf` First decoupler fine power (P)
`fattn` Fine attenuator (P)

tr Repetition time in imaging and localized spectroscopy experiments (P)

Applicability: Systems with imaging capabilities.

Description: Sets the repetition time of an experiment. The definition of repetition time can vary somewhat from pulse sequence to pulse sequence. In general, for imaging experiments, `tr` is the time required to complete one transient of one phase

encode step, including relaxation delay, excitation, data acquisition, and any post-acquire events, such as rf spoiling, phase encode rewinding, and gradient turn-off.

For multislice and/or multiecho imaging sequences, `tr` includes the complete multislice/multiecho train (for standard arrayed slice acquisitions, where the second character in `seqcon` is `s`, the complete train is not included, and `tr` is the repetition time for each slice position).

Some 1D experiments, such as STEAM and ISIS are also written using `tr`, with the similar definition that `tr` is the repetition time per transient.

`tr` describes the total duration of all events in a pulse sequence, and will never be directly found as an argument to “delay.” Instead, `tr` will generally be used in precalculations to determine the time required to pad the sum of programmed events up to the desired repetition time. This padding delay will often be found in the pulse sequence as “predelay.”

See also: *User Guide: Imaging*

Related: `seqcon` Acquisition loop control (P)

trace Mode for *n*-dimensional data display (P)

Applicability: All systems; however, *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems can only process 3D data and cannot acquire such data.

Description: Sets the multidimensional data display mode.

Values: 'f1' displays the f_1 axis horizontally and allows f_1 traces to be displayed.
 'f2' displays the f_2 axis horizontally and allows f_2 traces to be displayed.
 'f3' displays the f_3 axis horizontally and allows f_3 traces to be displayed if the data set is 3D.

See also: *User Guide: Liquids NMR*

transfer Move parameters to target experiment (M)

Applicability: Systems with imaging capabilities.

Syntax: `transfer(data_type, <scout_exp,>target_exp)`

Description: Transfers selectively parameter data from a scout data set to the target experiment in preparation for the next or future scanning operation. The following series of actions are carried out: (1) `transfer` joins the scout experiment and saves the current parameters in the `userdir+ '/parlib'` directory, under the file name `TRANSFER.par`. Any previous parameter sets with this file name are removed. (2) `transfer` then joins the target experiment and displays the transfer menu. The user may then use the menu to selectively copy groups of parameters from `TRANSFER.par` to the target experiment. The groups that may be transferred include:

Nucleus	<code>tn, resto</code>
Voxel	<code>pos1-pos3, vox1-vox3, psi1, theta1, mopos, scpos</code>
Slice	<code>pss, psi, phi, theta, mopos, scpos</code>
FOV	<code>lro, lpe</code>
Coil	<code>rfcoil, gcoil</code>
Sample	<code>mopos, scpos</code>

If any of the parameters `pos1, pos2, pos3, psi1, theta1, psi, phi`, or `theta` are arrayed in the scout experiment, in addition to copying the voxel or

slice list, `transfer` sets the `array` parameter in the target experiment. Other parameters copied by `transfer` cannot legally be arrayed, except `pss`.

Parameters `tn`, `gcoil`, and `pss` are special cases that trigger `_macros` execution. `transfer` executes the `_tn`, `_gcoil`, and `_pss` (setloop) programs once if these parameters are copied to the target. This execution ensures that all the normal side effects of setting these parameters are properly executed.

Arguments: `data_type` is a keyword defining the type of data for transfer as 'slice' or 'voxel', which can be abbreviated to 's' or 'v', respectively.

`scout_exp` is the number of the scout experiment. The default is the current experiment is the source of the scout parameter data.

`target_exp` is the number of the target experiment.

Examples: `transfer('s',5)`
`transfer('v',5,6)`

See also: *User Guide: Imaging*

Related:	<code>gcoil</code>	Read data from gradient calibration tables (P)
	<code>lpe</code>	Field of view size for phase encode axis (P)
	<code>lro</code>	Field of view size for readout axis (P)
	<code>phi</code>	Euler angle from magnet frame (P)
	<code>psi</code>	Euler angle from magnet frame (P)
	<code>pss</code>	Slice position (P)
	<code>resto</code>	NMR resonance offset frequency (P)
	<code>rfcoil</code>	RF pulse calibration identity (P)
	<code>theta</code>	Euler angle from magnet frame (P)
	<code>tn</code>	Nucleus for observe transmitter (P)
	<code>userdir</code>	VNMR user directory (P)

traymax **Sample changer tray slots (P)**

Applicability: Systems with an automatic sample changer.

Description: Specifies the type of sample changer. It also can be used to disable the sample changer. The value is set using the Sample Changer label in the CONFIG window (opened from `config`).

Values: 0 is setting for no sample changer present or, if a sample changer is attached, to disable the changer (None choice in the CONFIG window).

9, 50, 100, 96, 48 are `traymax` values that indicate the number of sample slots for the corresponding sample changer (9 is for Carousel, 50 is for SMS/ASM 50 Sample, 100 is for SMS/ASM 100 Sample, 96 is for VAST, and 48 is for NMS).

See also: *VNMR and Solaris Software Installation; Getting Started*

Related: `config` Display current configuration and possibly change it (M)

trfunc **Translate screen coordinates (M)**

Applicability: Systems with imaging capabilities.

Syntax: `trfunc($x,$y):$xincm,$yincm`

Description: Translates screen coordinates to hertz or centimeters depending upon the `axis` parameter.

Arguments: `$x` is a coordinate . . .

`$y` is a coordinate . . .

`$xincm` is a coordinate . . .

\$yincm is a coordinate . . .

See also: *User Guide: Imaging*

Related: **axis** Axis label for displays and plots (P)
trfuncd Translate screen distance (M)

trfuncd Translate screen distance (M)

Applicability: Systems with imaging capabilities.

Syntax: `trfuncd($screenlength):$imagelength`

Description: Translates a screen distance into centimeters in a real image. It is only useful in `axis='cc'` (aspect ratio constrained) images.

Arguments: `$screenlength` is the length of the display screen.
`$imagelength` is the length of the image.

See also: *User Guide: Imaging*

Related: **axis** Axis label for displays and plots (P)
trfunc Translate screen coordinates (M)

trise Gradient rise time (P)

Applicability: Systems with imaging capabilities.

Description: Stores the time required for an x, y, or z magnetic field gradient to change from zero to maximum gradient (**gmax**). Because the gradient system is adjusted by Varian at installation time so that all three gradients have the same rise time, only one parameter is used to describe the rise time for all three gradients.

This parameter accurately describes the time required for gradient changes only in systems that use slew-rate-limited gradient amplifiers, such as the Oxford GPS 2239 gradient amplifier supplied with most imaging systems. Do not confuse this gradient rise time with the amount of time required by a pulse sequence to transmit the DAC value that initiates a gradient value change (see the `gradient` and `vgradient` statements in the manual *VNMR User Programming* for a discussion of that timing).

`trise` is used in some sequences to control various aspects of gradient timing, including the automatic setup of gradient refocusing. This parameter does not need to be declared and initialized in pulse sequence source code files, because it is a standard PSG parameter and is therefore already declared and initialized by the Varian-supplied PSG library. See the source file `sems.c` for an example.

`trise` is defined in the system gradient table files found in the directory `$vnmrsystem/gradtables`, and is automatically set from one of those files when a value is entered for the parameter **gcoil**.

Values: 0.005 seconds (nominal).

See also: *User Guide: Imaging*

Related: **boresize** Magnet bore size (P)
gcoil Read data from gradient calibration tables (P)
gmax Maximum gradient strength (P)

troesy Set up parameters for TROESY pulse sequence (M)

Applicability: Not on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems.

Syntax: `troesy`

Description: Sets up parameters for the transverse cross-relaxation experiment in a rotating frame.

See also: *User Guide: Liquids NMR*

tshift **Adjust tau2 to current cursor position (M)**

Applicability: Systems with a solids module.

Syntax: `tshift`

Description: Adjusts `tau2` to make the current time cursor position the start of acquisition. As the time-domain cursor can move between points, this macro allows the accurate adjustment of `tau2` so as to start another acquisition exactly at the top of an echo.

See also: *User Guide: Solid-State NMR*

tspoil **Gradient spoiling time (P)**

Applicability: Systems with imaging capabilities.

Description: Delay parameter for use in controlling a spoiling gradient. Many imaging sequences use `tspoil` to set the additional time that the slice-select gradient is on, symmetrically bracketing the 180° refocusing pulse, to spoil any magnetization excited by the 180 itself.

See also: *User Guide: Imaging*

Related: `gcrush` Crusher gradient level (P)
`gspoil` Spoiler gradient level (P)
`tcrush` Crusher gradient control (P)

tugain **Amount of receiver gain used by qtune (P)**

Applicability: ^{UNITY}*INOVA*, *UNITYplus*, *UNITY*, and *VXR-S* systems.

Description: Sets the amount of receiver gain used by the interactive probe tuning program `qtune`. On some systems, the default receiver gain of 50 causes the signal to saturate, which `qtune` displays as a mostly flat line. To adjust the receiver gain to avoid saturation, set `tugain` to an appropriate value for the system before `qtune` is started.

Values: 0 to 60, in steps of 2 dB (60 represents the highest possible receiver gain and 0 the lowest). On ^{UNITY}*INOVA* and *UNITYplus* (500-MHz and higher), low-band gain is limited 18 to 60.

See also: *Getting Started*

Related: `qtune` Tune probe using swept-tune graphical tool (C)

tune **Assign a frequency to a channel for probe tuning (C)**

Applicability: ^{UNITY}*INOVA* and *UNITYplus* systems.

Syntax: (1) `tune(freq1, <freq2, freq3, freq4>)`
(2) `tune(chan1, freq1, <chan2, freq2, . . . >)`

Description: Assigns a frequency to a channel when tuning the probe. The frequency assignment remains in effect (as a tune frequency) until the next `su` or `go` command is executed. Although only the first synthesizer is connected to the tuning system, the console is programmed to set this synthesizer to the desired frequency based on the channel shown on the CHAN readout on the TUNE INTERFACE unit.

The `tune` program has two formats. If syntax 1 is used, frequencies are assigned to channels based on the order of the arguments. The first argument is interpreted and assigned to the first (observe) channel, the second argument is assigned to the second (decoupler) channel. A third or fourth argument would be interpreted and assigned in a similar manner.

If syntax 2 is used, the arguments are entered in pairs, with the first argument specifying the rf channel and the next argument specifying the frequency.

`tune` selects the format based on the first argument. If the first argument is a name for an rf channel, syntax 2 is assumed; otherwise, syntax 1 is used.

Arguments: `freq1`, `freq2`, `freq3`, and `freq4` specify the frequency of the rf channel as a value in MHz (e.g., 200 or 300) or indirectly using the nucleus for tuning the probe (e.g., 'H1' or 'C13'). If a nucleus is entered, it must be found in the nucleus table. The frequency of any channel without an argument is unaffected. For example, `tune('H1', 'C13', 'N15')` sets the first channel to tune at the ^1H , the second channel at ^{13}C , and the third channel at ^{15}N . If a fourth channel is present, it is not affected. Entering `tune('H1', 'C13', 200)` assigns the same frequencies for the first and second channels but the third channel tunes to 200 MHz, regardless of the proton frequency.

`chan1`, `chan2`, `chan3`, and `chan4` specify the channel directly:

- 'todev' or 'ch1' specify channel 1 (observe transmitter).
- 'dodev' or 'ch2' specify channel 2 (first decoupler).
- 'do2dev' or 'ch3' specify channel 3 (second decoupler).
- 'do3dev' or 'ch4' specify channel 4 (third decoupler).

Only one of these keywords is used per channel (do not enter the channel using just its number). If a channel does not have a keyword entered as an argument, that channel is not affected (e.g., `tune('ch4', 'P31')` selects the frequency corresponding to ^{31}P on the fourth channel, but leaves the first three channels unaffected).

Examples: `tune('H1', 'C13', 'N15')`
`tune('H1', 'C13', 200)`
`tune('ch4', 'P31')`

See also: *Getting Started*

Related:	<code>btune</code>	Tune broadband channel on <i>MERCURY-Series</i> , <i>GEMINI 2000</i> (M)
	<code>ctune</code>	Tune carbon channel on $^1\text{H}/^{13}\text{C}$ <i>GEMINI 2000</i> (M)
	<code>dfrq</code>	Transmitter frequency of first decoupler (P)
	<code>dfrq2</code>	Transmitter frequency of second decoupler (P)
	<code>dfrq3</code>	Transmitter frequency of third decoupler (P)
	<code>dtune</code>	Tune lock channel on <i>GEMINI 2000</i> (M)
	<code>go</code>	Submit experiment to acquisition (C)
	<code>htune</code>	Tune proton channel on <i>GEMINI 2000</i> (M)
	<code>qtune</code>	Tune probe using swept-tune graphical tool (C)
	<code>sfrq</code>	Transmitter frequency of observe nucleus (P)
	<code>spcfrq</code>	Display frequencies of rf channels (M)
	<code>su</code>	Submit a setup experiment to acquisition (C)
	<code>tuneoff</code>	Turn off probe tuning mode, <i>MERCURY-Series</i> , <i>GEMINI 2000</i> (M)

`tuneoff` Turn off probe tuning mode on *MERCURY-Series*, *GEMINI 2000* (M)

Applicability: *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems.

Syntax: `tuneoff`

T

Description: Takes a *MERCURY-VX*, *MERCURY*, *GEMINI 2000* broadband, or *GEMINI 2000* $^1\text{H}/^{13}\text{C}$ system out of tuning mode by turning off the transmitter directing rf to the probe. After entering `tuneoff`, be sure to change the cables on the probe and magnet leg back to the normal BNC connectors (as they were before they were moved for tuning purposes).

See also: *Getting Started; Autoswitchable NMR Probes Installation*

Related:	<code>btune</code>	Tune broadband channel on <i>MERCURY</i> -Series, <i>GEMINI 2000</i> (M)
	<code>ctune</code>	Tune carbon channel on $^1\text{H}/^{13}\text{C}$ <i>GEMINI 2000</i> (M)
	<code>dtune</code>	Tune lock channel on <i>GEMINI 2000</i> (M)
	<code>htune</code>	Tune proton channel on <i>GEMINI 2000</i> (M)
	<code>sethw</code>	Set values for hardware in acquisition system (C)
	<code>su</code>	Submit a setup experiment to acquisition (M)

U

undospins Restore spin system as before last iterative run (M)

Syntax: `undospins`

Description: Returns the values of the line assignments and the chemical shifts and coupling constants existing before the last iterative adjustment with `spins('iterate')`, and then runs `spins`. The parameters are returned from the file `spini.inpar` and the transitions from the file `spini.savela` in the current experiment.

See also: *User Guide: Liquids NMR*

Related: `spins` Perform spin simulation calculation (C)

undosy Restore original 1D NMR data from subexperiment (M)

Syntax: `undosy`

Description: Restores the 1D DOSY data stored by the `dosy` macro (if data exists) by recalling the data stored in the file `subexp/dosy2Ddisplay` in the current experiment. `undosy` and `redosy` enable easy switching between the 1D DOSY data (spectra as a function of `gzlv11`) and the 2D DOSY display (signal as a function of frequency and diffusion coefficient).

See also: *User Guide: Liquids NMR*

Related: `dosy` Process DOSY experiments (M)
`redosy` Restore 2D DOSY display from subexperiment (M)

unit Define conversion units (C)

Syntax: `unit<(suffix,label,m<,tree><,'mult'|'div'> \`
`,b<,tree><,'add'|'sub'>>`

Description: Defines a linear relationship that can be used to enter parameters with units. The unit is applied as a suffix to the numerical value (e.g., 10k, 100p). The definition of the linear relations follows the traditional $y=mx+b$ equation, where x is the input value and y is the converted result.

Entering the `unit` command with no arguments displays all currently defined units. To remove a unit, define the unit with a 0 for the slope.

A convenient place to put `unit` commands for all users is in the `bootup` macro. Put private `unit` commands in a user's `login` macro.

Arguments: `suffix` is a string identifying the name for the unit. The length of the string is limited to 12 characters.

`label` is a string for the name to be displayed when the `axis` parameter is set to the value of the suffix (if the suffix is only a single character). The length of the string is limited to 12 characters.

`m` is the slope of the linear relationship, defined either as a numerical value or as the name of a parameter. If a parameter name is used, it may be optionally followed with the parameter `tree` to use (argument `tree`) and by another optional keyword that specifies whether the parameter value should be a multiplier (keyword `'mult'`) or divisor (keyword `'div'`).

`tree` is the parameter tree to use (i.e., 'current', 'processed', 'global', or 'systemglobal'). The default tree is 'current'.
 'mult' is a keyword that specifies that a parameter value used for the slope should be a multiplier. This is the default for the slope.

'div' is a keyword that specifies that a parameter value used for the slope should be a divisor.

`b` is the intercept of the linear relationship, defined either as a numerical value or as the name of a parameter. If a parameter name is used, it may be optionally followed with the parameter tree to use (argument `tree`) and by another optional keyword that specifies whether the parameter value should be added (keyword 'add') or subtracted (keyword 'sub').

'add' is a keyword that specifies that a parameter value used for the intercept should be added. This is the default for the intercept.

'sub' is a keyword that specifies that a parameter value used for the intercept should be subtracted.

Examples: `unit`

Displays all currently defined units

```
unit('k','kHz',1000)
```

`r1=10k` will set `r1` to 10000

```
unit('p','ppm','reffrq','processed')
```

`r1=10p` will set `r1` to $10 \times \text{reffrq}$, where `reffrq` from processed tree

```
unit('p','',0)
```

`r1=10p` will set `r1` to 10 and give an error "unknown unit p"

```
unit('F','degF',5/9,-32*5/9)
```

`r1=212F` will set `r1` to 100 (degrees C)

```
unit('C','degC',9/5,32)
```

`r1=100C` will set `r1` to 212 (degrees F)

See also: *Getting Started, VNMR User Programming*

Related: `axis` Axis label for displays and plots (P)
`bootup` Macro executed automatically when VNMR is activated (M)

`unix_vxr` **Convert UNIX text files to VXR-style format (M,U)**

Syntax: (From VNMR) `unix_vxr(UNIX_file,VXR_file)`
 (From UNIX) `unix_vxr UNIX_file VXR_file`

Description: Converts a UNIX text file to the VXR-style format used with Gemini, VXR-4000, and XL systems. The conversion must be done before moving the file to the VXR system.

Arguments: `UNIX_file` is the name of the input file, which must be a text file.
`VXR_file` is the name of the output file after conversion. The names of the input and output files must be different.

Examples: (From VNMR) `unix_vxr('oldtextfile','newtextfile')`
 (From UNIX) `unix_vxr oldtextfile newtextfile`

See also: *Getting Started*

Related: `convert` Convert data set from a VXR-style system (C,U)
`decomp` Decompose a VXR-style directory (C)
`vxr_unix` Convert VXR-style text files to UNIX format (M,U)

unlock **Remove inactive lock and join experiment (C)**

Syntax: `unlock(exp_number, 'force')`

Description: In attempting to join another experiment, the `jexp` command may abort claiming the experiment is locked. This feature prevents two users from processing the same experimental data at the same time, which could corrupt the data (a “user” can also be a background operation invoked by the same user, such as in `wexp` processing). This lock can be left behind if the program or the computer crashes.

The `unlock` command removes the lock if it is inactive and joins the unlocked experiment. The command will fail if the lock is still active (i.e., the process that made the lock is still executing) or if the lock was placed on the experiment by a remote host. The latter situation can only occur when one or more nodes are sharing the same file system (and experimental data).

Arguments: `exp_number` is the number of the experiment from 1 to 9 to be unlocked.
`force` unlocks an experiment under all circumstances and joins the unlocked experiment.

Examples: `unlock(3)`

See also: *Getting Started*

Related: `jexp` Join existing experiment (C)

updatepars **Update all parameter sets saved in a directory (M)**

Syntax: `updatepars(directory)`

Description: Corrects saved parameter sets. Starting with VNMR version 4.2, all parameters, upper limit, lower limit, and step sizes have been tightened. Further additions were made in VNMR 4.3. `updatepars` searches a directory for parameter and FID files and corrects the `procpars` files found. This macro overwrites parameters in the current experiment. The corrections applied to the parameter sets are defined by the `parfix` macro. Because `updatepars` uses the current experiment to process the parameter sets, the experiment chosen for running `updatepars` should not contain a valuable data set.

Arguments: `directory` is the name of the directory to be searched.

Examples: `updatepars('myparlib')`
`updatepars('mydata')`

See also: *Getting Started*

Related: `parfix` Update parameter sets (M)
`parversion` Version of parameter set (P)

updateprobe **Update probe file (M)**

Syntax: `updateprobe(<probe|'tmpl't'><,'system'>)`

Description: Updates the current existing probe file or probe template.

Arguments: `probe` is the probe parameter to update. The default is the current probe parameter value.

`'tmpl't'` is a keyword to update the local probe template. The default is the current probe file.

`'system'` is a keyword to update the system template or probe file, providing you have write permission to the file. The default is to update the local template or probe file.

Examples: `updateprobe`
`updateprobe('autosw')`
`updateprobe('autosw','system')`
`updateprobe('tmpl')`

See also: *Getting Started*

Related: `addparams` Add parameter to current probe file (M)
`getparam` Receive parameter from probe file (M)
`setparams` Write parameter to current probe file (M)

updaterev Update after installing new VNMR version (M)

Syntax: `updaterev`

Description: Updates experiment parameters and the global file following installation of a new VNMR software version. `updaterev` is called by the `makeuser` command during the installation process.

See also: *VNMR and Solaris Software Installation*

updtgcoil Update gradient coil (M)

Applicability: Systems with three-axis gradients.

Syntax: `updtgcoil`

Description: Creates the `gcoil` parameter, if it does not exist, and sets it to the current value of the system gradient coil `sysgcoil`. `updtgcoil` only executes if gradients are configured in the system.

The `updtgcoil` macro is called when a new experiment is joined or new parameters are read into an experiment; however, it is only called at these times if the `gcoil` parameter exists. If `sysgcoil` is set to a gradient table name and if the values of `sysgcoil` and `gcoil` are different, a message is displayed in the Status window to let the user know that the gradient coil parameters have been updated.

`updtgcoil` can be called directly if the user wants to update the parameter set with the `gcoil` and gradient table parameters.

See also: *Getting Started; VNMR User Programming; User Guide: Imaging*

Related: `createtable` Generate system gradient table (M)
`gcoil` Read data from gradient calibration tables (P)
`sysgcoil` System gradient coil (P)

updtparam Update specified acquisition parameters (C)

Syntax: `updtparam`

Description: Enables interactive updating of specified acquisition parameters.

See also: *SpinCAD*

Related: `psgupdateoff` Prevent update of acquisition parameters (C)
`psgupdateon` Enable update of acquisition parameters (C)

usemark Use “mark” output as deconvolution starting point (M)

Syntax: `usemark`

Description: In some cases it is not possible to produce a line list that is a suitable starting point for a deconvolution (e.g., lines may overlap so severely that a line list does not find them). In this case, or in any case, the results of a “mark” operation

during a previous spectral display (**ds**) may be used to provide a starting point. If the “mark” has been made with a single cursor, the information in the file `mark1d.out` contains only a frequency and intensity, and the starting linewidth is taken from the parameter **slw**.

If the “mark” is made with two cursors, placed symmetrically about the center of each line at the half-height point, `mark1d.out` contains two frequencies and an intensity. In this case, the starting frequency is taken as the average of the two cursor positions; the starting linewidth is taken as their difference (thus allowing different starting linewidths for each line).

See also: *User Guide: Liquids NMR*

Related: **ds** Display a spectrum (C)
slw Spin simulation linewidth (P)

userdir VNMR user directory (P)

Description: Stores the full UNIX path of the directory that contains a user's private VNMR files. These include a user's private `maclib`, `menulib`, `shims`, `psglib`, `experiments`, etc. This parameter is initialized at bootup by the UNIX environmental variable `vnmruser`.

Values: Typical value is `/home/vnmr2/vnmrsys`

See also: *Getting Started*

Related: **curexp** Current experiment directory (P)
systemdir VNMR system directory (P)

usergo Experiment setup macro called by **go**, **ga**, and **au** (M)

Syntax: `usergo`

Description: Called by macros **go**, **ga**, or **au** before starting an experiment. The user typically creates `usergo` as a means to set up general experiment conditions.

See also: *Getting Started*

Related: **au** Submit experiment to acquisition and process data (M)
ga Submit experiment to ac acquisition and FT the result (M)
go Submit experiment to acquisition (M)
go_ Pulse sequence setup macro called by **go**, **ga**, and **au** (M)

userfixpar Macro called by **fixpar** (M)

Syntax: `userfixpar`

Description: Called by the macro **fixpar** to provide an easy mechanism to customize parameter sets.

See also: *Getting Started*

Related: **fixpar** Correct parameter characteristics in experiment (M)

vast1d Set up initial parameters for VAST experiments (M)

Applicability: Systems with VAST accessory.

Syntax: `vast1d`

Description: Sets up initial VAST parameters from the `/vnmr/stdpar` directory or from the user's `stdpar` directory if the appropriate file exists there. Any changes made to the files in these directories are reflected in the setup. The file `/vnmr/stdpar/vast1d.par` contains the “default” parameters for VAST spectra and should be modified as needed to produce spectra under desirable conditions. `vast1d` is typically run by using the corresponding button in the menu system. After running `vast1d`, the solvent parameter can be set by choosing it from the list of solvents listed in `/vnmr/solvents`.

See also: *User Guide: Liquids NMR*

vastget Selects and displays VAST spectra (M)

Applicability: Systems with VAST accessory.

Syntax: `vastget(<well>, <well>, ...)>`

Description: Selects and displays the spectra from any arbitrary well or wells using the well label(s) as arguments. the spectra are displayed in a `dss` stacked plot.

Arguments: `well` is the well label from which you want to select and display spectra. The wells are labeled `[A->H][1-8]`.

Examples: `vastget('B6', 'B7', 'C11', 'G3')`

See also: *User Guide: Liquids NMR*

vastglue Assemble related 1D datasets into a 2D (or pseudo-2D) dataset (M)

Applicability: Systems with the VAST accessory.

Syntax: `vastglue(<rack>, <zone>)`
`vastglue(<glue order>, <plate>)`

Description: Used to artificially reconstruct a 2D dataset from a series of 1D data sets having similar filenames. It is crucial to ensure that the format of the file names of each of the 1D data sets is identical. `vastglue` reads in each 1D file, in succession, and adds it to the previous data, but in a 2D format. It assumes that file names are of the format obtained when using the default setting of `autoname` (`autoname=''`). If `autoname` has been redefined, use a macro like `vastglue2`. Save the resulting reconstructed 2D dataset in the normal manner using `svf`.

Arguments: `rack` is the rack number; the default is 1. If you enter a `rack` number, you must also enter a `zone` number.

`zone` is the zone number; the default is 1. If you want to specify a `zone` number, you must enter a `rack` number.

`glue order` is the specific glue order to be defined based on the order defined in a `plate_glue` file. If `glue order` is specified, you can provide a

plate number as the second argument and used with the `glue order` argument.

See also: *User Guide: Liquids NMR*

Related: `autoname` Prefix for automation data file (P)
`vastglue2` Assemble related 1D datasets into a 2D (or pseudo-2D) dataset (M)

vastglue2 Assemble related 1D datasets into a 2D (or pseudo-2D) dataset (M)

Applicability: Systems with the VAST accessory

Syntax: `vastglue2<(number)>`

Description: Used to artificially reconstruct a 2D data set from a series of 1D datasets having similar filenames. It is crucial to ensure that the format of the file names of each of the 1D datasets is identical. `vastglue2` reads in each 1D file, in succession, and adds it to the previous data, but in a 2D format. It assumes that file names are of the format obtained using a nondefault setting of `autoname` (`autoname='filename_R%RACK:%_Z%ZONE:%_S%SAMPLE#:%_'`). This definition must be hard coded into the macro by the user. If `autoname` has not been redefined, use a macro like `vastglue`. Save the resulting reconstructed 2D data set in the normal manner using `svf`.

Arguments: `number` is used to specify that only spectra from 1 through `number` are to be glued. The default is to glue all the spectra stored in the current directory that have the proper file name format (from 1 through `arraydim`).

See also: *User Guide: Liquids NMR*

Related: `autoname` Prefix for automation data file (P)
`vastglue` Assemble related 1D datasets into a 2D (or pseudo-2D) dataset (M)

vastgo Turn off LC stopped flow automation and start VAST automation (M)

Applicability: Systems with the LC-NMR and VAST accessory

Syntax: `vastgo`

Description: Turns off LC stopped flow use of automation and starts VAST automation run.

vbg Run VNMR processing in background (U)

Syntax: (From UNIX) `vbg exp_number command_string <prefix>`

Description: Enables user to perform VNMR tasks in the background. `vbg` (for “VNMR background processing”) must be run from within a UNIX shell, and *no* foreground or other background processes can be active in the designated experiment (e.g., if you are working in `exp2` in VNMR (in the foreground), you cannot execute background processing in `exp2` as well).

Foreground processing causes a lock file to be placed in the appropriate experiment. The file has a format such as `f.1268`, where 1268 indicates the process number in the process table (accessed in UNIX by entering the command `ps -e`). Background processing causes a lock file to be in the appropriate experiment as well. This file has a format such as `b.4356`, where 4356 indicates the process number. By displaying the files within an experiment, the user can readily determine whether any foreground or background processes are active in that experiment.

Arguments: `exp_number` is the number of the experiment, from 1 to 9, in the user’s directory in which the background processing is to take place.

`command_string` is the command string to be executed by VNMR in the background. Double quotes enclosing the string are mandatory (e.g., `"fn=4096 fn1=2048 wft2da"`).

`prefix` is a prefix to be added to the name of the log file, making the name `prefix_bgf.log`. The default name is `exp_number_bgf.log`, where `exp_number` is the experiment number. The log file is placed in the experiment in which the background processing takes place.

Examples: (From UNIX) `vbg 1 "wft2da bc('f1')"`
 (From UNIX) `vbg 3 "vsadj pl pscale pap page" plotlog`

See also: *VNMR User Programming*

vf Vertical scale of FID (P)

Description: In normalized intensity (**nm**) mode, `vf` is the height of the largest FID. In absolute intensity (**ai**) mode, `vf` is a multiplier that is adjusted to produce a desired vertical scale, using the appearance on the display screen as a guide (full scale on the screen gives full scale on the plotter).

`vf` can be entered in the usual way or interactively controlled by clicking the middle mouse button in the graphics window during a FID display (click above the FID to increase `vf` or below the FID to decrease it).

Values: 1e-6 to 1e9, in mm (in **nm** mode) or as a multiplier (in **ai** mode).

See also: *Getting Started*

Related:	ai	Select absolute intensity mode (C)
	df	Display a single FID (C)
	nm	Select normalized intensity mode (C)
	sf	Start of FID (P)
	wf	Width of FID (P)

vi Edit text file with vi text editor (M)

Syntax: `vi (file)`

Description: Invokes the UNIX text editor `vi` for editing the file name given. On the Sun workstation, a popup screen contains the editing window. On the GraphOn terminal, the main screen becomes the editing window. `vi` is a powerful text editor, but its user interface is limited: the mouse is not used, menus are not available, and status information is virtually nonexistent.

`vi` operates in three modes: the *command mode* (for moving the cursor and editing text), the *insert mode* (for inserting text into the file), and the *last line mode* (for special operations). Each mode is described below.

Command mode

`vi` starts up in the command mode. In this mode, user commands consist mostly of a single character, sometimes in combination with another character, or a number, or both. A number preceding a command typically defines how many times a command should be executed (e.g., `3dd` means delete three lines). The commands available include the following:

G	go to the start of the last line in the file
3G	go to the start of line 3
0	(zero) go to the start of the current line
\$	go to the end of the current line
Return or +	go to start of next line

-	(hyphen) go to start of previous line
Ctrl-d	scroll down (forward) half a screen
Ctrl-f	scroll forward by a full screen
Ctrl-u	scroll up (back) half a screen
Ctrl-b	scroll back by a full screen
/expression	find next <i>expression</i> and jump to its first character
?expression	find previous <i>expression</i> , jump to its first character
n	find next <i>expression</i> (from the last search)
N	find previous <i>expression</i> (from the last search)
dd	delete one line and put it into the buffer
3dd	delete three lines and put them into the buffer
dw	delete word
x	erase one character forward (under cursor)
X	erase one character backwards (before cursor)
3x	erase three characters forward
rcharacter	erase character and replace with <i>character</i>
ZZ	write if necessary and quit <i>vi</i>
.	(period) repeat the last command
u	undo the last command
J	join the next line to the current line
yy or Y	yank one line and put into a buffer (called yank buffer)
p	put contents of yank buffer after the cursor
P	put contents of yank buffer before the cursor
"aY	yank line into buffer a (buffers b to z also available)
"ap	put contents of buffer a below current line
"aP	put contents of buffer a above current line

Because there is no command line, these commands do not show up on the screen but are *executed immediately* (without pressing the Return key).

Insert mode

In the insert mode, characters typed on the keyboard (except for the Esc key) show up in the text. The insert mode is entered by typing one of the following commands from the command mode:

a text Esc	append <i>text</i> after the current cursor position
A text Esc	append <i>text</i> to the end of current line
i text Esc	insert <i>text</i> before current cursor position
cw word Esc	change <i>word</i> from current cursor position to end
2cw words Esc	change two <i>words</i> from current cursor position to end
o text Esc	open line below current line and append <i>text</i>
O text Esc	open line above current line and append <i>text</i>

The only way to exit the insert mode is by pressing the Esc key, which leads back to the command mode. Unfortunately, there is no indication on the screen whether *vi* is in the command mode or in the insert mode. Inexperienced users often press the Esc key to make sure they are still in the command mode. The Esc key can also be used to avoid execution of commands that have been typed partially (e.g., the number has been typed, but not the last character).

You can insert special (normally nondisplayable) characters into the text if they are preceded by a Ctrl-v (e.g., entering Ctrl-v Ctrl-q is displayed in the text as ^Q).

Changing selected occurrences

The following actions find one or more occurrences of a particular word and change it to another word:

- First, type `/word` and press Return, where `/` is a forward slash and `word` is word you want to change.
- Next, press `n` as necessary until you reach the occurrence of the word you want to change.
- Finally, type `cw newword` and press Esc, where `newword` is replacement word.
- To repeat for another occurrence of `word`, press `n` as necessary to scan forward, and then type `.` (a period) to repeat `cw newword` (or whatever was the last change)

Changing selected occurrences of an expression (one or more words) is similar. To change two words, for example, take the same actions as above but use the command `2cw` (or `c2w`) instead.

Last line mode

The last line mode is initiated with a colon; thereafter, commands such as the following can be used (press Return to execute these commands):

```
:r filename      read file named filename (insert in currently open file)
:w              write (save) file
:w filename      write under a new file named filename
:e filename      edit a different file named filename
:q              quit vi (only possible if file has been written back)
:wq             write back file (save changes) and quit vi
:q!            quit vi without saving changes
```

Exiting from `vi` is accomplished by using the `ZZ` command in the command mode, or with the `:q`, `:wq`, or `:q!` commands in the last line mode.

This description lists only a selection of the most important commands. For more information on `vi`, refer to UNIX books and manuals.

Examples: `vi(userdir+'psglib/apt.c')`
`vi(curexp+'text')`

See also: *VNMR User Programming*

Related:	<code>edit</code>	Edit a file with user-selectable editor (M)
	<code>paramvi</code>	Edit a parameter and its attributes with <code>vi</code> text editor (M)
	<code>macrovi</code>	Edit a user macro with the <code>vi</code> text editor (C)
	<code>menuvi</code>	Edit a menu with the <code>vi</code> text editor (M)
	<code>textvi</code>	Edit text file of current experiment (M)

vn

Start VNMR directly (U)

Syntax: (From UNIX) `vn <-display Xserver> <-fn font> &`

Description: Starts the VNMR application directly without checking the operating system and attempting to run the window manager.

Arguments: `-display Xserver` specifies X server display (e.g., `hostname:0.0`). The default is the environment set by the `DISPLAY` variable.

`-fn` `font` specifies the size of the font displayed (e.g., 9x15, 8x13, or 7x13). The default is the font set in the `.Xdefaults` file. Note that the size of the font affects the size of the VNMR window.

Examples: `vn &`
`vn -display hostname:0.0 &`
`vn -font 8x13 &`

See also: *Getting Started*

Related: `vnmr` Start VNMR (U)

vnmr Start VNMR in current windowing system (U)

Syntax: (From UNIX) `vnmr`

Description: Starts the VNMR application using the current windowing system. If the Open Window system is running on the Sun, `VnmrX` starts. If Motif is running on an IBM workstation, `VnmrI` starts. `vnmr` can also be used to start VNMR from terminals. In this case, the `vnmr` command is equivalent to the `vn` command.

See also: *Getting Started*

Related: `vn` Start VNMR in window environment (U)

vnmr2sc VNMR to SpinCAD pulse sequence translator (M)

Syntax: `vnmr2sc<('sequence_name'<,rfchannels<,gradchannels>>>)`

Description: Converts the pulse sequence pointed to by the `seqfil` parameter in the current VNMR parameter set from a C program into a SpinCAD pulse sequence. The conversion result is stored in the local `spincad/psglib` under the same name as the C pulse sequence (i.e., the name stored in the `seqfil` parameter), but without the `.c` extension.

`vnmr2sc` uses `dps` output to generate the SpinCAD code, i.e., the pulse sequence must be compiled and must be displayable with `dps`. Pulse sequences that do not compile with the `dps` option cannot be translated. For the same reason, `vnmr2sc` cannot translate features that do not show up in `dps`. This means that `go`-time decisions (such as flag-based `C if` constructs) will *not* show up in the translated SpinCAD sequence. In such cases, you have two options:

- Translate the sequence several times, once for each of the relevant flag settings. That is, generate several (simpler) SpinCAD pulse sequences from a single C sequence.
- Translate the sequence once (preferably with all options turned on), then manually insert the necessary `if` statements and other missing elements using SpinCAD.

Arguments: `sequence_name` is an optional argument that permits the name of the resulting SpinCAD pulse sequence to be specified. By default, `vnmr2sc` creates a SpinCAD sequence with the name specified in the `seqfil` parameter (i.e., the SpinCAD sequence has the same name as the C pulse sequence). `sequence_name` is particularly useful if a C sequence is to be translated into multiple SpinCAD sequences; see the examples.

`rfchannels` is an optional numeric argument specifying the number of rf channels. Use it when you want the SpinCAD sequence to address more rf channels. By default, `vnmr2sc` determines the number of rf channels from the source sequence. You can only *increase* the number of rf channels. If you specify 0 rf channels, the number of rf channels is left unchanged.

`gradchannels` is a second optional numeric argument specifying the number of gradient channels or axes. Use it when you want to convert a nongradient sequence to a gradient sequence or when you want the SpinCAD sequence to address more gradient axes than the source sequence. By default, `vnmr2sc` determines the number of gradient axes from the source sequence. You can only *increase*, not decrease, the number of gradient axes.

Examples: `vnmr2sc`
`setup('H1','CDCl3') hmqc null=0.2 vnmr2sc`
`null=0 mbond='y' vnmr2sc('hmbc')`
`vnmr2sc('gcosy',2,3)`
`nt=256 vnmr2sc`
`vnmr2sc(4,1)`
`vnmr2sc(0,1)`

See also: *SpinCAD Manual*

Related: `dps` Display pulse sequence (C)
`spincad` Open graphical pulse sequence generation utility (C)

`vnmr_accounting` Open VNMR Accounting window (U)

Syntax: (From UNIX) `vnmr_accounting`

Description: Opens a window for creating and maintaining cost accounting data for groups of users on a spectrometer system. The program accommodates multiple rate schedules for spectrometer usage. A calendar tool can be used to define holidays for holiday rates. There is no limit on the number of rates that can be defined. Multiple printers can be selected.

Any user can view the accounting information (enter `cd /vnmr/bin` followed by `./vnmr_accounting`), but to update information, the user must have root privileges.

See also: *System Administration*

`vnmrexit` Exit from the VNMR system (C)

Syntax: `vnmrexit`

Description: Exits from the VNMR system in a graceful manner by writing parameters and data to the disk, removing lock files, and restoring the terminal (if on a GraphOn). To provide flexibility when exiting VNMR, the macro `exit` calls `vnmrexit` to exit from VNMR.

CAUTION: When you exit from the VNMR user interface on your X display system, whether you are using an X terminal or a Sun computer, and whether you are using OpenWindows, CDE, or Motif, you must first exit from any copy of VNMR running on your system. Failure to do this can cause current parameter values and even current data to be lost.

See also: *Getting Started*

Related: `exit` Call the `vnmrexit` command (M)

`vnmrplot` Plot files (U)

Syntax: (From UNIX) `vnmrplot <file>`

Description: A UNIX command that plots files from inside VNMR commands. To plot a file, you should use the `page` command, which uses `vnmrplot` internally.

Arguments: `file` is the name of the file to be plotted.

See also: *Getting Started*

Related: `vnmrprint` Print text files (U)

vnmrprint Print text files (U)

Syntax: (From UNIX) `vnmrprint printfile <printcap>`
`<printer_type <clear|file>>`

Description: A UNIX command installed as part of the VNMR system to print text files. The `printon` and `printoff` commands use `vnmrprint` to print files. `vnmrprint` can also be used to delete a print file or save a print file to a different name.

Arguments: `printfile` is the name of the text file to be printed.
`printcap` is a UNIX printcap entry (e.g. `LaserJet_300`) for the printer to print the text file. The default is the printer selected by the `-p` option of the UNIX `lp` command.
`printer_type` is the type of printer from the list of VNMR printers (e.g., `LaserJet_300`). `printer_type` is required as an argument when it is desired to clear the printer file or save the printer file to another name.
`clear` is a keyword to delete the current print file. Deleting this file also requires that the `printfile`, `printcap`, and `printer_type` arguments be entered so that `clear` is the fourth argument.
`file` is the name of the file to use in saving the `printfile`. If a file with the name specified already exists, it is overwritten. Saving the file also requires that the `printfile`, `printcap`, and `printer_type` arguments be entered so that `file` is the fourth argument.

Examples: `vnmrprint /vnmr/psglib/tocsy.c LaserJet_300`
`vnmrprint myfile LaserJet_300 LaserJet_300 clear`
`vnmrprint myfile ps PS_AR yourfile`

See also: *Getting Started*

Related: `printoff` Stop sending text to printer and start print operation (C)
`printon` Direct text output to printer (C)
`vnmrplot` Plot files (U)

vo Vertical offset (P)

Description: For 1D data sets, sets the vertical offset of the each spectrum in a *stacked display* with respect to the previous spectrum. The parameter `ho` sets the horizontal offset. For a “left-to-right” presentation, `ho` is typically negative; for a “bottom-to-top” presentation, `vo` is positive.

For 2D data sets, the parameter `wc2` sets the distance between the first and last trace and the `vo` parameter is inactive.

Values: Number, in mm.

See also: *Getting Started; User Guide: Liquids NMR*

Related: `ho` Horizontal offset (P)
`wc2` Width of chart in second direction (P)

vox1, vox2, vox3 Voxel dimensions (P)

Applicability: Systems with imaging capabilities.

Description: Defines the dimensions of a desired voxel for localized spectroscopy experiments.

Values: Number, in mm.

See also: *User Guide: Imaging*

Related: [transfer](#) Move parameters to target experiment (M)

voxplan Set voxel parameters for voxel defined by 2D box cursor (M)

Applicability: Systems with imaging capabilities.

Syntax: `voxplan`

Description: Calculates and sets the voxel parameters for the voxel defined by the position of the 2D box cursor. The parameter for the voxel can be calculated and set using the Calculate Target button of the voxel planning menu. This uses the `voxplan` macro. See the [plan](#) macro for details.

See also: *User Guide: Imaging*

Related: [drawslixw](#) Display target slices (M)
[drawvox](#) Display target voxels (M)
[plan](#) Display menu for planning a target scan (M)
[ssplan](#) Set slice parameters for target slice (M)

vp Vertical position of spectrum (P)

Description: Contains vertical position of spectrum with respect to the bottom of the display or plotter.

Values: -200 to +200, in mm.

See also: *Getting Started*

Related: [vpf](#) Current vertical position of FID (P)
[vpfi](#) Current vertical position of imaginary FID (P)

vpf Current vertical position of FID (P)

Description: Contains the current vertical position of an FID. To create this parameter and the other FID display parameters [axisf](#), [crf](#), [deltaf](#), [dotflag](#), and [vpfi](#) (if the parameter set is older and lacks these parameters), enter `addpar('fid')`.

Values: Number, in mm. If `vpf=0`, the FID is positioned in the middle of the screen.

See also: *Getting Started*

Related: [addpar](#) Add selected parameters to the current experiment (M)
[axisf](#) Axis label for FID displays and plots (P)
[crf](#) Current time-domain cursor position (P)
[deltaf](#) Difference of two time-domain cursors (P)
[dotflag](#) Display FID as connected dots (P)
[vp](#) Vertical position of spectrum (P)
[vpfi](#) Current vertical position of imaginary FID (P)

vpfi Current vertical position of imaginary FID (P)

Description: Contains the current vertical position of the imaginary part of an FID. To create this parameter and the other FID display parameters [axisf](#), [crf](#), [deltaf](#), [dotflag](#), and [vpf](#) (if the parameter set is older and lacks these parameters), enter `addpar('fid')`.

Values: Number, in mm. In `vpfi=0`, the imaginary part is positioned in the middle of the screen.

See also: *Getting Started*

Related: `addpar` Add selected parameters to the current experiment (M)
`axisf` Axis label for FID displays and plots (P)
`crf` Current time-domain cursor position (P)
`deltaf` Difference of two time-domain cursors (P)
`dotflag` Display FID as connected dots (P)
`vp` Vertical position of spectrum (P)
`vpf` Current vertical position of FID (P)

vs Vertical scale (P)

Description: In normalized (`nm`) mode, `vs` is the height of the largest peak in the spectrum. In absolute intensity (`ai`) mode, `vs` is a multiplier that is adjusted to produce a desired vertical scale, using the appearance on the display screen as a guide (full scale on the screen gives full scale on the plotter). `vs` can be entered in the usual way or interactively controlled by clicking the middle mouse button on a display.

Values: 1e-6 to 1e9, in mm (in `nm` mode) or as a multiplier (in `ai` mode).

See also: *Getting Started; User Guide: Liquids NMR*

Related: `ai` Select absolute intensity mode (C)
`isadj` Adjust integral scale (M)
`nm` Select normalized intensity mode (C)
`thadj` Adjust threshold for peak printout (M)
`vsadj` Automatic vertical scale adjustment (M)
`vsadj2` Automatic vertical scale adjustment by powers of two (M)
`vsadjc` Automatic vertical scale adjustment for ¹³C spectra (M)
`vsadjh` Automatic vertical scale adjustment for ¹H spectra (M)

vs2d Vertical scale for 2D displays (P)

Description: Sets a multiplier for 2D spectra and images that is adjusted to produce a desired vertical scale for display or plotting. `vs2d` takes the place of `vs` for 2D data display and can be adjusted by explicitly setting it to a value or by clicking the middle mouse button when pointing to a point on a 2D display. If `vs2d` does not exist, it can be created by running `par2d`.

Related: `par2d` Create 2D acquisition, processing, and display parameters (M)
`vs` Select vertical scale (C)
`vsproj` Adjust vertical scale for projections and traces (M)

vsadj Automatic vertical scale adjustment (M)

Syntax: `vsadj<(height)>`

Description: Automatically sets the vertical scale `vs` in the absolute intensity (`ai`) mode so that the largest peak is at the requested height.

Arguments: `height` is the desired height, in mm, of the largest signal in the displayed portion of the spectrum. The default is `0.9*(wc2max-vp-sc2)`.

Examples: `vsadj`
`vsadj(100)`

Alternate: Adj VS button in the 1D Data Manipulation menu.

See also: *Getting Started*

Related:	ai	Select absolute intensity mode (C)
	isadj	Adjust integral scale (M)
	thadj	Adjust threshold for peak printout (M)
	vs	Vertical scale (P)
	vsadj2	Automatic vertical scale adjustment by powers of two (M)
	vsadjc	Automatic vertical scale adjustment for ¹³ C spectra (M)
	vsadjh	Automatic vertical scale adjustment for ¹ H spectra (M)
	wc2max	Maximum width of chart in second direction (P)

vsadj2 Automatic vertical scale adjustment by powers of 2 (M)

Syntax: `vsadj2<(height)>:scaling_factor`

Description: Adjusts the vertical scale by powers of two as required for expansion plots (see **aexppl** for more information).

Arguments: `height` is desired height of largest (or largest relevant) signal in displayed portion of the spectrum. The default is $0.9 * (wc2max - vp - sc2)$.
`scaling_factor` returns to the calling macro the ratio of the new compared to the old value of **vs**.

Examples: `vsadj2`
`vsadj2(50):r1`

Alternate: Adj VS button in the 1D Data Manipulation menu.

See also: *Getting Started*

Related:	aexppl	Automatic expansions plot (M)
	isadj	Adjust integral scale (M)
	sc2	Start of chart in second direction (P)
	thadj	Adjust threshold for peak printout (M)
	vp	Vertical position of spectrum (P)
	vs	Vertical Scale (P)
	vsadj	Automatic vertical scale adjustment (M)
	vsadjc	Automatic vertical scale adjustment for ¹³ C spectra (M)
	vsadjh	Automatic vertical scale adjustment for H1 spectra (M)
	wc2max	Maximum width of chart in second direction (P)

vsadjc Automatic vertical scale adjustment for ¹³C spectra (M)

Syntax: `vsadjc<(height)>`

Description: Functionally the same as the macro **vsadj**, except excludes solvent and TMS signals from the carbon spectra for the adjustment of **vs**.

Arguments: `height` is desired height of largest (or largest relevant) signal in displayed portion of the spectrum. The default is $0.9 * (wc2max - vp - sc2)$.

Examples: `vsadjc`
`vsadjc(wc2max-sc2-wc2-5)`

Alternate: Adj VS button in the 1D Data Manipulation menu.

See also: *Getting Started*

Related:	isadj	Adjust integral scale (M)
	thadj	Adjust threshold for peak printout (M)
	vs	Vertical Scale (P)
	vsadj	Automatic vertical scale adjustment (M)
	vsadj2	Automatic vertical scale adjustment by powers of two (M)
	vsadjh	Automatic vertical scale adjustment for H1 spectra (M)

vsadjh Automatic vertical scale adjustment for ¹H spectra (M)

Syntax: `vsadjh<(height<,do_not_ignore_solvent>)>`

Description: Works as the same as the macro `vsadj`, except disregards solvent and TMS signals from proton spectra and, if from the remaining spectrum the highest line is more than three times as high as the second highest line, the spectrum is scaled to this second highest signal (otherwise the highest signal is taken as relevant).

Arguments: `height` is desired height of largest (or largest relevant) signal in displayed portion of the spectrum. If `height` is 0 or a negative value, it defaults to `0.9*(wc2max-vp-sc2)`, which is also the default with no arguments.
`do_not_ignore_solvent` is any second argument. If present, it signals `vsadjh` to not ignore the solvent line and regard the solvent line as normal signal (i.e, only exclude the TMS line). This argument was added for the situation where frequently there are high “real” signals at the position of the solvent line. Such signals could otherwise be regarded as solvent line and would then be ignored. This could then lead to overscaling in the result.

Examples: `vsadjh`
`vsadjh(0.7*wc2max)`

Alternate: Adj VS button in the 1D Data Manipulation menu.

See also: *Getting Started*

Related:	<code>isadj</code>	Adjust integral scale (M)
	<code>sc2</code>	Start of chart in second direction (P)
	<code>thadj</code>	Adjust threshold for peak printout (M)
	<code>vs</code>	Vertical scale (P)
	<code>vsadj</code>	Automatic vertical scale adjustment (M)
	<code>vsadj2</code>	Automatic vertical scale adjustment by powers of two (M)
	<code>vsadjc</code>	Automatic vertical scale adjustment for ¹³ C spectra (M)

vsproj Vertical scale for projections and traces (P)

Description: Sets a multiplier that is adjusted to produce a desired vertical scale for projections or traces of 2D data sets. `vsproj` can be explicitly adjusted by setting it to a value or by clicking the middle mouse button when pointing at the projection or trace. When interactively adjusting the scale with the mouse, the higher the pointer is in the trace display, the larger the vertical scale. If the parameter does not exist, it can be created by running the `par2d` macro.

Related:	<code>par2d</code>	Create 2D acquisition, processing, and display parameters (M)
	<code>vs</code>	Select vertical scale(C)
	<code>vs2d</code>	Adjust vertical scale for 2D displays (M)

vtc Variable temperature cutoff point (P)

Applicability: Systems with a variable temperature (VT) module.

Description: Sets a VT cutoff point. Above this temperature, VT air flows straight into the probe, past the heater, then past the sample. Below this temperature, air goes first through the heat exchange bucket, for cooling by the heat exchange fluid, and then into the probe and past the heater.

Values: 0 to 50, in degrees celsius. `vtc` is typically set 5°C higher than the supply gas used for VT regulation.

See also: *Getting Started; User Guide: Liquids NMR*

Related: `temp` Sample temperature (P)
`tin` Temperature interlock (P)

vttype Variable temperature controller present (P)

Description: In the CONFIG window, this parameter specifies whether a variable temperature (VT) controller is present or not on the system. The value is set using the VT Controller label in the CONFIG window (opened from `config`).

When entered from command line in VNMR, control of the variable temperature (VT) controller from the current experiment is either engaged (`vttype=2`) or disengaged (`vttype=0`). The current state of the variable temperature (VT) controller is not changed when `vttype` is set in the command window.

The variable temperature (VT) controller setting in CONFIG is not affected by entering `vttype` on the command line.

Values: 2 is setting for VT controller (Present choice in CONFIG window).

0 is setting for no VT controller (Not Present choice in CONFIG window).

Examples: If `temp= 'some temperature'` while `vttype=2` and `vttype` is then changed to `vttype=0` on the command line, the variable temperature (VT) controller will continue regulate the sample at the value set by `temp`. While `vttype=0` changes to `temp` will have no effect.

See also: *VNMR and Solaris Software Installation; User Guide: Liquids NMR*

Related: `config` Display current configuration and possibly change values (M)
`masvt` Type of variable temperature system (P)

vtwait Variable temperature wait time (P)

Applicability: Systems with a variable temperature (VT) module.

Description: Sets a time for establishing temperature regulation. If temperature interlock `tin` is set and regulation is not established after the time set by `vtwait`, VNMR displays the message “VT FAILURE” and aborts the experiment.

Values: Number, in seconds, A typical value is 180 seconds.

See also: *User Guide: Liquids NMR*

Related: `pad` Preacquisition delay (P)
`tin` Temperature interlock (P)

vxr_unix Convert VXR-style text files to UNIX format (M,U)

Syntax: (From VNMR) `vxr_unix(VXR_file<,UNIX_file>)`
(From UNIX) `vxr_unix VXR_file UNIX_file`

Description: Converts a VXR-style text file (from a Gemini, VXR, or XL system) to the UNIX format.

Arguments: `VXR_file` is the name of the input file, which must be a text file.

`UNIX_file` is the name of the output file after conversion. The names of the input and output files must be different.

Examples: (From VNMR) `vxr_unix('oldtextfile','newtextfile')`
(From UNIX) `vxr_unix oldtextfile newtextfile`

See also: *Getting Started*

Related: `convert` Convert data set from a VXR-style system (C,U)
`decomp` Decompose a VXR-style directory (C)
`unix_vxr` Convert UNIX text files to VXR-style format (M,U)

`vxrprint` Script for interface between VNMR and UNIX printing (obsolete)

Description: The `vxrprint` macro is no longer in VNMR. It was replaced by `vnmrprint`.

Related: `vnmrprint` Print text files (U)

W

- w** **Who is using system (C)**
- Syntax: `w`
- Description: Displays information about users currently on the system. It functions like the UNIX command of the same name.
- See also: *VNMR User Programming*
-
- walkup** **Walkup automation (M)**
- Syntax: `walkup`
- Description: Enables using sample changers for continuous “walk-up” operation. To use this macro, enter `walkup`, provide an automation directory name, fill in the Sample Entry Form window (`enter` program) that appears, and submit the experiment. The macro creates a new automation directory each day with the name `auto_dd.mm.yy`, where `dd` is the day of the month, `mm` is the month, and `yy` is the year (e.g., `auto_070497`). The automation directory is saved in a directory specified by the global parameter `globalauto`. `walkup` creates the directory `globalauto` and the parameter `globalauto`, and then sets the `globalauto` parameter.
- See also: *User Guide: Liquids NMR*
- Related: `enter` Enter sample information for automation run (M,U)
`globalauto` Automation directory name (P)
-
- waltz** **WALTZ decoupling present (P)**
- Description: Sets whether system is equipped for WALTZ decoupling. The value is changed by normal parameter entry rather than using the CONFIG window.
- Values: `'n'` sets WALTZ decoupling not present.
`'y'` sets WALTZ decoupling present.
- See also: *VNMR and Solaris Software Installation*
-
- wbs** **Specify action when bs transients accumulate (C)**
- Syntax: `wbs(string)`
- Description: Specifies what action to take when `bs` transients accumulate. The *command* `wbs` sets the corresponding *parameter* `wbs`. Using the command, rather than setting the parameter value explicitly, notifies the acquisition process that the associated parameter value has changed. Thus, the desired operation can be effected even if the experiment has already started.
- Arguments: `string` is a string argument containing the command or macro to be executed when this event happens. The string must be enclosed in single quotes. If single quotes are required *within* the text string, place a backslash character before each of the interior single quotes (`\'`). Maximum length of the string is 256 characters. To turn off `wbs` processing, enter `wbs('')`, where the argument is two single quotes with no space between.

Examples: `wbs('dg wft ')`
`wbs('mf(3) ')`
`wbs(' ')`

See also: *Getting Started*

Related: `bs` Block size (P)
`makefid` Make a FID element using numeric text input (C)
`phfid` Zero-order phasing constant for np FID (P)
`wbs` When block size (P)
`werr` Specify action when error occurs (C)
`wexp` Specify action when experiment completes (C)
`wnt` Specify action when nt transients accumulate (C)

wbs When block size (P)

Description: Invokes an action to occur automatically after each `bs` block of transients is completed. For example, `wbs= ' wft '` results in an automatic weighting and Fourier transformation after each `bs` transients. To specify no `wbs` processing, set `wbs` to the null string. If the acquisition has already started, the `wbs` *command* must be used to change this parameter.

Values: Command, macro, or null string (`wbs= ' '` , where the value is given by two single quotes with no space between them).

See also: *Getting Started; User Guide: Liquids NMR*

Related: `bs` Block size (P)
`wbs` Specify action when bs transients accumulate (C)

wc Width of chart (P)

Description: Specifies the width of the chart (plotting or printing area).

Values: 5 to `wcmax`, in mm.

See also: *Getting Started; User Guide: Liquids NMR*

Related: `wc2` Width of chart in second direction (P)
`wcmax` Maximum width of chart (P)

wc2 Width of chart in second direction (P)

Description: Specifies width of chart (plotting or printing area) along the second axis (or y axis) of a 2D contour plot or 2D “stacked display.” For plots made in the `cutoff` mode, `wc2` specifies the width of the plotted area along the y-axis.

Values: Width, in mm.

See also: *User Guide: Liquids NMR*

Related: `cutoff` Data truncation limit (P)
`ho` Horizontal offset (P)
`sc2` Start of chart in second direction (P)
`wcmax` Maximum width of chart (P)
`wc2max` Maximum width of chart in second direction (P)

wcmax Maximum width of chart (P)

Description: Specifies the maximum width of a chart (plotting or printing area). Set when plotter or printer is installed.

Values: Width, in mm.

See also: *Getting Started*

Related: `wc` Width of chart (P)
`wc2` Width of chart in second direction (P)

wc2max Maximum width of chart in second direction (P)

Description: Specifies the maximum width of a chart (plotting or printing area) in the second direction (y-axis). Set when the plotter or printer is installed.

Values: Width, in mm.

See also: *Getting Started*

Related: `wc2` Width of chart in second direction (P)
`wcmax` Maximum width of chart (P)

werr Specify action when error occurs (C)

Syntax: `werr(string)`

Description: Specifies what action to take if an error occurs during acquisition. The *command* `werr` sets the corresponding *parameter* `werr`. Using the *command*, rather than setting the parameter value explicitly, notifies the acquisition process that the associated parameter value has changed. Thus, the desired operation can be effected even if the experiment has already started.

Arguments: `string` is a string argument containing the command or macro to be executed when this event happens. The string must be enclosed in single quotes. If single quotes are required *within* the text string, place a backslash character before each of the interior single quotes (`\'`). Maximum length of the string is 256 characters. To turn off `werr` processing, enter `werr('')`, where the argument is two single quotes with no space between them.

Examples: `werr('react')`
`werr('')`

See also: *Getting Started*

Related: `wbs` Specify action when bs transients accumulate (C)
`werr` When error (P)
`wexp` Specify action when experiment completes (C)
`wnt` Specify action when nt transients accumulate (C)

werr When error (P)

Description: Specifies a macro (e.g., `werr='react'`) that will take appropriate action when an error occurs during acquisition. To specify no `werr` processing, set `werr` to the null string. If the acquisition has already been started, the *werr* *command* must be used to change the *werr* *parameter*. Arrayed parameter `acqstatus` provides the error code to `werr` in `acqstatus[1]` and `acqstatus[2]`. For a list of error codes, refer to the description of `acqstatus` or view the file `acq_errors` in directory `/vnmr/manual`.

Values: Macro or null string (`werr=''`, where the value is given by two single quotes with no space between them).

See also: *Getting Started; User Guide: Liquids NMR*

Related: `acqstatus` Acquisition status (P)
`react` Recover from error conditions during `werr` processing (M)
`werr` Specify action when error occurs (C)

- wet1d** **Set up parameters for a WET1D pulse sequence (M)**
 Applicability: Systems with LC-NMR accessory.
 Syntax: `wet1d`
 Description: Sets up for a WET1D LC-NMR experiment.
 See also: *User Guide: Liquids NMR*
- wetdqcosy** **Set up parameters for a WETDQCOSY pulse sequence (M)**
 Applicability: Systems with LC-NMR accessory.
 Syntax: `wetdqcosy`
 Description: Sets up for a WETDQCOSY LC-NMR experiment.
 See also: *User Guide: Liquids NMR*
- wetgcosy** **Set up parameters for a WETGCOSY pulse sequence (M)**
 Applicability: Systems with LC-NMR accessory.
 Syntax: `wetgcosy`
 Description: Sets up for a WETGCOSY LC-NMR experiment.
 See also: *User Guide: Liquids NMR*
- wetghmqcps** **Set up parameters for a WETGHMQCPS pulse sequence (M)**
 Applicability: Systems with LC-NMR accessory.
 Syntax: `wetghmqcps`
 Description: Sets up for a WETHMQCPS LC-NMR experiment.
 See also: *User Guide: Liquids NMR*
- wetghsqc** **Set up parameters for a WETGHSQC pulse sequence (M)**
 Applicability: Systems with LC-NMR accessory.
 Syntax: `wetghsqc('nucleus')`
 Description: Sets up for a WETGHSQC LC-NMR experiment.
 See also: *User Guide: Liquids NMR*
- wetgmqcosy** **Set up parameters for a WETGHSQC pulse sequence (M)**
 Applicability: Systems with LC-NMR accessory.
 Syntax: `wetgmqcosy`
 Description: Sets up for a WETGMQCOSY LC-NMR experiment.
 See also: *User Guide: Liquids NMR*
- wetnoesy** **Set up parameters for a WETNOESY pulse sequence (M)**
 Applicability: Systems with LC-NMR accessory.
 Syntax: `wetnoesy`
 Description: Sets up for a WETNOESY LC-NMR experiment.
 See also: *User Guide: Liquids NMR*.

wetpwxcal **Set up parameters for a WETPWXCAL pulse sequence (M)**

Applicability: Systems with LC-NMR accessory.

Syntax: `wetnoesy`

Description: Sets up for a WETPWXCAL LC-NMR pulse width calibration.

See also: *User Guide: Liquids NMR*

wettntocsy **Set up parameters for a WETTNTOCSY pulse sequence (M)**

Applicability: Systems with LC-NMR accessory.

Syntax: `wetnoesy`

Description: Sets up for a WETTNTOCSY LC-NMR experiment.

See also: *User Guide: Liquids NMR*

wetshape **Shape for pwwet pulses (P)**

Applicability: Systems with LC-NMR accessory.

Description: Sets the name of the shape used for pwwet pulses (e.g., `wetshape= 'wet'`).

See also: *User Guide: Liquids NMR*

wexp **Specify action when experiment completes (C)**

Syntax: `wexp(string)`

Description: Specifies what action to take when the experiment completes. The `wexp` command sets the corresponding parameter `wexp`. Using the command, rather than setting the parameter value explicitly, notifies the acquisition process that the associated parameter value has changed. Thus, the desired operation can be effected even if the experiment has already started.

Arguments: `string` is a string argument containing the command or macro to be executed when the experiment completes. The string must be enclosed in single quotes. If single quotes are required *within* the text string, place a backslash character before each of the interior single quotes (`\'`). Maximum length of the string is 256 characters. To turn off `wexp` processing, enter `wexp('')`, where argument is two single quotes with no space between them.

Examples: `wexp('wft(\ 'all\ ') calcT1')`
`wexp('')`

See also: *Getting Started*

Related:	<code>wbs</code>	Specify action when bs transients accumulate (C)
	<code>werr</code>	Specify action when error occurs (C)
	<code>wexp</code>	When experiment completes (P)
	<code>wnt</code>	Specify action when nt transients accumulate (C)

wexp **When experiment completes (P)**

Description: Invokes a single action to occur automatically after the experiment is finished, which can occur after a single FID or after a number of FIDs in a multi-FID experiment. To specify no `wexp` processing, set `wexp` to the null string. If the acquisition has already started, the `wexp` command must be used to change the `wexp` parameter. For `wexp` to execute after an experiment finishes, the `execute` the experiment with the `au` command.

wexp processing occurs after wnt processing in a single FID experiment, and both can be used. wexp also occurs after wnt during the last FID of a multi-FID experiment. Thus, wnt='wft(\ 'all\ ')' wexp='calcT1' and wexp='wft(\ 'all\ ') calcT1' transforms each FID in a T₁ experiment as it is performed, and when each of the FIDs has been collected, performs the calculation of the T₁ using a hypothetical macro command calcT1. Notice the use of the backslash to include a single quotation mark inside the string.

Values: Command, macro, or null string (wexp=' ', where the value is given by two single quotes with no space between them). If the command or macro uses a file name as an argument, specifying an absolute path is best. Be sure the path is valid and you have the appropriate write permission.

See also: *Getting Started; User Guide: Liquids NMR*

Related: **wexp** Specify action when experiment completes (C)
wnt When number of transients (P)
au Submit experiment to acquisition and process data (C)

wf Width of FID (P)

Description: Width of the FID display. This parameter can be entered in the usual way or interactively controlled by selecting the sf wf button during a FID display.

Values: 0 to the value of **at**, in seconds.

See also: *Getting Started*

Related: **at** Acquisition time (P)
dcon Display noninteractive color intensities map (C)
dconi Interactive 2D data display (C)
df Display a single FID (C)
sf Start of FID (P)
vf Vertical scale of FID (P)
wf1 Width of interferogram in 1st indirectly detected dimension (P)
wf2 Width of interferogram in 2nd indirectly detected dimension (P)

wf1 Width of interferogram in 1st indirectly detected dimension (P)

Description: Sets the width of the interferogram display in the first indirectly detected dimension.

Values: 0 to $(2 \times \text{ni}) / \text{sw1}$, in seconds.

See also: *User Guide: Liquids NMR*

Related: **ni** Number of increments in 1st indirectly detected dimension (P)
sf1 Start of interferogram in 1st indirectly detected dimension (P)
sw1 Spectral width in 1st indirectly detected dimension (P)
wf Width of FID (P)

wf2 Width of interferogram in 2nd indirectly detected dimension (P)

Description: Sets the width of the interferogram display in the second indirectly detected dimension.

Values: 0 to $(2 \times \text{ni2}) / \text{sw2}$, in seconds.

See also: *User Guide: Liquids NMR*

Related: **ni2** Number of increments in 2nd indirectly detected dimension (P)
sf2 Start of interferogram in 2nd indirectly detected dimension (P)

sw2 Spectral width in 2nd indirectly detected dimension (P)
wf Width of FID (P)

wfgtest **Waveform generator test (M)**

Applicability: Systems with a waveform generator.

Description: Retrieves a parameter set and pulse sequence, and compiles the sequence, in order to set up an experiment to test the waveform generators.

See also: *Waveform Generator Kit Installation*

wft **Weight and Fourier transform 1D data (C)**

Syntax: (1) `wft(<options>,<'nf'>,<start>,<finish>,<step>)`
 (2) `wft('inverse',exp_number,expansion_factor)`

Description: Performs a Fourier transform on one or more 1D FIDs with weighting applied to the FID. The command executes a left-shift, zero-order phase rotation, and a frequency shift according to the parameters **lsfid**, **phfid**, and **lsfrq**, respectively, on the time-domain data prior to the weighting and Fourier transformation. The type of Fourier transformation to be performed is determined by **proc**. **wft** uses the same arguments as the command **ft**, and except for weighting, it functions the same as the **ft** command.

Alternate: Weight, Transform button on 1D Data Processing menu.

See also: *Getting Started; User Guide: Liquids NMR*

Related: **ft** Fourier transform 1D data (C)
lsfid Number of points to left-shift np FID (P)
lsfrq Frequency shift of the fn spectrum in Hz (P)
phfid Zero-order phasing constant for np FID (P)
proc Type of processing on np FID (P)

wft1d **Weight and Fourier transform f₂ for 2D data (C)**

Syntax: (1) `wft1d(element_number)`
 (2) `wft1d(<options>,<coefficients>)`

Description: Performs the first Fourier transformation along the dimension defined by **sw**, with weighting and matrix transposition. This allows the display of t₁ interferograms with the **dcon** and **dconi** commands.

Except for weighting, **wft1d** functions the same as the **ft1d** command. See the description of **ft1d** for further information.

Arguments: Same as the arguments to **ft1d**. See the **ft1d** command for details.

See also: *User Guide: Liquids NMR*

Related: **dcon** Display noninteractive color intensity map (C)
dconi Interactive 2D data display (C)
ft1d Fourier transform along f₂ dimension (C)
sw Spectral width in directly detected dimension (P)

wft1da **Weight and Fourier transform phase-sensitive data (M)**

Syntax: `wft1da(<options>)`

Description: Processes 2D FID data as well as 2D planes at particular t₁ or t₂ times from a 3D data set for a pure absorptive display.

`wft1da` differs from `ft1da` only in that weighting of the time-domain data is performed prior to the Fourier transform. See the description of `ft1da` for further information.

Arguments: Same as arguments to `ft2da`. See the `ft2da` command for details.

See also: *User Guide: Liquids NMR*

Related:	<code>ft1da</code>	Fourier transform phase-sensitive data (M)
	<code>ft2da</code>	Fourier transform phase-sensitive data (M)
	<code>wft2da</code>	Weight and Fourier transform phase-sensitive data (M)

wft1dac **Combine arrayed 2D FID matrices (M)**

Syntax: `wft1dac(<mult1>, <mult2>, . . . <multn>)`

Description: Allows the ready combination of 2D FID matrices within the framework of the 2D Fourier transform program. Weighting is performed. This command requires that the data be acquired either without f_1 quadrature or with f_1 quadrature using the TPPI method. `wft1dac` is used with TOCSY (with multiple mixing times).

Arguments: `mult1`, `mult2`, . . . , `multn` are multiplicative coefficients. The n th argument is a real number and specifies the multiplicative coefficient for the n th 2D FID matrix.

See also: *User Guide: Liquids NMR*

Related:	<code>ft1dac</code>	Combine arrayed 2D FID matrices (M)
	<code>tocsy</code>	Set up parameters for TOCSY pulse sequence (M)
	<code>wft2dac</code>	Combine arrayed 2D FID matrices (M)

wft2d **Weight and Fourier transform 2D data (C)**

Syntax: `wft2d(<options>, >coefficients)`

Description: Performs a complete 2D transformation with weighting after 2D data has been acquired. If the first Fourier transformation has already been done using `ft1d`, `wft1d`, `ft1da`, or `wft1da`, then the `wft2d` command performs only the second transform.

For arrayed 2D experiments, a single array element can be transformed and weighted using the array element number as an argument. Interferograms can be constructed explicitly using the following coefficient table:

`wft2d(rr1, ir1, rr2, ir2, . . . ri1, ii1, ri2, ii2, . . .)`.

`wft2d('ptype', . . .)` transforms P-type spectra, and

`wft2d('ntype', . . .)` transforms N-type spectra. The default is N-type.

`wft2d` also *completes* a 2D transform that has been started with `wft1d` (or related commands such as `wft1da`). The first transform will not be done again if it has already been performed. For phase-sensitive 2D experiments, the coefficients must be applied as part of the first transform (e.g., with `wft1da`) since the interferograms are formed at that stage. These coefficients need not be repeated when invoking the subsequent transform: a simple `wft2d` or `ft2d` can suffice.

See the `ft2d` command description for further information.

Arguments: Same as the arguments to `ft2d`. See the `ft2d` command for details.

Examples: `wft2d(1, 0, 0, 0)`
`wft2d(2)`
`wft2d(1, 0, 1, 0, 0, 1, 0, 1)`
`wft2d(.67, 0, .33, 0, 0, .67, 0, .33)`

See also: *User Guide: Liquids NMR*

Related:	<code>dconi</code>	Interactive 2D data display (C)
	<code>ft1d</code>	Fourier transform along f_2 dimension (C)
	<code>ft1da</code>	Fourier transform “halfway” for pure absorption 2D data (M)
	<code>ft2d</code>	Fourier transform 2D data (C)
	<code>wft1d</code>	Weight and Fourier transform f_2 for 2D data (C)
	<code>wft1da</code>	Weight and FT “halfway” for pure absorption 2D data (M)
	<code>wft2da</code>	Weight and transform for pure absorption 2D data (M)

wft2da **Weight and Fourier transform phase-sensitive data (M)**

Syntax: `wft2da<(options)>`

Description: Processes 2D FID data, as well as 2D planes at particular t_1 or t_2 times, from a 3D data set for a pure absorptive display.

`wft2da` differs from `ft2da` only in that weighting of the time-domain data is performed prior to the Fourier transform. See the description of `ft2da` for further information.

Arguments: Same as used with `ft2da`. See the `ft2da` command for details.

See also: *User Guide: Liquids NMR*

Related:	<code>ft1da</code>	Fourier transform phase-sensitive data (M)
	<code>ft2da</code>	Fourier transform phase-sensitive data (M)
	<code>wft1da</code>	Weight and Fourier transform phase-sensitive data (M)

wft2dac **Combine arrayed 2D FID matrices (M)**

Syntax: `wft2dac<(<mult1><,mult2>,...<,multn>)>`

Description: Allows the ready combination of 2D FID matrices within the framework of the 2D Fourier transform program. Weighting is performed. This command requires that the data be acquired either without f_1 quadrature or with f_1 quadrature using the TPPI method. `wft2dac` is used with TOCSY (with multiple mixing times).

Arguments: `mult1, mult2, ..., multn` are multiplicative coefficients. The n th argument is a real number and specifies the multiplicative coefficient for the n th 2D FID matrix.

See also: *User Guide: Liquids NMR*

Related:	<code>ft1dac</code>	Combine arrayed 2D FID matrices (M)
	<code>ft2dac</code>	Combine arrayed 2D FID matrices (M)
	<code>tocsy</code>	Set up parameters for TOCSY pulse sequence (M)
	<code>wft1dac</code>	Combine arrayed 2D FID matrices (M)

wftt3 **Process f_3 dimension during 3D acquisition (M)**

Syntax: `wftt3`

Description: Allows f_3 processing of 3D data to be performed concurrently with data acquisition. To invoke this function, set `wnt = 'wftt3'` and use `au` to start the acquisition of the 3D data. When `wftt3` detects that all the FIDs comprising a (t_1, t_2) block have been acquired, it starts up the `ft3d` program in background to process that block of FIDs in f_3 .

The 3D processing information file, created by entering `set3dproc` within VNMR, does not need to contain valid f_1 and f_2 processing information but only valid f_3 processing information. Once the f_3 processing is complete, a new 3D

information file can be created for the f_1 - f_2 processing stages that contains valid f_1 and f_2 processing information.

The non-standard VNMR string parameter `path3d` can be used to specify the directory into which the f_3 processed 3D data is to be stored. Normally, `path3d` is absent in the parameter set. If this is the case or if `path3d= ' '`, the f_3 -processed 3D data is stored in the directory `curexp/datadir`. `path3d` can be created by entering `create('path3d','string')` `setgroup('path3d','display')`.

See also: *User Guide: Liquids NMR*

Related:	<code>au</code>	Submit experiment to acquisition and process data (C)
	<code>create</code>	Create new parameter in a parameter tree (C)
	<code>ft3d</code>	Perform a 3D Fourier transform (M,U)
	<code>getplane</code>	Extract planes from a 3D spectral data set (M)
	<code>path3d</code>	Path to currently displayed 2D planes from a 3D data set (P)
	<code>select</code>	Select a spectrum or 2D plane without displaying it (C)
	<code>set3dproc</code>	Set 3D processing (C)
	<code>setgroup</code>	Set group of a parameter in a tree (C)
	<code>wnt</code>	When number of transients (P)

which **Display which VNMR command or macro is used (M)**

Syntax: `which(name)`

Description: Searches VNMR libraries and then displays on line 3 which VNMR command or macro with the given name will be executed. For macros, `which` displays the type of macro (user, local, application, or Varian) and the path to the library.

Arguments: `name` is the name of a command or macro.

Examples: `which('wft')`

See also: *VNMR User Programming*

Related:	<code>exists</code>	Determin if a parameter, file, or macro exists (C)
	<code>hidecommand</code>	Execute macro instead of command with same name (M)

wnt **Specify action when nt transients accumulate (C)**

Syntax: `wnt(string)`

Description: Specifies what action to take when `nt` transients accumulate. The `wnt` *command* sets the corresponding *parameter* `wnt`. Using the command, rather than setting the parameter value explicitly, notifies the acquisition process that the associated parameter value has changed. Thus, the desired operation can be effected even if the experiment has already started.

Arguments: `string` is a string argument containing the command or macro to be executed when this event happens. The string must be enclosed in single quotes. If single quotes are required within the text string, place a backslash character before each of the interior single quotes (`\'`). Maximum length of the string is 256 characters. To turn off `wnt` processing, enter `wnt('')`, where the argument is two single quotes with no space between them.

Examples: `wnt('wft(\'all\')`
`wnt('')`

See also: *Getting Started*

Related:	<code>nt</code>	Number of transients (P)
	<code>wbs</code>	Specify action when <code>bs</code> transients accumulate (C)
	<code>werr</code>	Specify action when error occurs (C)

wexp When experiment completes (P)
wnt When number of transients (P)

wnt When number of transients (P)

Description: Invokes a single action to occur automatically after the FID is finished (`ct=nt`) or after each FID in a multi-FID experiment involving an arrayed parameter. The most common processing to occur after an FID is an automatic weighting and Fourier transformation (i.e., `wnt= 'wft'`); however, this is normally not needed because the command `ga` is the exact equivalent of `wnt= 'wft (\ 'acq\ ') ' au` (i.e., `ga` sets the `wnt` action automatically). To specify no `wnt` processing, set `wnt` to the null string. If the acquisition has already been started, the `wnt` command must be used to change this parameter.

Values: Command, macro, or null string (`wnt= ' '`, where the value is given by two single quotes with no space between them).

See also: *Getting Started; User Guide: Liquids NMR*

Related: **nt** Number of transients (P)
wnt Specify action when `nt` transients accumulate (C)

wp Width of plot in directly detected dimension (P)

Description: Sets the width of the displayed or plotted region of the spectrum.

Values: Always stored in Hz, but can be entered in ppm by using the `p` suffix (e.g., `wp=6p` sets the width of plot to 6 ppm).

See also: *Getting Started; User Guide: Liquids NMR*

Related: **wp1** Width of plot in 1st indirectly detected dimension (P)
wp2 Width of plot in 2nd indirectly detected dimension (P)

wp1 Width of plot in 1st indirectly detected dimension (P)

Description: Analogous to the `wp` parameter except that `wp1` applies to the first indirectly detected dimension of a multidimensional data set.

See also: *User Guide: Liquids NMR*

Related: **wp** Width of plot in directly detected dimension (P)
wp2 Width of plot in 2nd indirectly detected dimension (P)

wp2 Width of plot in 2nd indirectly detected dimension (P)

Description: Analogous to the `wp` parameter except that `wp2` applies to the second indirectly detected dimension of a multidimensional data set.

See also: *User Guide: Liquids NMR*

Related: **wp** Width of plot in directly detected dimension (P)
wp1 Width of plot in 1st indirectly detected dimension (P)

write Write formatted text to a device (C)

Syntax: (1) `write('keywords' ><, color | pen ><, 'reverse' >, x, y <, template >) <:height >`
(2) `write('alpha' | 'printer' | 'line3' | 'error', template)`
(3) `write('reset' | 'file' | 'fileline', file <, template >)`

Description: Writes text to a graphics screen or plotter in a given format (syntax 1), writes formatted text to another device (syntax 2), clears a file (syntax 3), or writes to

a file (syntax 3). The input to the command comes from arguments in `template`, which can be parameters such as `n1` or `pw`.

- Arguments: 'keywords' identify the output device ('graphics' | 'plotter') and the drawing mode ('xor' | 'normal' | 'newovly' | 'ovly' | 'ovlyC').
- 'graphics' | 'plotter' is a keyword selecting the output device. The default is 'plotter'. The output selected is passed to subsequent `pen`, `move`, or `draw` commands and remains active until a different mode is specified.
 - 'xor', 'normal' is a keyword for the drawing mode when using the 'graphics' output device. The default is 'normal'. In the 'xor' mode, if a line is drawn such that one or more points of the line are in common with a previous 'xor' line, the common points are erased. In the normal mode, the common points remain. The mode selected is passed to subsequent `pen`, `move`, and `draw` commands and remains active until a different mode is specified.
 - 'newovly', 'ovly', and 'ovlyC' are keywords that specify an interactive drawing capability that is slightly slower than the 'xor' mode but more consistent in color. 'newovly' clears any previous draws, boxes, and writes made with the 'ovly' modes and draws the figure. 'ovly' draws without clearing so that multi-segment figures can be created. 'ovlyC' clears without drawing.

`color` is the color of the text on a color display: 'red', 'yellow', 'green', 'cyan', 'blue', 'magenta', and 'white'. The default is 'yellow'.

`pen` is the plotter pen: 'pen1', 'pen2', etc.

'reverse' is a keyword specifying a sideways orientation of the output.

`x` and `y` are coordinates on the screen or plotter, in mm.

`template` is a string of formatting characters along with arguments to those characters. The format is the same as used with the UNIX `printf` command (for details, see any basic UNIX manual or enter `man printf` in UNIX). For example, '`pw = %12.5f`' is a template to format the parameter `pw` as fixed point with a field width of 12 spaces and 5 decimal places. The following format characters are implemented:

character	%c
integer	%d
hexadecimal	%h
exponential:	%e
fixed point	%f
exponential/fixed point	%g
octal	%o
string	%s
write a % character	use <code>write(...'%s', '%')</code>

`height` returns the height of the characters on the screen or plotter. This is useful for positioning multiple-line displays. See the source code of the macro `dtext` in the `maclib` directory for an example of usage.

'alpha' is a keyword to write text to the alphanumeric screen.

'printer' is a keyword to print text on the printer

'line3' is a keyword to write text as a message on line 3.

'error' is a keyword to write text as an error on line 3 and sound a beep.

'reset' is a keyword to clear the file specified.

'file' is a keyword to append data to the file specified. Existing data in the file is not overwritten. By writing repeated 'file' calls, a formatted data file can be created (see the fifth example below). Each write command automatically appends a carriage return (linefeed) to the end of the string defined by the template argument. To append data without the automatic linefeed, use the 'fileline' keyword instead of 'file'.

'fileline' is a keyword to append data to the file specified, the same as using the 'file' keyword, but without automatically appending a carriage return (linefeed) to the end of the data. Any linefeeds desired must be explicitly defined (using \n) by the template argument (see the sixth example below).

file is the name of the file used with the 'reset', 'file', and 'fileline' keywords.

Examples:

```
write('graphics',100,100):$ys
write('plotter',20,180, 'pw = %12.5f',pw)
write('line3', 'Too many arguments')
write('reset','templ')
write('file','templ','%10f %10.1f',n1,pw)
write('fileline','templ','\nEnd of data\n\n')
```

See also: *VNMR User Programming*

Related: [dtext](#) Display a text file in the graphics window (M)

writefid Write numeric text file using a FID element (C)

Syntax: `writefid(file<,element_number>)`

Description: Writes a text file using data from the selected FID element. The program writes two values per line—the first is the value from the X (or real) channel and the second is the value from the Y (or imaginary) channel. `writefid` writes the raw FID data (i.e., FID data processing based on the parameters [phfid](#), [lsfid](#), and [lsfrq](#) does not occur).

Arguments: `file` is the name of a text file to store the data.

`element_number` is an integer larger than 0 for the number of a FID element. The default is 1.

See also: *Getting Started, VNMR User Programming*

Related: [lsfid](#) Number of complex points to left-shift np FID (P)
[lsfrq](#) Frequency shift of fn spectrum in Hz (P)
[makefid](#) Make a FID element using numeric text input (C)
[phfid](#) Zero-order phasing constant for np FID (P)

wsram Send hardware configuration to acquisition console (C)

Applicability: UNITY*INOVA* and UNITY*plus* systems.

Syntax: `wsram<:$success>`

Description: Sends new hardware configuration information to the acquisition console when [config](#) is used (e.g., to set [lockfreq](#)). `wsram` (write to static RAM) is not normally entered directly by the user.

Arguments: `success` returns 1 if `wsram` is successful, or 0 otherwise.

See also: *VNMR and Solaris Software Installation*.

Related: `config` Display current configuration and possibly change it (M)
`lockfreq` Lock frequency (P)

wshim Conditions when shimming is performed (P)

Description: Specifies when automatic shimming is to be used, according to the method specified by the parameter `method`.

Values: 'n' sets that no automatic shimming is performed. Even with `wshim` set to this value, the shimming procedure specified by the parameter `method` can be activated by using the `shim` command.

'e' or 'exp' sets that automatic shimming is done before data acquisition.

's' or 'samp' sets that automatic shimming is done only at the beginning of the first experiment, following the change of a sample using the automatic sample changer.

'g' sets that automatic shimming using gradient shimming is done only at the beginning of the first experiment, following the change of a sample using the automatic sample changer. The parameter `method` is ignored. This option is only available in automation and is not used with the `go`, `ga`, or `au` commands.

'f' or 'fid' set automatic shimming is done prior to the data collection of each new array member in a multi-FID experiment (this option not implemented on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems).

'fn', where *n* is an integer, sets shimming is done prior to data collection of every *n*th FID (e.g., `wshim='f16'` shims prior to acquiring FIDs 1, 17, 33, etc.). This method is only relevant to arrayed or 2D experiments (this option not implemented on *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* systems).

See also: *Getting Started*

Related: `gf` Prepare parameters for FID/spectrum display in `acqi` (M)
`method` Autoshim method (P)

wtfile User-defined weighting in directly detected dimension (P)

Description: Set to name of the file containing the user-written weighting function along the directly detected dimension. This dimension is referred to as the f_2 dimension in 2D data sets, the f_3 dimension in 3D data sets, etc. The shellscript `wtgen` is used to compile the user-written weighting module into an executable program. The source file is stored in the directory `vnmruser+'wtlib'` with a `.c` file extension. The executable file is in the same directory and has the same name as the source file but has no file extension.

Values: `file` is the name of the executable weighting function or the name of the weighting function text file.

' ' (two single quotes with no space in between) indicates `wtfile` is inactive and VNMR should not look for a user-written weighting function.

See also: *Getting Started; VNMR User Programming*

Related: `wtfile1` User-defined weighting in 1st indirectly detected dimension (P)
`wtfile2` User-defined weighting in 2nd indirectly detected dimension (P)
`wtgen` Compile user-written weighting functions (C,U)

- wtfile1** **User-defined weighting in 1st indirectly detected dimension (P)**
- Description: Set to the name of the file containing the user-written weighting function for the first indirectly detected dimension. This dimension is often referred to as the f_1 dimension of a multidimensional data set. Otherwise, `wtfile1` is analogous to `wtfile`.
- See also: *User Guide: Liquids NMR; VNMR User Programming*
- Related: `wtfile` User-defined weighting in directly detected dimension (P)
`wtfile2` User-defined weighting in 2nd indirectly detected dimension (P)
- wtfile2** **User-defined weighting in 2nd indirectly detected dimension (P)**
- Description: Set to the name of the file containing the user-written weighting function along the second indirectly detected dimension. This dimension is often referred to as the f_2 dimension of a multidimensional data set. `wtfile2` can be set with `wti` on the 2D interferogram data. Otherwise, `wtfile2` is analogous to `wtfile`.
- See also: *User Guide: Liquids NMR; VNMR User Programming*
- Related: `wtfile` User-defined weighting in directly detected dimension (P)
`wtfile1` User-defined weighting in 1st indirectly detected dimension (P)
`wti` Interactive weighting (C)
- wtgen** **Compile user-written weighting functions (M,U)**
- Syntax: (From VNMR) `wtgen(file<.c>)`
(From UNIX) `wtgen file<.c>`
- Description: Allows compilation of a user-written weighting function that subsequently can be executed from within VNMR. `wtgen` performs the following functions:
- Checks for the existence of the `/vnmr/bin` directory and aborts if the directory is not found.
 - Checks for files `usrwt.o` and `weight.h` in the `/vnmr/bin` directory and aborts if either of these two files cannot be found there.
 - Checks for the existence of the user's directory and creates this directory if it does not already exist.
 - Establishes in the `wtlib` directory soft links to `usrwt.o` and `weight.h` in the `/vnmr/bin` directory.
 - Compiles the user-written weighting function, which is stored in the `wtlib` directory, link loads it with `usrwt.o`, and places the executable program in the same directory; any compilation and/or link loading errors are placed in the file `errmsg` in `wtlib`.
 - Removes the soft links to `usrwt.o` and `weight.h` in the `/vnmr/bin` directory.
- The name of the executable program is the same as that for the source file without a file extension (e.g., `testwt.c` is the source file for the executable file `testwt`).
- Examples: (From VNMR) `wtgen('testwt')`
(From UNIX) `wtgen testwt.c`
- See also: *VNMR User Programming*
- Related: `wtfile` User-defined weighting for t_2 (P)
`wtfile1` User-defined weighting for t_1 (P)
`wtfile2` User-defined weighting in `ni2` dimension (P)

wti **Interactive weighting (C)**

Syntax: `wti<(element_number)>`

Description: Allows weighting parameters to be set interactively for both t_2 FIDs and t_1 interferograms. *wti* responds appropriately to `phfid` and `lsfid` for t_2 FIDs and to `phfid1` and `lsfid1` for t_1 interferograms. The following parameters can be interactively weighted:

- `awc`, `awc1`, and `awc2` set the additive weighting constant; added in to the weighting function after the `lb` and `sb` (or `sbs`) contributions but before the `gf` (or `gfs`) contributions.
- `gf`, `gf1`, and `gf2` set the Gaussian apodization constant, in seconds.
- `gfs`, `gfs1`, and `gfs2` set the Gaussian function shift, in seconds; shifts the origin of the Gaussian function; active only if `gf` (or `gf1`) is active.
- `lb`, `lb1`, and `lb2` set the line broadening factor, in Hz; a positive value gives sensitivity enhancement; a negative value gives resolution enhancement.
- `sb`, `sb1`, and `sb2` set the sinebell time period, in seconds; a negative value give a sine squared bell.
- `sbs`, `sbs1`, and `sbs2` set the sinebell shift, in seconds; shifts the origin of the sine bell; active only if `sb` (or `sb1`) is active.

These parameters can be typed in or changed with the left mouse button in the proper field. The right mouse button turns off the spectrum for a faster response to changes in the weighting function.

Arguments: `element_number` specifies which FID element or interferogram trace is to be used in adjusting the weighting parameters. The default is the currently active element or trace.

Examples: `wti`
`wti(3)`

Alternate: Adj Weighting button in the 1D Data Processing Menu, or Adj Weighting button in the 2D Data Processing Menu, or Adj Weighting button in the 2D Interferogram Processing Menu.

See also: *Getting Started; User Guide: Liquids NMR*

Related:	<code>lsfid</code>	Number of complex points to left-shift <code>np</code> FID (P)
	<code>lsfid1</code>	Number of complex points to left-shift <code>ni</code> interferogram (P)
	<code>phfid</code>	Zero-order phasing constant for <code>np</code> FID (P)
	<code>phfid1</code>	Zero-order phasing constant for <code>ni</code> interferogram (P)
	<code>wtia</code>	Interactive weighting for 2D absorptive data (C)

wtia **Interactive weighting for 2D absorptive data (M)**

Syntax: `wtia<(element_number)>`

Description: Allows weighting parameters to be set interactively for both t_2 FIDs and t_1 interferograms in 2D absorptive data. Refer to the description of the `wti` command for further information.

Arguments: `element_number` specifies which FID element or interferogram trace is to be used in adjusting the weighting parameters. The default is the currently active trace.

See also: *User Guide: Liquids NMR*

Related:	<code>lsfid</code>	Number of complex points to left-shift <code>np</code> FID (P)
	<code>lsfid1</code>	Number of complex points to left-shift <code>ni</code> interferogram (P)

W

`phfid` Zero-order phasing constant for np FID (P)
`wti` Interactive weighting (C)

wysiwyg **Set plot display or full display (P)**

Description: Sets whether the window display is the same as the plot (“what you see is what you get,” or WYSIWYG) or is expanded to fill the window. This allows the user to scale the image to the full window, making it easier to view. This parameter is in the user’s global parameter file.

Values: 'y' makes the window picture size depend on the current plotter setting. Scaling the window does not change the ratio of the picture. This value is the default display condition.

'n' makes the window display expand, giving a full display.

See also: *Getting Started*

X

- x0** **X-zero position of HP pen plotter or Postscript device (P)**
- Applicability: Systems with a Hewlett-Packard pen plotter or a Postscript output device.
- Description: Adjusts the *x*-zero position on the chart. Use `hpa` to adjust `x0` (and `y0`) to place the numbers in a pleasing position when filled in on the blank lines. `x0` is part of `vnmr/sys/global` and hence common to all experiments.
- Values: Number, in mm.
- See also: *Getting Started*
- Related: `hpa` Plot parameters on special preprinted chart paper (C)
`y0` Y-zero position of HP plotter or Postscript device (P)
- x1** **X1 shim gradient (P)**
- Description: Holds current setting of the X1 radial shim gradient.
- Values: If `shimset` is 1, 2, 8, 10: -2048 to +2047, steps of 1, 0 is no current.
If `shimset` is 3 to 7, 9: -32768 to +32767, steps of 1, 0 is no current.
- See also: *Getting Started*
- Related: `shimset` Type of shim set (P)
- x2y2** **X2Y2 shim gradient (P)**
- Description: Holds current setting of the X2Y2 radial shim gradient.
- Values: If `shimset` is 1, 2, 8, 10: -2048 to +2047, steps of 1, 0 is no current.
If `shimset` is 3 to 7, 9: -32768 to +32767, steps of 1, 0 is no current.
- See also: *Getting Started*
- Related: `shimset` Type of shim set (P)
- x3** **X3 shim gradient (P)**
- Description: Holds current setting of the X3 radial shim gradient.
- Values: If `shimset` is 1, 2, 10: -2048 to +2047, steps of 1, 0 is no current.
If `shimset` is 3 to 7, 9: -32768 to +32767, steps of 1, 0 is no current.
- See also: *Getting Started*
- Related: `shimset` Type of shim set (P)
- x4** **X4 shim gradient (P)**
- Description: Holds current setting of the X4 radial shim gradient.
- Values: -32768 to +32767, steps of 1, 0 is no current.
- See also: *Getting Started*
- Related: `shimset` Type of shim set (P)
- xdiag** **Threshold for excluding diagonal peaks when peak picking (P)**
- Description: Used by the `ll2d` program to exclude diagonal peaks when peak picking.

To create the 2D peak picking parameters `xdiag` and `th2d` in the current experiment, enter `addpar('112d')`.

Values: Peaks within `xdiag` Hz of the diagonal will not be picked by `112d`. Setting `xdiag` to 0.0 will cause `112d` to pick all peaks, including diagonal peaks.

See also: *User Guide: Liquids NMR*

Related: `addpar` Add selected parameters to the current experiment (M)
`112d` Automatic and interactive 2D peak picking (C)
`th2d` Threshold for integrating peaks in 2D spectra (P)

xgate Load time counter (M)

Applicability: Systems with a solids module.

Syntax: `xgate(counts)`

Description: Loads the (12-bit) time counter on the pulse programmer with the specified number of counts and switches the counter to the external time base (the external trigger). On each trigger, the counter counts one unit down, and the next pulse sequence event starts when the count reaches zero. Often that time count will be just 1 (1.0, as the argument must be a floating point number). If the final pulse is to be performed after a longer delay, two options are available:

- Perform a normal delay, followed by the `xgate(1.0)` call.
- Calculate how many rotor cycles that delay would be (calculation is typically done based on a VNMR parameter `srate`) and then perform `xgate` with that calculated number of rotor triggers. Be aware that the only number of rotor cycles that can be counted this way is 4096, because the pulse programmer uses a 12-bit counter). At typical rotor speeds of 5 to 10 kHz, the “counted” delay is limited to 0.8 to 0.4 seconds.

Arguments: `counts` is the number of counts to load into the time counter. The value must be a floating point number.

Examples: `xgate(5.0)`

See also: *User Guide: Solid-State NMR; VNMR Pulse Sequences*

Related: `srate` Spinning rate for magic angle spinning (P)

xpol Cross-polarization (P)

Applicability: Systems with a solids module.

Description: Selects cross-polarization or direct polarization in solid-state NMR experiments such as XPOLAR and XPOLAR1.

Values: 'n' sets the experiment for direct polarization.
'y' sets the experiment for cross-polarization.

See also: *User Guide: Solid-State NMR*

Related: `xpolar` Set up parameters for XPOLAR pulse sequence (M)
`xpolar1` Set up parameters for XPOLAR1 pulse sequence (M)

xpolar Set up parameters for XPOLAR pulse sequence (M)

Applicability: UNITY systems with a solids module.

Syntax: `xpolar`

Description: Sets up a solid-state NMR cross-polarization experiment.

Alternate: XPOLAR button in the 1D Pulse Sequence Setup Secondary menu.

See also: *User Guide: Solid-State NMR*

Related:	<code>hsrotor</code>	Display rotor speed for solids operation (P)
	<code>rotorsync</code>	Rotor synchronization (P)
	<code>srate</code>	Spinning speed (P)
	<code>xpolar1</code>	Set up parameters for XPOLAR1 pulse sequence (M)

xpolar1 **Set up parameters for XPOLAR1 pulse sequence (M)**

Applicability: ^{UNITY}*INOVA* and *UNITYplus* systems with a solids module.

Syntax: `xpolar1`

Description: Sets up the solid-state NMR cross-polarization experiment XPOLAR using the parameters preferred for the ^{UNITY}*INOVA* and *UNITYplus*. Otherwise, `xpolar1` contains the same functionality as `xpolar`.

See also: *User Guide: Solid-State NMR*

Related:	<code>hsrotor</code>	Display rotor speed for solids operation (P)
	<code>rotorsync</code>	Rotor synchronization (P)
	<code>xpolar</code>	Set up parameters for XPOLAR pulse sequence (M)

xy **XY shim gradient (P)**

Description: Holds current setting of the XY radial shim gradient.

Values: If `shimset` is 1, 2, 8, 10: -2048 to +2047, steps of 1, 0 is no current.
If `shimset` is 3 to 7, 9: -32768 to +32767, steps of 1, 0 is no current.

See also: *Getting Started*

Related: `shimset` Type of shim set (P)

xz **XZ shim gradient (P)**

Description: Holds current setting of the XZ radial shim gradient.

Values: If `shimset` is 1, 2, 8, 10: -2048 to +2047, steps of 1, 0 is no current.
If `shimset` is 3 to 7, 9: -32768 to +32767, steps of 1, 0 is no current.

See also: *Getting Started*

Related: `shimset` Type of shim set (P)

xz2 **XZ2 shim gradient (P)**

Description: Holds current setting of XZ2 radial shim gradient.

Values: If `shimset` is 2, 8: -2048 to +2047, steps of 1, 0 is no current.
If `shimset` is 3 to 7, 9: -32768 to +32767, steps of 1, 0 is no current.

See also: *Getting Started*

Related: `shimset` Type of shim set (P)

Y

- y0** **Y-zero position of HP pen plotter or Postscript device (P)**
- Applicability: Systems with a Hewlett-Packard pen plotter or a Postscript output device.
- Description: Adjusts the y-zero position on the chart. Use `hpa` to adjust `y0` (and `x0`) to place numbers in a pleasing position when filled in on the blank lines. `y0` is part of `vnmr/sys/global`; therefore, it is common to all experiments.
- Values: Number, in mm.
- See also: *Getting Started*
- Related: `hpa` Plot parameters on special preprinted chart paper (C)
`x0` X-zero position of HP plotter or Postscript device (P)
-
- y1** **Y1 shim gradient (P)**
- Description: Holds current setting of the Y1 radial shim gradient.
- Values: If `shimset` is 1, 2, 8, 10: -2048 to +2047, steps of 1, 0 is no current.
If `shimset` is 3 to 7, 9: -32768 to +32767, steps of 1, 0 is no current.
- See also: *Getting Started*
- Related: `shimset` Type of shim set (P)
-
- y3** **Y3 shim gradient (P)**
- Description: Holds current setting of the Y3 radial shim gradient.
- Values: If `shimset` is 1, 2, 10: -2048 to +2047, steps of 1, 0 is no current.
If `shimset` is 3 to 7, 9: -32768 to +32767, steps of 1, 0 is no current.
- See also: *Getting Started*
- Related: `shimset` Type of shim set (P)
-
- y4** **Y4 shim gradient (P)**
- Description: Holds current setting of the Y4 radial shim gradient.
- Values: -32768 to +32767, steps of 1, 0 is no current.
- See also: *Getting Started*
- Related: `shimset` Type of shim set (P)
-
- yz** **YZ shim gradient (P)**
- Description: Holds current setting of the YZ radial shim gradient.
- Values: If `shimset` is 1, 2, 8, 10: -2048 to +2047, steps of 1, 0 is no current.
If `shimset` is 3 to 7, 9: -32768 to +32767, steps of 1, 0 is no current.
- See also: *Getting Started*
- Related: `shimset` Type of shim set (P)
-
- yz2** **YZ2 shim gradient (P)**
- Description: Holds current setting of the YZ2 radial shim gradient.

Values: If `shimset` is 2, 8: -2048 to +2047, steps of 1, 0 is no current.
If `shimset` is 3 to 7, 9: -32768 to +32767, steps of 1, 0 is no current.

See also: *Getting Started*

Related: `shimset` Type of shim set (P)

Z

z Add integral reset point at cursor position (C)

Syntax: `z<(reset1,reset2,...)>`

Description: Resets the integral to zero at the point marked by the displayed cursor. The command `cz` removes all such integral resets and it should generally be used before starting to enter a series of integral zeros (resets). The resets are stored as frequencies and do not change if `fn` is changed.

Arguments: `reset1, reset2, ...` are reset points entered, in either Hz or ppm. The default is the cursor position). Reset points can be entered in any order.

Examples: `z`
`z(7.5*sfrq,5*sfrq,2.5*sfrq,0.1*sfrq)`

See also: *Getting Started*

Related:	<code>cz</code>	Clear integral reset points (C)
	<code>dlni</code>	Display list of normalized integrals (C)
	<code>ds</code>	Display a spectrum (C)
	<code>fn</code>	Fourier number in directly detected dimension (P)
	<code>nli</code>	Find integral values (C)
	<code>nlni</code>	Find normalized integral values (C)

z0 Z0 field position (P)

Description: Holds current setting of the Z0 setting. The value of `z0` can be set by `su`. Starting with VNMR 6.1, and only on ^{UNITY}INOVA systems, `lockfreq` can be used to find the lock signal or resonance. To use the lock frequency, deactivate `z0` by typing the statement `z0='n'`. To activate `z0`, enter `z0='y'`.

Values: If `shimset` is 1, 2, 8, 10: -2048 to +2047, steps of 1, 0 is no current.
 If `shimset` is 3 to 7, 9: -32768 to +32767, steps of 1, 0 is no current.

See also: *Getting Started*

Related:	<code>lockfreq</code>	Lock frequency (P)
	<code>su</code>	Submit a setup experiment to acquisition (M)

z1 Z1 shim gradient (P)

Description: Holds current setting of the Z1 axial shim gradient.

Values: If `shimset` is 1, 2, 8, 10: -2048 to +2047, steps of 1, 0 is no current.
 If `shimset` is 3 to 7, 9: -32768 to +32767, steps of 1, 0 is no current.

See also: *Getting Started*

Related:	<code>shimset</code>	Type of shim set (P)
----------	----------------------	----------------------

z1c Z1C shim gradient (P)

Description: Holds current setting of the Z1C axial shim gradient.

Values: If `shimset` is 1, 2, 10: -2048 to +2047, steps of 1, 0 is no current.
 If `shimset` is 5 or 9: -32768 to +32767, steps of 1, 0 is no current.

See also: *Getting Started*

Related: `shimset` Type of shim set (P)

z2 Z2 shim gradient (P)

Description: Holds current setting of the Z2 axial shim gradient.

Values: If `shimset` is 1, 2, 8, 10: -2048 to +2047, steps of 1, 0 is no current.
If `shimset` is 3 to 7, 9: -32768 to +32767, steps of 1, 0 is no current.

See also: *Getting Started*

Related: `shimset` Type of shim set (P)

z2c Z2C shim gradient (P)

Description: Holds current setting of the Z2C axial shim gradient.

Values: If `shimset` is 1, 2, 10: -2048 to +2047, steps of 1, 0 is no current.
If `shimset` is 5 or 9: -32768 to +32767, steps of 1, 0 is no current.

See also: *Getting Started*

Related: `shimset` Type of shim set (P)

z2x2y2 Z2X2Y2 shim gradient (P)

Description: Holds current setting of the Z2X2Y2 radial shim gradient.

Values: -32768 to +32767, steps of 1, 0 is no current.

See also: *Getting Started*

z2x3 Z2X3 shim gradient (P)

Description: Holds current setting of the Z2X3 radial shim gradient.

Values: -32768 to +32767, steps of 1, 0 is no current.

See also: *Getting Started*

z2xy Z2XY shim gradient (P)

Description: Holds current setting of the Z2XY radial shim gradient.

Values: -32768 to +32767, steps of 1, 0 is no current.

See also: *Getting Started*

z2y3 Z2Y3 shim gradient (P)

Description: Holds current setting of the Z2Y3 radial shim gradient.

Values: -32768 to +32767, steps of 1, 0 is no current.

See also: *Getting Started*

z3 Z3 shim gradient (P)

Description: Holds current setting of the Z3 axial shim gradient.

Values: If `shimset` is 1, 2, 8, 10: -2048 to +2047, steps of 1, 0 is no current.
If `shimset` is 3 to 7, 9: -32768 to +32767, steps of 1, 0 is no current.

Z

See also: *Getting Started*

Related: `shimset` Type of shim set (P)

z3c Z3C shim gradient (P)

Description: Holds current setting of the Z3C radial shim gradient.

Values: -32768 to $+32767$, steps of 1, 0 is no current.

See also: *Getting Started*

z3x Z3X shim gradient (P)

Description: Holds current setting of the Z3X radial shim gradient.

Values: -32768 to $+32767$, steps of 1, 0 is no current.

See also: *Getting Started*

z3x2y2 Z3X2Y2 shim gradient (P)

Description: Holds current setting of the Z3X2Y2 radial shim gradient.

Values: -32768 to $+32767$, steps of 1, 0 is no current.

See also: *Getting Started*

z3x3 Z3X3 shim gradient (P)

Description: Holds current setting of the Z2X3 radial shim gradient.

Values: -32768 to $+32767$, steps of 1, 0 is no current.

See also: *Getting Started*

z3xy Z3XY shim gradient (P)

Description: Holds current setting of the Z3XY radial shim gradient.

Values: -32768 to $+32767$, steps of 1, 0 is no current.

See also: *Getting Started*

z3y Z3Y shim gradient (P)

Description: Holds current setting of the Z3Y radial shim gradient.

Values: -32768 to $+32767$, steps of 1, 0 is no current.

See also: *Getting Started*

z3y3 Z3Y3 shim gradient (P)

Description: Holds current setting of the Z3Y3 radial shim gradient.

Values: -32768 to $+32767$, steps of 1, 0 is no current.

See also: *Getting Started*

z4 Z4 shim gradient (P)

Description: Holds current setting of the Z4 shim gradient.

Values: If `shimset` is 1, 2, 8, 10: -2048 to $+2047$, steps of 1, 0 is no current.
If `shimset` is 3 to 7, 9: -32768 to $+32767$, steps of 1, 0 is no current.

See also: *Getting Started*

Related: `shimset` Type of shim set (P)

z4c Z4C shim gradient (P)

Description: Holds current setting of the Z4C shim gradient.

Values: -32768 to $+32767$, steps of 1, 0 is no current.

See also: *Getting Started*

z4x Z4X shim gradient (P)

Description: Holds current setting of the Z4X shim gradient.

Values: -32768 to $+32767$, steps of 1, 0 is no current.

See also: *Getting Started*

z4x2y2 Z4X2Y2 shim gradient (P)

Description: Holds current setting of the Z4X2Y2 radial shim gradient.

Values: -32768 to $+32767$, steps of 1, 0 is no current.

See also: *Getting Started*

z4xy Z4XY shim gradient (P)

Description: Holds current setting of the Z4XY radial shim gradient.

Values: -32768 to $+32767$, steps of 1, 0 is no current.

See also: *Getting Started*

z4y Z4Y shim gradient (P)

Description: Holds current setting of the Z4Y shim gradient.

Values: -32768 to $+32767$, steps of 1, 0 is no current.

See also: *Getting Started*

z5 Z5 shim gradient (P)

Description: Holds current setting of the Z5 axial shim gradient.

Values: If `shimset` is 2, 10: -2048 to $+2047$, steps of 1, 0 is no current.
If `shimset` is 3 to 7, 9: -32768 to $+32767$, steps of 1, 0 is no current.

See also: *Getting Started*

Related: `shimset` Type of shim set (P)

z5flag Z5 shimming present (obsolete)

Description: A configuration parameter no longer in VNMR.

Related: `shimset` Type of shim set (P)

z5x Z5X shim gradient (P)

Description: Holds current setting of the Z5X radial shim gradient.

Values: -32768 to $+32767$, steps of 1, 0 is no current.

Z

See also: *Getting Started*

z5y **Z5Y shim gradient (P)**

Description: Holds current setting of the Z5Y radial shim gradient.

Values: -32768 to $+32767$, steps of 1, 0 is no current.

See also: *Getting Started*

z6 **Z6 shim gradient (P)**

Description: Holds current setting of the Z6 axial shim gradient.

Values: -32768 to $+32767$, steps of 1, 0 is no current.

See also: *Getting Started*

z7 **Z7 shim gradient (P)**

Description: Holds current setting of the Z7 axial shim gradient.

Values: -32768 to $+32767$, steps of 1, 0 is no current.

See also: *Getting Started*

z8 **Z8 shim gradient (P)**

Description: Holds current setting of the Z8 shim gradient.

Values: -32768 to $+32767$, steps of 1, 0 is no current.

See also: *Getting Started*

zap **Set up for gradient refocused high-speed imaging sequences (M)**

Applicability: Systems with imaging capabilities.

Syntax: `zap`

Description: Sets up a pulse sequence consisting of a slice-selective excitation pulse to generate transverse magnetization.

See also: *User Guide: Imaging*

Related: `gss` Slice selection gradient strength in DAC units (P)

zeroneg **Set all negative intensities of 2D spectra to zero (C)**

Syntax: `zeroneg`

Description: Sets to zero all negative intensities of 2D-J spectra.

See also: *User Guide: Liquids NMR*

Related: `foldj` Fold J-resolved 2D spectrum about $f_1=0$ axis (C)
`rotate` Rotate 2D data (C)

zoom **Adjust display to given width (M)**

Syntax: `zoom(width)`

Description: Adjusts the display limits. It is useful in the display of powder patterns after `split` has been used. `zoom` both zooms in and out from the current display.

Arguments: `width` is the total display width, in Hz. Display limits are set to $\pm\text{width}/2$.

See also: *Getting Started*

Related: `split` Split the difference between two cursors (M)

zx2y2 ZX2Y2 shim gradient (P)

Description: Holds current setting of the ZX2Y2 shim gradient.

Values: If `shimset` is 2, 8: -2048 to +2047, steps of 1, 0 is no current.
If `shimset` is 3 to 7, 9: -32768 to +32767, steps of 1, 0 is no current.

See also: *Getting Started*

Related: `shimset` Type of shim set (P)

zx3 ZX3 shim gradient (P)

Description: Holds current setting of the ZX3 shim gradient.

Values: -32768 to +32767, steps of 1, 0 is no current.

See also: *Getting Started*

zxy ZXY shim gradient (P)

Description: Holds current setting of the ZXY shim gradient.

Values: If `shimset` is 2, 8: -2048 to +2047, steps of 1, 0 is no current.
If `shimset` is 3 to 7, 9: -32768 to +32767, steps of 1, 0 is no current.

See also: *Getting Started*

Related: `shimset` Type of shim set (P)

zy3 ZY3 shim gradient (P)

Description: Holds current setting of the ZY3 shim gradient.

Values: -32768 to +32767, steps of 1, 0 as no current.

See also: *Getting Started*

Symbols

& (ampersand) character, 523
 .def files, copying, 551
 .talk file, 319
 / (slash) character, 318
 ? (question mark) character, 456
 @ (at) character, 318

Numerics

¹³C gHSQC exp, setting up parameters for, 262
¹³C HMQC exp, changing parameters for, 286
¹³C HMQCTOXY exp, changing params for, 287
¹³C HSQC exp, changing parameters for, 292
¹³C HSQCTOXY exp, changing parameters for, 293
¹⁵N gHMQC exp, setting up parameters for, 260, 261
¹⁵N gHSQC exp, setting up parameters for, 262
¹⁵N HMQC exp, changing parameters for, 286
¹⁵N HMQCTOXY exp, changing params for, 287
¹⁵N HSQC exp, changing parameters for, 292
¹⁵N HSQCTOXY exp, changing parameters for, 293
 16-bit integer precision, 168
 180-degree pulse power calibration, 577
 180-degree refocus pulse
 length, 381
 shape, 382
 1st indirectly detected dimension
 absolute value display mode, 80
 additive weighting constant, 82
 clear reference line, 118
 cursor difference, 137
 cursor position, 96, 115
 data display mode, 159
 first-order phase, 330
 Fourier number, 228
 Gaussian function, 259
 Gaussian shift constant, 259
 incremented delay, 124
 line broadening, 315
 number of increments of evolution time, 366
 phased spectra display mode, 383, 407
 power display mode, 452
 reference line frequency, 474
 reference line position, 473
 scale spectral width, 491
 set frequency referencing, 516
 set reference line, 477
 sinebell constant, 488
 sinebell shift constant, 489
 spectra width, 558
 start of plot, 532
 user-defined weighting, 618
 width of plot, 614
 zero-order phase, 481
 200-kHz receiver option, 168
 2D display, showing, 246
 2D DOSY display
 building up, 229
 2D experiments
 acquire and Fourier transform, 248
 axis labels, 83
 baseline correction, 86
 color intensity map, 128
 combine arrayed FID matrices, 237, 242
 control dconi display, 131
 convert compressed data to standard format, 225
 copy peak picking file to another file, 322
 create experiment, 97
 create new parameters, 498
 create parameters, 386
 create peak picking parameters, 390
 cross-relaxation experiment, 574
 data display, 129
 display a spectrum, 180
 display FIDs, 140
 display resolution, 61
 display spectra in whitewash mode, 181, 182
 draw grid on 2D display, 272
 exchange experiment, 371
 f₂ ridges, 232
 find and integrate peaks, 320
 first point multiplier, 97, 232
 first-order phase set to zero, 93
 fold J-resolved spectrum, 230
 Fourier transform 2D data, 238
 Fourier transform arrayed 2D FID data, 238
 general setup, 498
 gray scale image, 297
 heteronuclear 2D-J, 284
 heteronuclear chemical shift correlation, 284
 homonuclear J-resolved 2D, 288
 horizontal axis selection, 579
 INADEQUATE pulse sequence, 301
 incremented delay, 124
 intensity of spectrum at a point, 350
 interactive weighting, 619
 interleaving control, 296
 J-correlation experiment, 574
 LC-NMR acquisition parameters, 316
 normalization, 369
 number of increments of evolution time, 366
 parameter creation, 46
 peak integration threshold, 570
 peak picking display control, 323
 peak picking parameters, 47
 phase selection, 409
 plot 2D peak picking results, 424
 plot grid over 2D plot, 421
 plot heteronuclear J-resolved 2D spectra, 421
 plot homonuclear J-resolved 2D spectra, 422
 plot X,H-correlation 2D spectrum, 423
 plotter units conversion, 295
 processing mode for 2D data, 430
 processing parameter group, 144
 project 2D data onto axis, 442
 pseudo-echo weighting parameters, 444
 reverse detection heteronuclear multiple quantum, 286
 rotate 2D data, 480
 search data set for maximum intensity, 402
 select for processing, 253
 set scaling factor, 491
 sinebell weighting, 527
 spectra plotting, 414
 spectra plotting in whitewash mode, 415

Index

- spectra processing, 440
- spectral drift correction, 127
- stacked spectra display, 189
- start of chart in second direction, 490
- submit to acquisition, 268
- symmetrize INADEQUATE data, 229
- t_1 dimension, 339
- type of data processing, 439
- vertical offset of traces, 597
- volume value, 303
- weight and Fourier transform, 610
- weight and Fourier transform 2D data, 611
- weight and Fourier transform phase-sensitive data, 610, 612
- 2D experiments setup, 497
- 2D Line List button, 322
- 2D phase encode image center position, 433
- 2D phasefiles
 - calculate, 298
 - format arguments, 299
- 2D spectra, plotting, 426
- 2D spectra, setting negative intensities of, 630
- ^2H chemical shift, 531
- 2nd evolution dimension
 - set spectral width, 517
- 2nd indirectly detected dimension
 - absolute value display mode, 80
 - additive weighting constant, 82
 - clear reference line, 118
 - cursor difference, 137
 - cursor position, 96, 115
 - data display mode, 159
 - first-order phase, 330
 - Fourier number, 229
 - Gaussian function, 259
 - Gaussian shift constant, 260
 - incremented delay, 124
 - line broadening, 315
 - number of increments of evolution time, 367
 - phased spectra display mode, 408
 - power mode processing, 452
 - reference line frequency, 474
 - reference line position, 473
 - right phase, 481
 - scale spectral width, 491
 - set frequency referencing, 516
 - set reference line, 478
 - sinebell constant, 489
 - sinebell shift constant, 489
 - spectral width, 558
 - start of plot, 532
 - user-defined weighting, 618
 - width of plot, 614
- 32-bit integer precision, 168
- 3D experiments
 - 3D plane index selected, 302
 - 3D plane projection selected, 302
 - 3D plane type currently displayed, 417
 - axis labels, 83
 - create experiment, 97
 - create parameters, 387
 - display 2D color map of plane from 3D data, 171
 - display 2D projection plane from 3D data, 172
 - display 3D data file, 149
 - display group of parameters, 145
 - display next 3D plane, 366
 - display previous 3D plane, 436
 - display series of 3D planes, 188
 - extract planes from 3D spectral data, 255
 - f_3 ridges, 232
 - find and integrate peaks on 2D plane, 320
 - first point multiplier, 232
 - Fourier transform 3D FID into 3D data, 242
 - Fourier transform arrayed 3D data sets, 238
 - horizontal axis selection, 579
 - incremented delay, 124
 - N-type display, 374
 - number of increments of evolution time, 367
 - parameter creation, 46
 - path to 2D planes, 393
 - phase cycling type, 409
 - plot peak picking on 2D plane, 425
 - plot series of 3D planes, 427
 - process f_3 dimension, 612
 - processing coefficient file, 498
 - region selective 3D processing, 447
 - reset parameters after partial transform, 467
 - select 2D plane without displaying, 494
 - selective 2D processing, 241
 - selective transformation, 240
 - set 3D processing, 498
 - spectral dc correction, 533
 - t_1 and t_2 dimensions, 339
 - terminate 3D FT process, 311
 - time-domain dc correction, 217
 - transformed data file, 108
 - type of data processing, 439
 - weight and FT phase-sensitive data, 610, 612
- 3rd indirectly detected dimension
 - incremented delay, 125
 - number of increments of evolution time, 367
 - spectral width, 558
- 3rd rf channel
 - create parameters, 224
 - display group of parameters, 145
 - display template for parameters, 387
 - parameter retrieval, 47
- 4D experiments
 - create acquisition parameters, 387
 - incremented delay, 125
 - number of increments of evolution time, 367
 - parameter creation, 47
 - phase cycling type, 409
- 4nuc (HCPF) experiment, 68
- 4th rf channel
 - create parameters, 225
 - display group of parameters, 145
- 5th rf channel
 - create parameters, 225
- 63-dB attenuator, 94
- 79-dB attenuator, 94
- 90-degree pulse, 381
- 90-degree pulse power calibration, 577
- 90-degree pulse width, 450

A

- aborting acquisition
 - with error, 35
 - with no error, 280
- absolute intensity display mode, 50
- absolute intensity group, 50
- absolute magnet frame data, generating, 481
- absolute-value 2D experiment, 409
- absolute-value COSY pulse sequence, 466
- absolute-value data display mode, 158, 159
- absolute-value data file, 108
- absolute-value display mode, 79
- absolute-value MQF COSY parameter set, 267
- absolute-value ROESY parameter set, 276
- accounting program, 596
- acq_errors file, 42
- acqaddr parameter, 233
- acqbin directory, 497
- acqfil directory, 211, 562
- Acqmeter window, 39, 40
- acquisition
 - abort with error, 35
 - abort with no error, 280
 - acquire FID with no processing, 268
 - acquisition parameter arrays, 126
 - action when bs transients accumulate, 604
 - action when error occurs, 606
 - action when specified transients accumulate, 613
 - array of acquisition parameter, 499
 - automated proton and carbon, 281
 - automated proton and COSY, 282
 - automated proton, carbon, DEPT, 282
 - automated proton, carbon, HETCOR, 282
 - calculate pixel size, 467
 - carbon, 92
 - carbon and APT automatically, 93
 - carbon and DEPT automatically, 95
 - create 2D parameters, 386
 - create 3D acquisition parameters, 387
 - create 4D acquisition parameters, 387
 - data points to acquire, 62
 - data points to be acquired, 372
 - date data is acquired, 126
 - delay before acquisition, 50
 - determine if active for experiment, 206
 - display status information, 526
 - DSP type, 187
 - estimate acquisition time, 571
 - fluorine, 212
 - GLIDE windows initiated, 263
 - hardware values, 507
 - interactive display, 38
 - LC-NMR 2D parameters, 316
 - loop control, 496
 - make equal to time requested, 571
 - number of echoes, 364
 - number of scans, 373
 - number of slices, 373
 - number of transients, 373
 - oversampling factor, 379
 - perform Autoshim experiment, 523
 - perform experiment, 64
 - phosphorus, 382
 - read hardware values, 461
 - recover from error, 460
 - resume paused queue, 468
 - resume stopped acquisition, 458
 - stop acquisition, 487
 - stopped by temperature interlock, 572
 - submit Autolock experiment, 325
 - submit change sample, Autoshim experiment, 487
 - submit setup experiment, 548
 - submit spin setup experiment, 534
 - time to acquire FID, 62
 - trigger pulses before acquisition, 571
 - trigger signals to wait, 374
- acquisition bus trap, 44
- acquisition computer
 - block size, 91
 - resetting, 35
- Acquisition Controller board, 220, 480
- acquisition parameters group, 144
- acquisition queue, resume after pause, 468
- acquisition status, 42
- acquisition status line, 38
- Acquisition Status window, 40, 41
- activating current window activity, 310
- active parameter, 376
- active pulse length parameter list, 424
- active pulse power level parameter list, 453
- activity in current window, 310
- ADC overflow warning, 42
- add series of FIDs together, 45
- Add Spectrum button, 533
- Add/Subtract button, 46
- add/subtract experiment, 538
 - add current FID, 44
 - add current spectrum, 532
 - clear experiment, 100
 - delete experiment, 100
 - interactive mode, 45
 - subtract FID, 549
 - subtract spectrum, 539
- additive weighting constant, 81, 619
- Adj VS button, 599, 600, 601
- Adj Weighting button, 619
- administration tool for GLIDE, 249
- All Params button, 386
- allocateWithId procedure, 362
- AM data conversion, 109
- ampersand (&) character, 40, 41
- amplifier band in use, 469
- amplifier mode control, 51
- amplifier type, 52
- AMX data conversion, 109
- AnalogPlus digital filter, 135
- analyze.inp file, 53, 207, 209
- analyze.list file, 54, 108, 560, 561
- analyze.out file, 54, 108, 209, 403
- AP Interface board, 57, 479
- AP Interface Type label, 57, 104
- App Mode button, 58
- application code dimension, 364
- application mode, 57
- APT acquisition, 281
- APT button, 58

Index

- APT experiment, changing parameters for, 58
- APT pulse sequence, 58
- APT spectra
 - plot automatically, 417
 - process automatically, 58
- arc cosine calculation, 37, 58
- arc sine calculation, 59
- arc sine of number, 61
- arc tangent calculation, 59, 62, 63
- arc tangent of two numbers (Y,X), 63
- array index for transformed image, 200
- array of an acquisition parameter, 499
- arrayed 1D spectra, 545
 - processing, 440
 - processing and plotting, 78
- arrayed 2D FID matrices, 611, 612
- arrayed experiment
 - control interleaving, 296
- arrayed imaging data
 - fit to T₁ data, 560
 - fit to T₂ data, 561
- arrayed parameters
 - enter as linearly spaced, 60
 - order and precedence, 60
- arraying LP parameters, 335
- assign sysgcoil, 504
- asynchronous decoupler mode, 154
- attached proton test, 58
- attenuator
 - coarse type, 94
 - control, 175
 - fine, 213
 - present in system, 63
 - upper safety limit, 173
- attributes of parameters, 150
- audio filter board, 67
- Audio Filter Type label, 67, 103
- audio filters bandwidth, 214
- auto assign button, 62
- auto lk gradient map generation, 69
- Auto Spinner label, 105, 106, 536
- auto.conf file, 201
- autocalibration, 36
- autocalibration macros, 65, 66
- autocalibration routines, 67, 68
- autocalibration with autotest sample, getting, 65
- autocalibration with autotest sample, set up, 65
- autocalibration with CH3I sample, getting, 65
- Autogain, see automatic gain
- AutoLIST, creating, 78
- Autolock, see automatic lock
- Automake Shimmap button, 266
- automated
 - analysis of DEPT data, 73
 - carbon acquisition, 92
 - carbon and APT acquisition, 93
 - carbon and DEPT acquisition, 95
 - fluorine acquisition, 212
 - phosphorus acquisition, 382
 - proton acquisition, 279
 - proton and carbon acquisition, 281
 - proton and COSY acquisition, 282
 - proton, carbon, APT acquisition, 281
 - proton, carbon, DEPT acquisition, 282
 - proton, carbon, HETCOR acquisition, 282
- automated gradient map generation macros, 68
- automatic
 - 2D normalization, 369
 - 2D peak picking, 320
 - 2D processing, 371
 - analysis of COSY data, 37
 - APT spectra processing, 58
 - calibration, 36
 - COSY- and NOESY-type spectra plot, 419
 - generic processing, 441
 - heteronuclear J-resolved 2D spectra plot, 421
 - homonuclear J-resolved 2D spectra plot, 422
 - integral scale adjustment, 306, 307
 - macro execution, 514
 - plot APT-type spectra, 417
 - process FIDs, 441
 - spectra plotting, 425
 - vertical scale adjustment, 599, 600
 - X,H-correlation 2D spectrum plot, 423
- Automatic button, 64
- automatic gain
 - enable Autogain, 249
 - errors, 43
- automatic lock
 - errors, 43
 - status, 51
 - submit Autolock experiment to acquisition, 325
- automatic phasing, 55
 - optimized, 56
 - zero-order term, 56
- automatic sequence setup for gradients, 412
- automatic shimming, 617
 - create shim method string, 365
 - method selection, 354
 - submit Autoshim experiment to acquisition, 487, 523
- automatic stacking for arrays, 78
- automatic vertical scale adjustment, 600, 601
- automation custom queue setup, 120
- automation data file prefix, 75
- automation directory
 - absolute path, 73
 - check for enter queue, 73
 - preparation for run, 70
- automation directory name, 264
- automation mode, 73
 - check if active, 70, 206
- automation parameter group, 147
- automation run
 - controlling macro, 71
 - enter sample information, 200
 - prepare automation directory, 70
 - resume suspended run, 77
 - starting, 73
 - suspend current run, 78
- Autophase button, 56
- autoscaling resumes, 78
- Autoshim on Z button, 265
- Autoshim, see automatic shimming
- autoshimming, 545
 - gradient, 264
- average value of input, 81
- axis gradients, 105

- axis labels, 82
 - FID displays and plots, 58, 83
 - units, 82

- B**
- background execution, 523
- background VNMR processing, 591
- backup current probe file, 36
- balance gradients, 197
- bandpass filter offset for downsampling, 185
- bandpass filter offset for oversampling, 378
- bandwidth for shaped pulse, 448
- bandwidth of audio filters, 214
- bandwidth of digital filter, 183
- baseline correction, 86, 573
 - linear, 127
 - sensitivity adjustment, 341
 - zero-order, 341
- baseline flatness, 290
- BB Atten Type label, 63, 106
- beeper sound, 87
- Bessel filters, 50
- beta delay, 50
- BINOM button, 87
- BINOM pulse sequence, 87
- binomial water suppression, 87
- blanked amplifiers, 479
- block size action, 605
- block size storage, 91
- block size transients, 604
- bore size of magnet, 88
- boxes
 - draw on plotter or display, 88
 - selected by mark command, 89
- BR24 pulse sequence, 90, 121
- Brickwall digital filter, 135
- broadband channel tuning, 91
- Bruker data files, 541
 - convert to VNMR, 109
 - read files from 9-track tape, 461
- Bruker data, phasing, 56
- bruker.par file, 541
- Butterworth filter, 50, 67
- button labels, 358
- button values, reporting, 263

- C**
- C13 & C13-detected experiments, getting parameters for, 66
- C13.par file, 519
- Calculate Target button, 416, 529, 543, 598
- calculated spectrum display, 186
- calibration
 - decoupler pulse, 454
 - gradient strength, 442
 - gradients, 277
 - rf pulse identity, 472
 - shaped pulses, 85
- calibration chain, running routines in, 71
- calibration directory, saving FID in, 72
- calibration file, printing, 71
- calibration routine, removing from GLIDE chain, 72
- calibration routine, starting, 71
- carbon
 - acquisition, 281, 282
 - automated acquisition, 92
 - plotting, 418, 428
 - process 1D carbon spectra, 92
 - vertical scale adjustment, 600
- carbon 1D experiment, getting parameters for, 65
- carbon channel tuning, 119
- carbon decoupler calibration macros, 66
- carbon experiments, getting parameters for, 66
- carbon gradient ratio calibration macros, 67
- carbon observe calibration macros, 67
- carbon parameter set, getting, 65
- carbon spectrum, setting up parameters for, 94
- carbon-enriched molecules, 281
- Carr-Purcell Meiboom-Gill T2, 113
- cartridge tape, 563
- Center button, 96
- center frequencies of nD experiments, 465
- center of screen display limits, 96
- center sequence calibration, 499
- chained acquisition, 74
- chained experiments, 99
- chained experiments, running, 74
- CHAN readout, 582
- change sample experiment, 98
- changing working directory, 94
- channels
 - assign frequencies for probe tuning, 582
 - available for use, 375
 - rf frequencies, 533
 - rf generation on each channel, 475
 - set frequency of rf channels, 502
 - waveform generator on channel, 476
- characters in a string, 318
- chart
 - maximum width, 605
 - maximum width in second direction, 606
 - starting position, 490
 - starting position in second direction, 490
 - width, 605
 - width in second direction, 605
- chart paper
 - preprinted paper for HP plotters, 291
- chemical shift offset frequency, 512
- chemical shifts list, storing, 165, 401
- chemist-style parameters, 89
- class C amplifiers, 52, 57, 94, 105
 - decoupler high-power control, 148
 - decoupler low-power control, 153
- Clear button, 100
- Clear Marks button, 416
- clearing
 - experiment text, 119
 - integral reset points, 121
- cmd parameter, 382
- coarse attenuator control, 153
- Coarse Attenuator label, 94, 104
- coarse attenuator type, 94
- coef file, 498, 534
- coefficient to construct interferogram, 213

Index

- coefficients for digital filtering, 182, 542
- coil calibration data, 448
- COLOC sequence, 423
- color intensity map, 298
 - display, 128
 - without screen erase, 131, 140
- color selection for drawing, 402
- colors for plotting, 100, 513
- combining arrayed 2D FID matrices, 237, 242
- command execution, 204
- commands
 - display which command or macro is used, 613
 - edit online description, 349
 - online description, 349
 - rename, 285
- comparing shim sets, 149
- compiling
 - user PSG object library, 444
 - user pulse sequences, 497
 - user-written weighting functions, 618
- compiling pulse sequence, 497
- completed FIDs in experiment, 95
- completed transients, 119
- complex 3D transformed data file, 108
- complex Fourier transform, 438, 439
- complex points to left-shift ni interferogram, 339
- complex points to left-shift ni2 interferogram, 339
- complex points to left-shift np FID, 338
- complex time-domain data points, 334
- compressed 2D data conversion, 225
- configuration information, 616
- configuration parameters
 - display and possibly change, 102
- Configure label, 104
- conjugate gradient list, 544
- conpar file, 102, 103, 116
- console hardware status, 296
- Console label, 103
- console parameter, 107
- contact time, 108
- continuous wave (CW) modulation, 160
- contour display
 - display control, 131
- contour plot, 128, 400
 - display, 168
 - width of plotting area, 605
 - without screen erase, 131
- contour plot display, 169
- conversion units for parameters, 585
- converting
 - 32-bit data files to VNMR, 541
 - Bruker data, 109
 - compressed 2D data to standard format, 225
 - data in table order to linear order, 562
 - Hz or ppm to plotter units, 294
 - UNIX text file to VXR-style format, 586
 - VXR-style data to VNMR, 114
 - VXR-style text files to UNIX, 602
- coordinate information from image display, 325
- copying
 - experiment data to subfile, 114
 - files, 112, 113
 - local file to remote host, 204
 - one parameter tree to another, 274
 - peak file to another file, 322
 - remote file to local host, 203
 - stored phasefile to current experiment, 483
 - system macro to become user macro, 345
 - user macro files, 342
- corrected difference between successive spectra, 219
- correlated spectroscopy, 112
- cosine value, 112
- cosine-squared window function, 540
- cost accounting, 596
- COSY
 - acquisition, 282
 - automatic analysis and plot, 384
 - automatic analysis of data, 37
 - phase-sensitive, 113
 - plotting, 419
 - pulse sequence, 112
- COSY button, 112
- COSY experiment, changing parameters for, 112
- COSY-like correlation spectra, 230
- COSYPS button, 113
- cp command (UNIX), 112, 113
- CPMG2 pulse sequence, 113
- Create New button, 97
- creating
 - FID display parameters, 219
 - LC-NMR parameters, 390
 - parameters in a parameter tree, 115
 - UNIX directory, 358
- cross-polarization, 622
- cross-relaxation, 574
- crusher gradient level, 252
- cubic curve fitting, 54, 208
- curecc file, 503
- curpar file, 116
- current experiment
 - correct parameter characteristics, 224
 - determine if acquisition active, 206
- current FID data block, 97
- current gradient coil, 251
- current window, 120
- current working directory, 451
- current-type parameter tree, 116
- cursor
 - adjust tau2 to start of acquisition, 582
 - difference of two frequency cursors, 137
 - difference of two time-domain cursors, 137
 - frequency offset calculation, 376
 - mode, 350
 - move cursor to center spectrum, 96
 - move cursor to nearest line, 367
 - move spectral window according to cursors, 360
 - reset integral to zero at cursor, 626
 - set decoupler frequency to cursor position, 492
 - split difference of two cursors, 538
 - state in df, ds, or dconi programs, 118
- cursor position, 115
 - time domain, 117
- curve fitting, 53, 207
- cutoff point for VT regulation, 601
- CW amplifier mode, 51
- cycle phase, 113
- cycled BR24 pulse sequence, 121

- cycled MREV8 pulse sequence, 121
- CYCLENOE sequence, 121

- D**
- D2PUL button, 124
- D2PUL pulse sequence, 124
- DAC, converting gauss/cm value to, 220
- data acquisition mode, 511
- data conversion to linear order, 562
- data display mode, 158, 159
- data file display in current experiment, 132
- data point, determining size of a, 56
- data points to be acquired, 372
- data processing type on FID, 438, 439
- data set conversion from VXR-style to VNMR, 109, 114
- data station system configuration, 559
- data truncation limit, 121
- data, saving in GLIDE run, 75
- data.fdf file, 243, 245
- datadir directory, 613
- date of data acquisition, 126
- DC button, 127
- dc correction, 217
- dc offsets removed from FIDs, 132
- dconi.out file, 350
- decay curves, 406
- deconvolution, 217, 223
 - display numerical results, 525
 - plotting, 420
 - starting point, 588
- decoupler
 - adjust tip-angle resolution time, 156
 - decoupling sequence, 183
 - field strength, 155
 - field strength calculation, 280
 - fine power attenuator, 175
 - frequency, 141
 - frequency offset array, 492
 - frequency offset control, 164
 - high-power control, 148
 - homodecoupling control, 289
 - linear modulator power, 176
 - low-power mode, 153
 - mode during status periods, 154, 155
 - modulation frequency, 103, 155, 156
 - modulation mode, 160, 161
 - nucleus lookup, 162
 - observe channel uses decoupler hardware, 486
 - power level with linear amplifier, 173
 - power to switchable probe caution, 173, 174, 175
 - proton decoupler pulse calibration, 432
 - pulse calibration, 454
 - pulse length, 432
 - pulse sequence diagram, 173
 - set frequency to cursor position, 492
 - tip-angle resolution, 179
 - used as transmitter, 124
 - used for pulsing, 160
 - WALTZ decoupling present, 604
- decoupler 2 parameter values
 - set from probe file, 501
- decoupler modulation frequency, 156
- decoupler parameter values
 - set from probe file, 501
- def file, 163
- Default button, 135
- default directory for Files menu system, 135
- defaultdomain file, 502, 512
- defaultrouter file, 502, 512
- Define excitation band, 395, 494
- Define excitation band for solvent suppression, 396
- delay
 - first, 123
 - incremented delay for pulse sequence, 124, 125
 - overhead delay between FIDs, 123
 - post-trigger, 288
 - preacquisition, 50, 384
 - pre-trigger, 460
 - wait for another trigger, 196
 - wait to acquire a spectrum, 196
- delay-type parameter, 116
- Delete button, 136
- deleting
 - experiments, 136
 - file, parameter, or FID directory, 135
 - files, 478
 - spectra from analysis, 136
 - user macro, 135
- DEPT
 - acquisition, 282
 - analysis and plot, 385
 - automated complete analysis, 73
 - automatic analysis and spectrum editing, 49
 - plotting data, 419
 - pulse sequence, 137
 - spectra array processing, 138
- DEPT button, 137
- DEPT experiment, changing parameters for, 138
- DEPT experiments, getting parameters for, 66
- dept.out file, 49
- DEPTGL pulse sequence, 138
- destroying
 - parameters, 138
 - parameters of a group, 139
- devicenames file, 513
- devicetable file, 513
- dg window, 565
- Dgroup of a parameter, 501
- diagonal parameter arrays, 60
- diagonal peaks threshold during peak picking, 621
- dialog box, 107
- dialog box from a macro, 148
- dialog, starting a, 163, 164, 165, 201
- difference between cursors in Hz, 137
- difference between cursors in seconds, 137
- difference NOE experiment, 121
- diffusion analysis, 208
 - add to current display, 210
 - add to current plot, 403
 - display, 209
- diffusion constant, 405
 - calculation, 406
- diffusion experiment analysis, 54
- diffusion gradient level, 252

Index

- Diffusion Ordered Spectroscopy (DOSY), 166
- digital filter type, 135
- digital filtering
 - bandwidth, 183, 542
 - bandwidth for oversampling, 378
 - coefficients for oversampling, 377
 - create downsampling parameters, 389
 - create parameters for oversampling, 392
 - downsampling factor, 168
 - file of FIR digital filter coefficients, 222
 - inline type, 187
 - number of coefficients, 182, 542
 - parameter creation, 47
- digital lock display, 461
- digital resolution measurement, 178
- digitally filtered FIDs, 149, 543
- dimension of application code, 364
- dimension of experiment, 61, 92
- dimensionality of experiment, 253
- dimensions of voxel, 597
- diode switching version, 63
- direct polarization, 622
- directly detected dimension
 - absolute value display mode, 79
 - additive weighting constant, 81
 - data display mode, 158
 - first-order phase, 330
 - Fourier number, 228
 - Gaussian function, 258
 - Gaussian shift constant, 259
 - line broadening, 315
 - phase angle display mode, 383
 - phased spectra display mode, 407
 - power display mode, 451
 - reference line frequency, 474
 - reference line position, 473
 - scale spectral width, 491
 - set reference line, 477
 - sinebell constant, 488
 - sinebell shift constant, 489
 - spectral width, 557
 - start of plot, 532
 - user-defined weighting, 617
 - width of plot, 614
 - zero-order phase, 480
- directories
 - change working directory, 94
 - create new UNIX directory, 358
 - default for Files menu system, 135
 - delete, 135
 - display current working directory, 451
 - get information about files, 254
 - list files, 149
 - list files in directory, 318, 338
 - move directory, 363, 466
 - path to current experiment, 120
 - remote VXR-style system, 200
 - remove empty directories, 479
 - remove from experiment, 99
 - rename directory, 363, 466
 - stored queue experiments, 73
 - user's macro directory path, 342
 - user's menu directory, 353
 - user's help directory, 284
 - user's manual directory, 349
 - user's private VNMR files, 589
 - VNMR system, 559
- disk file errors, 44
- display
 - acquisition information, 41
 - lock level, 40
 - spinner speed, 40
 - temperature, 40
- Display FID button, 140
- display limits
 - set for full screen, 246
 - set for full screen with room for traces, 246
 - set for left half of screen, 317
 - set for right half of screen, 476
- display mode for plotter, 428
- display parameters
 - create 3D display parameters, 387
 - full recall of set, 233
 - move between experiments, 353
 - recall set, 458
 - save as a set, 486
 - set to full spectrum, 212
- display parameters group, 145
- display templates for 3rd rf channel parameters, 387
- display templates for pulse sequence, 518
- displaying
 - 2D color map of 3D plane, 171
 - 2D data interactively, 129
 - 2D spectra in whitewash mode, 181
 - 2D spectra in whitewash mode with no screen erase, 182
 - 3D data file, 149
 - 3D parameter group, 145
 - 3D plane projection, 172
 - 3D planes, 188
 - 3rd/4th rf channel parameter group, 145
 - acquisition information, 40
 - acquisition parameter group, 144
 - acquisition status information, 526
 - acquisition time, 211
 - add another diffusion analysis, 210, 403
 - adjust display parameters, 223
 - arrayed acquisition parameters, 126
 - automation parameters, 147
 - color intensity map, 128
 - contour plot, 128
 - contour plot with screen erase, 169
 - contour plots, 168
 - create 2D parameters, 386
 - current working directory, 451
 - data file in current experiment, 132
 - dialog box from a macro, 148
 - display parameter group, 145
 - error messages, 203
 - Ethernet address, 197
 - experiment library, 210
 - experiment time, 211
 - exponential curves, 209
 - FID, 139, 140
 - FID as connected dots, 167
 - FID file in current experiment, 132
 - FIDs in whitewash mode, 144
 - FIDs of 2D experiment, 140

- files in directory, 149
 - formatted text, 614
 - full window display, 620
 - GLIDE administration tool, 249
 - grid on 2D display, 272
 - help information, 284
 - horizontal LC axis, 126
 - inset spectrum, 304
 - integral amplitudes, 170
 - integral with a spectrum, 180
 - integrals at reset points, 151
 - LC-NMR parameters, 147
 - limNET nodes, 163
 - line frequencies above threshold, 152
 - linear prediction parameter group, 147
 - lock level, 39
 - log file for experiment, 210
 - menu for planning target scan, 415
 - message on acquisition status line, 38
 - multiple images, 160, 185
 - next 3D plane, 366
 - noninteractive gray scale image, 297
 - normalized integral amplitudes, 170
 - normalized integrals, 153
 - parameter screen menu, 147
 - parameter value, 456
 - parameters and their attributes, 150
 - peak frequencies, 169
 - phase file in current experiment, 132
 - plot is same as display, 620
 - plotted contours, 168
 - polynomial curves, 209
 - previous 3D plane, 436
 - processing parameter group, 144
 - pulse calibration data file, 448
 - pulse sequence diagram, 172
 - pulse sequence generation errors, 444
 - recalculated simulated spectrum, 186
 - remote VXR-style directory, 200
 - scale under spectrum or FID, 182
 - set for center of screen, 96
 - set full screen with room for traces, 247
 - set limits for full screen, 246
 - shim method string, 185
 - shim parameter group, 147
 - spectra in whitewash mode, 195
 - spectrum, 180
 - spin simulation parameter arrays, 151
 - spin simulation parameter group, 146
 - spinner speed, 39
 - stacked FIDs, 142, 143, 144
 - stacked spectra, 189
 - stacked spectra automatically, 190
 - stacked spectra automatically with no screen erase, 192
 - stacked spectra horizontally, 192
 - stacked spectra horizontally with no screen erase, 193
 - stacked spectra with no screen erase, 194
 - strings in text window, 198
 - system macro file, 344
 - target slices, 178
 - target voxels, 178
 - temperature, 39
 - text file for current experiment, 568
 - text file in graphics window, 195
 - text files, 94
 - time of acquisition, 211
 - user macro files, 342
 - which command or macro is used, 613
 - width adjustment, 630
 - Distortionless Enhancement by Polarization Transfer, 137
 - done codes, 42
 - DOSY (Diffusion Ordered Spectroscopy) experiment, 166
 - double-precision data acquisition, 168
 - double-precision VNMR FID data, 101
 - double-quantum filtered COSY pulse sequence, 177
 - downsampling
 - bandpass filter offset, 185
 - bandwidth of digital filtering, 183
 - creating parameters, 388
 - digital filter coefficients, 182
 - factor, 168
 - inline type, 187
 - parameter creation, 47
 - setting parameters, 359
 - DPFGSE-noe experiment, changing parameters for, 371, 575
 - DQCOSY button, 177
 - DQCOSY experiment, 177
 - DQCOSY experiment, changing parameters for, 177
 - drawing a line between points, 177
 - drift correction
 - 2D spectra traces, 127
 - activity flag, 128
 - calculation, 127
 - cancel, 95
 - group, 128
 - ds, 130
 - ds.out file, 350
 - DSP parameter creation, 47
 - DSP type (see digital filtering), 187
 - dummy scans, 541
 - Dynamic Angle Spinning (DAS), 126
 - dynamic binding, 497
- ## E
- ecc file, 197
 - ecctabl reference table, 197, 254
 - eccTool window display, 198
 - echo command (UNIX), 198
 - echo planar imaging. *See* EPI experiments
 - echo position, determine, 270
 - echo time, 566
 - echoes
 - index for transformed image, 198
 - number to be acquired, 364
 - eddy current
 - compensation data, 197
 - compensation data analysis, 198
 - compensation file, 120, 402
 - settings, 199
 - testing, 272
 - eddylib directory, 197, 254

Index

- editing
 - files, 199
 - macros, 343
 - menu file, 354
 - parameter file with user-selected editor, 387
 - parameter file with vi editor, 388
 - UNIX text files, 592
 - user macro, 346
- ejection of sample, 197, 200
- elliptical filters, 50, 67
- enter.conf file, 201
- enterexp file, 201
- enumerated values, 116
 - remove from a variable, 501
- EPI experiments
 - acquisition delay time, 567
 - apply phase correction map, 399
 - calculate slice gradient, 544
 - calculate slice selection parameters, 544
 - centering echoes, 568
 - close phase correction map, 399
 - collect EPI data, 202
 - control phase encoding gradient, 297
 - display EPI data, 202
 - display image, 201
 - effective echo position, 199
 - generate phase correction map, 399
 - generate phase file, 201
 - number of EPI images to collect, 297
 - open phase correction map, 400
 - process EPI data, 202
 - process image, 201
 - readout dephasing gradient adjuster, 274
 - readout gradient adjuster, 273
 - readout gradient dephaser, 199
 - reverse spectral data, 202
 - save images in FDF for ImageBrowser, 202
 - set up parameters, 202
- Ernst angle pulse calculation, 203
- errmsg file, 618
- error codes, 42
- error conditions recovery, 460
- error during acquisition, 42, 606
- error handling control, 301
- error message display, 203, 204
- errors in pulse sequence generation, 444
- Ethernet
 - address display, 197
 - disconnect host computer, 512
 - host computer connection, 502
- Euler angle from magnet frame, 411, 446, 571
- evolution dimension
 - set spectral width, 517
- evolution time increments, 366, 367
- excitation pulse, 381
- excitation pulse power, 577
- executing VNMR command, 204
- Exit VNMR button, 206
- exiting from VNMR, 206, 596
- Exp# button, 308
- exp5 (add/subtract experiment), 46, 100
- experiment data retrieval, 484
- experiment directory path, 120
- experiment numbers list, 349
- experiment parameters, restoring, 309
- experiment text file
 - append string, 63
 - clear text, 119
- experiment time display, 210
- experiment, cleaning up, 72
- experiment, removing from GLIDE chain, 72
- experimental frequency of transition, 100
- experimental lines, assigning transitions, 61
- experiments
 - abort acquisition with no error, 280
 - acquire and Fourier transform, 248
 - acquisition time estimate, 571
 - action when bs transients accumulate, 604
 - action when error occurs, 606
 - action when experiment completes, 608
 - action when nt transients accumulate, 613
 - add parameters, 46
 - add parameters for FID display, 219
 - append string to text file, 63
 - calculate dimension, 92
 - clear text file, 119
 - completed transients, 119
 - correct parameter characteristics, 224
 - correct parameter limits and step sizes, 390
 - create workspace, 97
 - delete an experiment, 136
 - determine if acquisition active, 206
 - dimension, 61
 - dimensionality, 253
 - display acquisition time, 211
 - display data file, 132
 - display FID file, 132
 - display log file, 210
 - display phase file, 132
 - edit text file, 569
 - experiment library display, 210
 - fit data to lineshapes, 223
 - get text from data file, 257
 - join existing experiment, 308
 - make FID element using numeric text input, 347
 - move display parameters between experiments, 353
 - move FIDs between experiments, 354
 - move parameters between experiments, 361
 - nucleus selection, 519
 - number of completed FIDs, 95
 - parameters for basic experiment, 519
 - pulse sequence setup, 269
 - recalculate number of transients, 571
 - remove inactive lock and join experiment, 587
 - remove old files and directories, 99
 - replace text file, 568
 - resume a stopped acquisition, 458
 - retrieve FIDs from a file, 481
 - retrieve parameters from file, 482
 - retrieve shim coil settings, 483
 - save FIDs, 551
 - save parameters, 554
 - save text to a data file, 449
 - select 1D experiment for processing, 252
 - select 2D experiment for processing, 253
 - set up T_1 experiment, 167

- setup macro, 589
 - setup macros, 225
 - shim values to use, 324
 - shimming conditions, 617
 - solvent selection, 519
 - stop acquisition, 487
 - string parameters for storage, 364
 - submit Autolock experiment to acquisition, 325
 - submit Autoshim experiment to acquisition, 523
 - submit change sample, Autoshim to acquisition, 487
 - submit setup experiment to acquisition, 548
 - submit to acquisition, 64, 268
 - text file display, 568
 - experiments, setting up parameters for, 77
 - expfit.out file, 403
 - exponential analysis, 560, 561
 - exponential curve, 207
 - exponential curve fitting, 54, 108, 208
 - exponential curves display, 209
 - exponential curves plot, 403
 - exponential T_1 or T_2 data fitting, 300
 - exponential value of number, 206
 - exponential weighting, 315
 - external time base, 622
 - extract entries in VXR-style directory, 134
 - extrapolated dispersion mode, 218
- F**
- F1 linear prediction parameters, setting, 512
 - f_1 scaling factor for 2D multipulse sequences, 491
 - f_1 , f_2 display, 374
 - f_2 ridges, 97
 - f_3 processing of 3D data, 612
 - fast attenuators present, 63
 - FDF files, 149, 215, 216, 243, 245, 551, 552, 553, 555
 - FDM program, running, 216
 - FID block, move, 355
 - FID block, reverse, 469
 - FID button, 268
 - FID data, move, 356
 - FID data, reverse, 472
 - fid file, 562
 - FID file, checking for, 72
 - FID file, memory map open, 356
 - FID trace, move, 357
 - FID trace, reverse, 474
 - FID, memory map close, 355
 - FID, saving, 72, 74
 - fid.orig file, 562
 - FIDs
 - absolute-value mode data display, 159
 - acquisition time, 62
 - action after FID finishes, 614
 - action after last FID, 608
 - add series of FIDs together, 45
 - add to add/subtract experiment, 44
 - arrayed 2D FID matrices, 611, 612
 - automatic processing, 441
 - axis label units, 83
 - combine arrayed 2D FID matrices, 237, 242
 - complex points to left-shift np FID, 338
 - compress double-precision FID data, 101
 - copy FIDs into exp5 as array, 264
 - create display parameters, 47, 219
 - current data block, 97
 - cursor difference, 137
 - delete FID directory, 135
 - digitally filtered FID, 543
 - display as connected dots, 167
 - display FID files in current experiment, 132
 - display FID of 2D experiment, 140
 - display scale, 182
 - display single FID, 139, 140
 - display stacked FIDs, 142, 143, 144
 - display whitewashed FIDs, 144
 - file name prefix, 75
 - filtered, 542
 - first point multiplier, 232
 - Fourier transform 1D FIDs, 235
 - Fourier transform 2D data, 238
 - Fourier transform 3D FID into 3D data, 242
 - hypercomplex 2D Fourier transform, 241
 - imaginary part display, 140
 - interactive display, 38
 - interleave FIDs during processing, 296
 - left-shift FID to time-domain cursor, 573
 - make FID element using numeric text input, 347
 - move FIDs between experiments, 354
 - noise level measurement, 371
 - number acquired, 366
 - number of completed FIDs, 95
 - overhead delay between, 123
 - plot a scale under a FID, 443
 - plot in whitewash mode, 404
 - plot one or more FIDs, 420
 - prepare parameters for acqi display, 258
 - pulse breakthrough effects, 480
 - remove dc offsets, 132
 - retrieve from a file, 481
 - retrieve from experiment subfile, 484
 - save FID data in FDF format, 551
 - save in current experiment, 551
 - solvent subtraction, 389
 - start of FID display, 521
 - subtract FID from add/subtract experiment, 549
 - TPPI 2D Fourier transformation, 241
 - type of data processing, 438
 - vertical position, 598
 - vertical position of imaginary FID, 598
 - vertical scale, 592
 - weight and Fourier transform 1D FIDs, 610
 - weighting interactively, 619
 - width of FID display, 609
 - write numeric text file using a FID element, 616
 - zero-order phasing constant, 410
 - FIDs, recalling stored, 79
 - FIDs, storing, 75
 - field of view for 2nd phase-encode axis, 332
 - field of view for phase-encode axis, 332
 - field of view for readout, 338
 - field position, 626
 - FIFO loop size, 220

Index

- Fifo Loop Size label, 104, 220
- FIFO underflow error, 44
- File button, 221
- files
 - append data to file, 616
 - automation data file name prefix, 75
 - Bruker data files for conversion, 541
 - check for existence, 204
 - clear a file, 614
 - delete, 135
 - delete one or more files, 478
 - display experiment library, 210
 - display in text window, 94
 - edit with user-selectable editor, 199
 - file name extension information, 254
 - find number of files in directory, 254
 - find words and lines in text file, 328
 - get text from data file, 257
 - handle interactively, 221
 - help information file, 284
 - lines or records in file, 373
 - links to files, 318
 - list files in directory, 149, 318, 338
 - load parameters from file into a tree, 233
 - make a copy, 112
 - make FID files using numeric text input, 347
 - making a copy, 113
 - move a file, 466
 - move file, 363
 - plot files, 596
 - plot text file, 428
 - print or plot to a file, 436
 - print text files, 446, 597
 - put text file into another file, 449
 - read 32-bit data files into VNMR, 541
 - read Bruker data files from tape, 461
 - remove old files from experiment, 99
 - rename a file, 466
 - rename file, 363
 - retrieve FIDs from file, 481
 - retrieve parameters from file, 482
 - retrieve shim coil settings from file, 483
 - return information from files display, 221
 - save FIDs in experiment, 551
 - save parameters from experiment, 554
 - save parameters from tree to a file, 234
 - save shim coil settings to a file, 555
 - text files display in graphics window, 195
 - transfer file from remote source, 203
 - transfer files to remote destination, 204
 - write formatted text to a file, 615
- Files menu system
 - default directory, 135
- filter bandwidth, 214
- filter delays, 50
- filter diagonalization method (FDM), 216
- filtdb directory, 222
- Find Correlation button, 37
- Find gzlvl1/gzwin button, 278
- Find gzwin button, 278
- fine attenuator, 578
- fine attenuator configuration, 213
- fine attenuator control, 153, 175
- Fine Attenuator label, 104, 105, 175, 214, 578
- fine power attenuator, 175, 176
- finite impulse response (FIR) coefficients, 222
- first delay in pulse sequence, 123
- first point multiplier, 232
- first pulse width, 381
- first-order baseline correction, 573
- first-order phase, 330
 - make zero, 93
- first-order phase correction, 126
- first-point multiplier, 97
- fitspec.data file, 223
- fitspec.inpar file, 223, 503
- fitspec.outpar file, 223, 503, 525
- fitting arrayed imaging data, 300, 560, 561
- flag-type parameter, 116
- flashc command, 225
- Flexible Data Format (FDF), 215, 553, 555
- flip angle
 - list, 227
 - set rf power levels, 502
- Flip button, 227
- flip time, 381, 450
- FLIPFLOP pulse sequence, 227
- flow encoding gradient level, 259
- fluorine
 - automated acquisition, 212
 - process 1D spectra, 212
- fluorine 1D experiment, getting parameters for, 67
- fluorine parameter set, getting, 67
- fluorine spectrum, setting up parameters for, 228
- fm-fm mode decoupling, 155
- fm-fm modulation (swept-square wave), 160
- folding-in problem, 214
- foreground processing, 591
- formatted text writing to a device, 614
- formatting real number as a string, 231
- Fourier number, 228, 229
- Fourier number scaled value of an integral, 303
- Fourier number scaled volume of a peak, 303
- Fourier transform
 - 1D data, 234, 610
 - 2D data, 236, 238, 610
 - 3D FID into 3D data, 242
 - mathematics, 352
 - phase-sensitive data, 237, 240
 - processing mode for 2D data, 430
- fourth decoupler
 - adjust tip angle resolution time, 158
 - decoupler mode, 155
 - frequency, 141
 - frequency offset control, 165
 - homodecoupling control, 290
 - modulation frequency, 156
 - modulation mode, 161
 - nucleus lookup, 163
 - power level with linear amplifier, 175
 - tip angle resolution, 179
- fp.out file, 108, 136, 231, 560, 561
- frequency limits of region, 256
- frequency of a line, 255
- frequency of decoupler, 141
- frequency of lock, 326
- frequency of NMR resonance offset, 468
- frequency of rf channels, 502

- frequency offset array for decoupler, 492, 493
 - frequency offset for decoupler, 164, 165
 - frequency offset for observe transmitter, 575
 - frequency offset of cursor, 376
 - Frequency Overrange label, 104, 379
 - frequency referencing
 - 2nd evolution dimension, 516
 - evolution dimension, 516
 - proton spectra, 515
 - frequency referencing, see reference line
 - frequency scale dimension adjustment, 491
 - frequency shift of fn spectrum, 340
 - frequency shift of fn1 spectrum, 340
 - frequency shift of fn2 spectrum, 341
 - frequency synthesizer latching, 314
 - frequency synthesizer overrange, 379
 - frequency synthesizer value, 447
 - frequency-independent phase, 408
 - frequency-independent term, 56
 - frequency-shifted quadrature detection, 188, 234
 - frequency-type parameter, 116
 - ft3d call name, 311
 - Full Analysis button, 73
 - full display, 620
 - Full Screen button, 246
 - full screen display limits, 246
 - full screen graphics window, 314
 - Full with Traces button, 247
 - full-band amplifier, 52
 - full-width at half-height (FWHH), 320
- G**
- gain of receiver, 249
 - gap between lines in spectrum, 250
 - GARP decoupling sequence, 155, 157, 160
 - gating time for receiver, 118, 479, 480
 - gauss/cm, converting to DAC value, 220
 - Gaussian apodization constant, 619
 - Gaussian fraction, 187
 - Gaussian fraction for lineshape, 503
 - Gaussian function, 258, 259
 - Gaussian function shift, 619
 - Gaussian lineshape, 223
 - Gaussian low-pass filter, 221
 - Gaussian shift constant, 259, 260
 - Gaussian time constant, see Gaussian function
 - Gaussian window function, 250
 - gcoil parameter, 251
 - gCOSY experiment, changing parameters for, 252
 - GCU (gradient compensation unit), 271
 - GEMINI 2000
 - broadband channel tuning, 91
 - carbon channel tuning, 119
 - console type, 107
 - Ethernet interface, 512
 - lock channel tuning, 196
 - probe tuning mode, 583
 - proton channel tuning, 294
 - spin hardware, 536
 - tune mode, 508
 - Gemini systems
 - convert data to VNMR, 109
 - convert files to UNIX format, 602
 - decompose files to UNIX files, 134
 - list contents of directory, 200
 - read tape, 563
 - UNIX text files conversion to, 586
 - general setup for 2D experiments, 498
 - generalized curve fitting to data, 207
 - generic automatic processing, 441
 - get carbon parameter set, 66
 - gHMBC experiment, changing parameters for, 260
 - gHMQC experiment, setting up parameters for, 260
 - gHMQCTOXY experiment, changing parameters for, 261
 - gHSQC experiment, changing parameters for, 262
 - gHSQCTOXY experiment, changing parameters for, 262
 - Gilson Control window, 263
 - gilson.conf file, 201
 - GLIDE administration tool, 249
 - GLIDE button, 264
 - GLIDE calibration parameters, adding, 72
 - GLIDE chain
 - adding experiment to, 70
 - removing calibration routine from, 72
 - removing experiment from, 72
 - GLIDE experiment
 - setting up from command line, 504
 - GLIDE group, adding users to, 504
 - GLIDE group, administrating, 504
 - GLIDE parameters, adding to parameter sets, 78
 - GLIDE selection, showing, 73
 - GLIDE-created macros
 - making a directory for, 75
 - storing, 75
 - GLIDE-run data, storing, 75
 - GLIDE-selected plot options, checking for, 77
 - global file, 116, 118
 - update after VNMR install, 588
 - global parameter tree
 - save parameters, 488
 - global-type parameter tree, 116
 - gmapz pulse sequence, 266, 267
 - gmapz.par file, 266
 - Go button, 268
 - Go,Wft button, 248
 - grad_cw_coef parameter, 405
 - grad_p_coef parameter, 405
 - grad_p1 array, 406
 - gradient
 - coil, 251
 - phase encode dephasing, 270
 - gradient amplifier installation tests, 566
 - Gradient Autoshim on Z button, 264
 - gradient autoshimming, 264
 - gradient axis, 270
 - gradient calibration constant, 250, 277, 503
 - gradient calibration constant retrieval, 254
 - gradient calibration parameters
 - boresize, 88
 - gradient calibration pulse sequence, 442
 - gradient calibration value, 197
 - gradient coil configuration, 559
 - gradient coil configuration file, 105
 - gradient coil updating, 588

Index

- gradient COSY pulse sequence, 252
- gradient evaluation pulse sequence, 248
- gradient level trim, 276
- gradient levels, 505
- gradient list, 544
- gradient map generation, 68, 69
- gradient map generation, automatic, 68
- gradient phase encoding increment, 269
- gradient refocused high-speed imaging sequences, 630
- gradient rise rate, 273, 581
- gradient set
 - internal usable diameter, 88
- gradient shape, 269
- gradient shimming
 - display menu, 265
 - gradient shimming menu, 266
 - map shims, 265
 - pulsed field gradient strength, 278
 - set parameters, 265
 - spectral width percentage, 278
 - start acquisition, 265
 - start gradient autoshimming, 264
 - z-axis shims number, 278
- Gradient Shimming System menu, 265
- gradient spoiling time, 582
- gradient step size, 270
- gradient strength, 273
 - maximum value, 267
 - voxel selection, 276
 - X, Y, Z gradients, 277
- gradient strength for each axis, 277
- gradient strengths calibration for PGE, 405
- gradient table generation, 117
- gradient total limit, 276
- gradients for X, Y, and Z axes, 270
- Gradients label, 104
- gradtables directory, 117, 504, 581
- graphics window, 530
 - display message with large characters, 85
 - display status, 271
 - display text file, 195
 - draw box, 88
 - flipping text window in and out, 227
 - full screen, 314
 - write formatted text to screen, 614
- Graphics Window colors, 499
- graphics window, dividing into rows and columns, 505
- GraphOn terminal window clearing, 99
- gray scale contrast adjustment, 272
- gray scale display adjustment, 271
- gray scale image display, 297
- gray scale image plot, 298
- grid lines over 2D plot, 421
- grid on a 2D display, 272
- gripper abort, 43
- Group A parameters, 146
- group of parameters in tree, 506
- H**
- H1.par file, 519
- Hardware button, 106
- hardware Ethernet address display, 197
- hardware shimming
 - list of shims, 283
- hardware shims, 460, 499
- hardware status of console, 296
- hardware values in acquisition system, 507
- hardware Z1 shimming, 283
- HCCHTOCSY sequence, 281
- HCPF experiment, 68
- height of peak, 231
- Help button, 284
- help directory, 284
- help information, 284
- HET2DJ button, 284
- HET2DJ pulse sequence, 284
- HETCOR acquisition, 282
- HETCOR button, 284
- HETCOR experiment, changing parameters for, 284
- HETCOR pulse sequence, 284, 423
- HETCORCP1 pulse sequence, 285
- HETCORPS pulse sequence, 285
- heteronuclear 2D-J experiment, 284
- heteronuclear chemical shift correlation, 284
- heteronuclear decoupling, 148
- heteronuclear J-resolved 2D spectra, 421
- heteronuclear multiple-quantum coherence, 286, 287
- heteronuclear Overbodenhausen experiment, 291
- Hewlett-Packard plotter pens, 513
- Hewlett-Packard plotters, 55, 291, 621, 624
- hiding a command, 285
- high signal handling, 435
- high-power pulse widths, calibrating, 66
- Hilbert transform algorithm, 218
- HMBC experiment, changing parameters for, 285
- HMBC sequence, 423
- HMQC experiment, changing parameters for, 286
- HMQC phase-sensitive PFG pulse sequence, 261
- HMQC pulse sequence, 260, 286, 423
- HMQCR pulse sequence, 287
- HMQC-TOCSY 3D pulse sequence, 288
- HMQCTOCSY sequence, 287
- HMQCTOXY experiment, changing parameters for, 287
- HOHAHA experiment, 575
- HOM2DJ button, 288
- HOM2DJ pulse sequence, 288
- Homo Dec. Offset label, 106, 283
- HOMODEC experiment, changing parameters for, 288
- Homodecoupler label, 105, 106, 289
- homodecoupler power level, 153
- homodecoupling control, 289, 290
- homonuclear correlation, 113, 574
- homonuclear decoupler offset configuration, 283
- homonuclear decoupler present, 289
- homonuclear decoupling, 148
- Homonuclear Hartmann-Hahn experiment, 575
- homonuclear J-resolved 2D experiment, 288
- homonuclear J-resolved 2D spectra, 422
- homospoil, 291
 - pulse length, 294
 - pulses, 291

- horizontal LC axis, 126, 394
 - horizontal offset, 288
 - horizontal projection of trace, 130
 - horizontally stacked spectra, 192
 - host computer
 - serial port connection to changer, 530
 - host computer connection to Ethernet, 502
 - host computer disconnect from Ethernet, 512
 - host disk errors, 44
 - hostname.le0 file, 502
 - hosts.3D file, 243
 - Hoult setting for final pulse times, 290
 - HP Params button, 291
 - HSQC experiment, changing parameters for, 292
 - HSQC pulse sequence, 261, 291
 - HSQC-TOCSY 3D pulse sequence, 293
 - hypercomplex points to left-shift interferogram, 339
- I**
- I1 and I2 values, 203
 - idle mode for amplifiers, 51
 - IF Frequency label, 104
 - imag.c file, 277
 - image
 - annotate display, 298
 - center on the readout axis, 437
 - coordinate display information, 325
 - field of view size for readout, 338
 - position on 2D phase encode axis, 433
 - Image button, 298
 - ImageBrowser application, 90
 - ImageBrowser FDF files, 552
 - ImageBrowser program, 553, 556
 - ImageCalc button, 299
 - images
 - calculate 2D phasefiles, 298
 - display multiple images, 160
 - generate as ImageBrowser files, 552
 - generate from experiment, 553
 - save as FDF files, 553
 - save as ImageBrowser files, 552
 - imaginary part of FID, 140
 - imaging
 - application mode, 58
 - attenuator, 94
 - echo time, 566
 - intensity of excitation pulse, 577
 - macros and menus, 57
 - readout position, 360
 - imaging experiments
 - repetition time, 578
 - Imaging Gradient Coil label, 105
 - imaging gradients setup, 301
 - imconi macro, 300
 - inactive parameter, 376
 - INADEQUATE data about 2-quantum axis, 229
 - INADEQUATE pulse sequence, 301
 - INADQT button, 301
 - Incredible Natural Abundance Double-Quantum
 - Transfer Experiment, 301
 - incremented delay for pulse sequence, 124, 125
 - index of experimental frequency of transition, 100
 - indirectly detected axis, 83
 - INEPT button, 302
 - INEPT pulse sequence, 302
 - info directory, 498
 - info_# file, 405
 - inline DSP, 187
 - Input board spectral width, 352
 - Insensitive Nuclei Enhanced by Polarization
 - Transfer, 302
 - inserting a sample, 296, 304
 - inset spectrum, 304
 - integer-type parameter, 116
 - integral
 - display, 180
 - display mode, 305
 - integral value, 303
 - largest value in region, 304
 - normalization scale, 303
 - offset, 306
 - regions, 291
 - reset points, 626
 - scale, 306
 - set value, 509
 - integral amplitudes display, 170
 - integral amplitudes plot, 413
 - integral scale adjustment, 306, 307
 - integrals
 - clear reset points, 121
 - data truncation limit, 121
 - display in normalized format, 153
 - display list, 151
 - find integral values, 367
 - find normalized integral values, 369
 - reset point amplitudes, 318
 - reset point frequencies, 319
 - Integrals button, 151
 - integration, 1D spectrum, 305
 - intensity of spectrum at a point, 350
 - intensity threshold, 546
 - interactive acquisition display, 38
 - Interactive button, 181
 - Interactive Mode button, 46
 - interactive phasing, 180
 - interactive probe tuning, 582
 - interactive UNIX shell, 523
 - Interactive View button, 416
 - interferogram coefficients, 213
 - interferograms
 - first-point multiplier, 232
 - start of display, 521
 - type of data processing, 439
 - weighting interactively, 619
 - width of display, 609
 - zero-order phasing constant, 411
 - interlock to control lock level and spin speed, 301
 - Internet address, 40, 41
 - inverse cosine calculation, 37
 - inverse Fourier transform, 235
 - inverse sine, 61
 - inverse tangent, 63
 - inversion prepulse recovery time, 571
 - inversion pulse intensity, 578
 - inversion pulse length, 412
 - inversion pulse shape, 413

Index

- inversion recovery experiments, 571
 - inversion recovery mode, 306
 - invert image, 299
 - iterated parameters list, 307
 - iterations in an iterative simulation, 367
- J**
- J-correlation experiment, 574
 - joining
 - an existing experiment, 308
 - joint arrays, 61
 - J-resolved 2D spectrum, 230
 - jump-and-return sequence, 310
 - JUMPRET sequence, 310
- K**
- keyboard entries record, 462
 - keyboard focus to input window, 229
 - keyboard input into variables, 302
 - kinetics analysis, 54, 208, 312, 313, 403
- L**
- label a stacked spectra display, 193
 - labeling an image display, 298
 - laboratory frame Overhauser experiment, 371
 - Large button, 314
 - lastlk file, 314
 - latching capabilities of frequency synthesizer, 314
 - Latching label, 104, 314
 - LC axis, 126, 394
 - LC-NMR
 - 2D acquisition parameters, 316
 - add series of FIDs, 45
 - create parameters, 390
 - create pseudo-2D dataset, 264
 - delay for trigger, 196
 - display horizontal LC axis, 126
 - display LC-NMR parameters, 147
 - general 2D experiment setup, 317
 - set up parameters for LC-NMR sequences, 317
 - set up pulse sequence for LC-NMR run, 316
 - set up scout run, 516
 - TOCSY sequence, 316
 - least-squares curve fitting, 53, 207
 - Left button, 317
 - left half of screen display limits, 317
 - left-shift FID to time-domain cursor, 573
 - left-shift ni interferogram, 339
 - left-shift ni2 interferogram, 339
 - left-shift np FID, 338
 - leg relay control, 318
 - lfs (low-frequency suppression) option, 389
 - Library button, 210
 - limits for scales in regression, 490
 - limits of parameter in a tree, 509
 - limNET nodes database, 163
 - line amplitudes list, 323
 - line assignments for spin simulation, 98, 151
 - line broadening, 315
 - line broadening factor, 619
 - line drawing between points, 177
 - line drawing capability, 359, 402
 - line frequencies, 323, 528
 - line frequencies and intensities
 - display list, 152
 - find values, 368
 - line in a text file, 328
 - line list plotting, 424
 - line listing intensity and frequency, 255
 - line narrowing sequence, 361
 - linear amplifiers, 52, 57, 105
 - decoupler power level, 173, 174
 - power level, 576
 - linear curve fitting, 54
 - linear fitting to data, 208
 - linear modulator power, 176, 578
 - linear monotonic order data, 562
 - linear prediction
 - algorithm, 330, 331
 - algorithm data extension, 335, 336
 - arraying parameters, 335
 - calculation start point, 547, 548
 - coefficients to calculate, 333, 334
 - create parameters, 391
 - data extension, 332, 333
 - data extension start point, 546, 547
 - multiple operations, 335
 - number of data points, 334
 - output spectrum, 337
 - parameter creation, 47
 - parameter group, 147
 - print output, 336, 337
 - printout, 335
 - type of data processing, 438, 439, 440
 - linear scaling of image intensity, 297
 - linearly spaced array values, 60
 - line-narrowing multiple-pulse, 90
 - Lines button, 152
 - lines of text, look up from a text file, 328
 - linshape modification, 503
 - linewidth for spin simulation, 529
 - linewidth measurement, 178
 - load time counter, 622
 - local file transfer to remote host, 204
 - local host name display, 197
 - local oscillator (L.O.), 328
 - localized spectroscopy, 598
 - localized spectroscopy experiments
 - repetition time, 578
 - location of sample in tray, 324
 - location to start a line, 359
 - lock
 - acquisition time constant, 326
 - automatic control, 51
 - automatic phase adjustment, 325
 - capture, 51
 - digital lock display, 461
 - frequency on UNITYplus, 510
 - gain value, 327
 - interactive, 38
 - lock frequency adjustment, 326
 - lock parameters setup, 510

- loop time constant, 328
 - phase value, 327
 - power value, 327
 - read current lock level, 461
 - remove inactive lock, 587
 - solvent selection, 531
 - solvent used, 314
 - time constant, 327
 - transmitter thumbwheel switches, 326
 - tuning on GEMINI 2000, 196
 - lock file, 591
 - lock frequency
 - track changes, 319
 - Lock Frequency label, 104, 105, 106, 326
 - lock level display, 39, 40
 - lock level interlock, 301
 - log file, 211
 - log file for experiment, 210
 - logarithm of a number, 323
 - login macro, 88
 - loop size of fifo, 220
 - looping control for real-time arrays, 511
 - looping processes control, 496
 - Lorentzian lineshape, 223, 503
 - low signal handling, 435
 - low-band amplifier, 52
 - lowercase format of string, 231
 - low-pass digital filter, 389
 - low-pass Gaussian filter, 221
 - lpanalyz.out.# file, 335, 336, 337
- M**
- macdir entries, removing, 78
 - maclib directory, 88, 115, 135, 205, 344, 462
 - macro, writing a string to a, 79
 - macros
 - activated by VNMR bootup, 88
 - automatic execution, 514
 - before experiment starts, 248
 - change action of abort command, 35
 - check for existence, 204
 - copy system macro to become user macro, 345
 - copy user macro file, 342
 - create without text editor, 115
 - delete user macro, 135
 - display dialog box, 148
 - display system macro, 344
 - display user macro in text window, 342
 - display which macro is used, 613
 - edit online description, 349
 - edit user macro with vi editor, 346
 - edit with macro editor, 343
 - hide command with same name, 285
 - keyboard entries, 462
 - list system macros, 345
 - list user macro file names, 343
 - load macro into memory, 343
 - name of invoking macro, 342
 - name storage for macros, 364
 - online description, 349
 - real-value storage parameters, 458
 - remove macro from memory, 449
 - remove system macro, 345
 - remove user macro from directory, 344
 - restore normal abort function, 36
 - return values to calling macro, 468
 - terminate calling macro, 35
 - terminate execution, 468
 - user's macro directory, 342
 - macros, deleting, 77
 - macros, saving in GLIDE run, 75
 - magic angle spinning, see MAS
 - Magnet Leg Driver Board Configuration ID, 435
 - magnet leg relay control, 318
 - magnetization recovery, 123
 - main magnetic field strength, 85
 - Make Shimmap button, 265
 - makeuser command, 588
 - manual directory, 349
 - map shims, 265
 - MARK button, 350, 351
 - Mark button, 416
 - mark output, 588
 - mark1d.out file, 62, 350, 535, 589
 - mark2d.out file, 89, 350, 416
 - MAS cross-polarization spin-lock contact time, 108
 - MAS spinning speed, 540
 - Max. Decoupler label, 106, 173
 - Max. Narrowband Width label, 103, 168, 353
 - Max. Spectral Width label, 103
 - Maximum DMF label, 103
 - maximum frequency of any transition, 530
 - Maximum gradient DAC value, 270
 - maximum gradient strength for each axis, 277
 - maximum limits on a parameter, 510
 - maximum parameter value array, 391
 - maximum transients accumulated, 363
 - mean of the data in regression.inp file, 431
 - measured line frequencies, 528
 - measured line frequencies array, 535
 - memory buffers, write to disk, 228
 - memory increased by removing macros, 449
 - memory map FID file, close, 355
 - memory map open FID file, 356
 - memory usage statistics, 362
 - Menu On button, 353
 - menulib directory, 221, 353
 - menus
 - button command string, 363
 - change status of menu system, 353
 - edit menu with vi editor, 354
 - label for button, 358
 - menu displayed by Return button, 314
 - path to user's menu directory, 353
 - return currently active menu, 365
 - select menu without activation, 365
 - MERCURY
 - console type, 107
 - spin hardware, 536
 - MERCURY series
 - broadband channel tuning, 91
 - MERCURY-VX
 - console type, 107
 - probe tuning mode, 583
 - spin hardware, 536
 - message

Index

- confirm using mouse, 107
- display with large characters, 85
- messages from send2Vnmr, 319
- method string, 545
- microimaging
 - center sequence calibration, 499
 - ECC tool window, 198
 - eddy current compensation analysis, 198
 - eddy current compensation data, 197
 - field of view for phase encode, 332
 - generate transverse magnetization, 630
 - gradient amplifier installation tests, 566
 - gradient calibration constant, 503
 - intensity of an inversion pulse, 578
 - inversion pulse shape, 413
 - move data into reference table, 197
 - orientation of slice plane, 377
 - phase encoding gradient increment, 269
 - refocusing pulse shape, 451
 - shape of excitation pulse, 381
 - shaped gradient tests, 274
 - update eddy current settings, 199
- Minimum button, 538
- minimum frequency of any transition, 530
- minimum intensity threshold, 546
- minimum limits on a parameter, 510
- minimum of two spectra, 538
- minimum parameter value array, 392
- MLEV-16 decoupling sequence, 155, 157, 160
- mode for n-dimensional data display, 579
- modulation frequency of decoupler, 155, 156
- modulation mode for decoupler, 160, 161
- mopos parameter, 579
- mouse
 - confirming a message, 107
- mouse position, reporting, 263
- moving
 - files, 363, 466
 - parameters between experiments, 361
 - spectral window according to cursors, 360
 - transmitter offset, 361
- MQCOSY pulse sequence, 361
- MREV8 pulse sequence, 121, 361
- multidimensional data display mode, 579
- multiecho sequences, 364
- multihost processing, 244, 245
- multiple image display, 160, 185
- multiple receivers
 - add transformed data files with weighting, 479
 - combine data, 49
 - number currently active, 364
 - number of receivers, 374
 - set filter bandwidth, 362
 - set gain, 362
 - weighting for different receivers, 459
 - which receivers to use, 459
- multiple-pulse line narrowing, 90, 361
- multiple-quantum filtered COSY, 361, 574
- multipulse experiments
 - f_1 scaling factor, 491
 - scaling factor, 490
- multislice experiments, 552
 - data set, 185
 - spin-echo imaging sequence, 495

N

- name of pulse sequence, 496
- name storage for macros, 364
- natural logarithm of number, 323
- negative intensities, setting 2D, 630
- newlink AuCALch3i, 65
- ni interferogram
 - number of complex point to left shift, 339
 - type of data processing, 439
- ni2 interferogram
 - type of data processing, 439
 - zero-order phasing constant, 411
- NMR resonance offset frequency, 468
- node files, 203, 204
- nodes file, 203, 204
- NOE difference experiment, 370
- NOE experiment, 121
- NOESY
 - parameter set, 267
 - plotting spectra, 419
 - pulse sequence, 370
- NOESY button, 371
- NOESY experiment, changing parameters for, 370
- NOESY1D experiment, changing parameters for, 371
- noise level estimate, 257
- noise level in spectrum, 372
- noise level of FID, 371
- noise mode decoupling, 155
- noise modulation, 160
- noise multiplier, 371
- normalized integral amplitudes, 170
- normalized integral amplitudes plot, 413
- normalized integral values, 369
- normalized integrals display list, 153
- normalized intensity mode, 369
- normalizing homodecoupler power output, 283
- nt array, 406
- N-type display, 374
- nucleus for decoupler, 162, 163
- nucleus for observe transmitter, 573
- nucleus selection, 519
- nucleus to add to probe file, 46
- Nucleus,Solvent button, 519
- nuctables directory, 162
- number of increments of evolution time, 366, 367
- Number of RF Channels label, 104, 375
- Nyquist frequency, 297

O

- object library for PSG, 444
- oblique imaging capability, 495
- observe nucleus transmitter frequency, 522
- observe receiver, 214
- Observe Receiver board, 459
- offset
 - horizontal, 288
 - integral, 306
 - vertical, 597
- offset frequency
 - calculate for nucleus and ppm, 512
- online description of command or macro, 349

- edit description, 349
- open reel tape, 563
- oph real-time variable, 113
- order of parameter array, 60
- orientation of slice plane, 377
- out.c file, 111
- overhead delay between FIDs, 123
- overrange of frequency synthesizer, 379
- oversampling
 - bandpass filter offset, 378
 - bandwidth, 378
 - factor for acquisition, 379
 - filter type, 378
 - number of coefficients, 377
 - parameter creation, 47, 392
 - setting parameters, 359
- Oxford shim supply, 524
- Oxford VT controller, 352
- Oxford-Sorenson VT controller, 352

- P**
- Page button, 385
- page change on plotter, 385
- par directory, 519
- parameter array, 499
- parameter directory
 - delete, 135
- parameter list
 - parameter names and values, 386
 - plotting, 432
 - power level parameters, 453
 - pulse length parameters, 424
 - pulse template parameters, 394
- parameter screens display menu, 147
- parameter set, converting to APT experiment, 58
- parameter sets
 - correct saved parameter sets, 587
 - file name of retrieved set, 220
 - update all sets in directory, 587
- parameter style, 392
- parameter tree
 - copy parameters of group, 274
 - create new parameter, 115
 - destroy parameters of a group, 139
 - display parameters with attributes, 150
 - limits of parameter, 509
 - load parameters from file into a tree, 233
 - make parameter active, 376
 - make parameter inactive, 376
 - prune extra parameters, 443
 - remove a parameter, 138
 - set Dgroup of a parameter, 501
 - set group of parameter, 506
 - set values of string parameter, 501
 - systemglobal-type tree, 102
 - types of trees, 116
 - value of parameter, 520
 - write parameters to file, 234
- parameters
 - 3rd rf/3D parameter group, 145
 - 4th rf channel parameter display group, 145
 - acquisition/processing group, 144
 - add for FID display, 219
 - add parameter to probe file, 48
 - add to current experiment, 46
 - adjust values from setup macros, 225
 - adjusting, 309
 - adjusting plot, 309
 - arrayed for acquisition, 126
 - arraying order and precedence, 60
 - automation parameter group, 147
 - basic experiment setup, 519
 - boxed for plotting, 89
 - center sequence calibration, 499
 - change type, 518
 - check existence, 204
 - chemist-style, 89
 - convert to PGE, 405
 - copy between trees, 274
 - correct limits and step sizes, 390
 - correct parameter characteristics, 224
 - create 2D parameters, 386
 - create 3D parameters, 387
 - create 4D acquisition parameters, 387
 - create for fourth channel, 225
 - create for linear prediction, 391
 - create for third rf channel, 224
 - create LC-NMR parameters, 390
 - create new parameter in tree, 115
 - create oversampling parameters, 392
 - create parameters for 2D peak picking, 390
 - create solvent subtractions parameters, 389
 - customize parameter sets, 589
 - destroy a parameter, 138
 - destroy parameters of a group, 139
 - display control, 55
 - display from tree with attributes, 150
 - display parameters group, 145
 - display templates for third rf channel, 387
 - display values in text window, 198
 - displaying value, 456
 - downsampling, 388
 - edit parameter and its attributes, 387, 388
 - full recall of display parameters, 233
 - full spectrum display, 212
 - get value, 257
 - gradient shimming, 265
 - limits of parameter in tree, 509
 - linear prediction parameter group, 147
 - linearly spaced steps, 60
 - list to be iterated, 307
 - lock parameters setup, 510
 - make parameter active, 376
 - maximum values, 391
 - minimum values, 392
 - move between experiments, 361
 - move display parameters between experiments, 353
 - move parameters to target experiment, 579
 - plot list automatically, 55
 - plot on special chart paper, 291
 - prepare for acqi, 258
 - print all, 55
 - protection mode, 514
 - prune parameters from tree, 443
 - pseudo-echo weighting, 444

Index

- read from file and load into tree, 233
- recall display parameter set, 458
- recalling stored parameters, 79
- reset after partial 3D FT, 467
- resolution enhancement, 468
- restoring current experiment, 309
- retrieve from experiment subfile, 484
- retrieve from file, 482
- retrieve individual parameters from file, 484
- retrieve parameter from probe file, 255
- save display parameters as set, 486
- save from experiment, 554
- save from tree to file, 234
- save parameters from global tree, 488
- set group of parameter in tree, 506
- set up for pulse sequences, 444
- set up standard two-pulse sequence, 486
- set voxel parameters, 598
- shaped gradients testing, 274
- shims parameter group, 147
- sine window function, 527
- sinebell weighting, 527
- sine-squared window function parameter values, 528
- spin simulation parameter arrays, 151
- spin simulation parameter group, 146
- spin system parameters to iterate, 302
- step size values, 392
- system configuration, 102
- test state of parameter, 376
- turn off active parameter, 376
- types of values, 116
- unit conversion, 585
- update after new VNMR install, 588
- value of parameter in tree, 520
- VAST experiment parameter setup, 590
- version of parameter set, 393
- Params button, 432
- parlib file, 445
- paths
 - 2D planes from a 3D data set, 393
 - current working directory, 451
 - user's macro directory, 342
 - user's menu directory, 353
 - user's shim settings directory, 525
 - VNMR system directory, 559
 - VNMR user directory, 589
- Pbox
 - add parameter definition to pbox.inp file, 398
 - assign Pbox calibration data, 454
 - convert to Pbox default units, 399
 - converts to default units, 399
 - create Pbox shape file, 114
 - create shape definition, 455
 - define excitation band, 395, 494
 - define excitation band for solvent suppression, 396
 - display interactive modulation pattern, 184
 - display interactive pulse shape, 184
 - display last generated pulse shape, 184
 - display modulation pattern, 184
 - display pulse shape, 184
 - extract dmf value, 396
 - extract dres value, 396
 - extract fine power level, 397, 398
 - extract name of last shape, 396
 - extract power level, 397
 - extract pulse length, 397
 - generate a single-band shapefile, 454
 - open shape definition file, 377
 - plot modulation pattern, 445
 - plot pulse excitation profile, 434
 - plot pulse shape, 445
 - plot the last created pulse shape, 445
 - print pulse header, 433
 - reset temporary pbox/Vnmr variables, 398
 - simulate Bloch profile for a shaped pulse, 455
 - write a wave into file, 449
 - write wave definition string, 520
- pbox
 - write wave definition string, 520, 522
- pbox shape file, 114
- pcss.outpar storage file, 165, 401
- peak frequencies display, 169
- peak frequencies plot, 433
- peak frequencies threshold, 570
- peak height or phase measurement, 231
- peak heights comparison, 50
- peak noise, 257
- peak number, 316
- peak picking, 320, 414
 - diagonal peak threshold, 621
 - parameters creation, 390
 - plot results, 424
- peak printout threshold, 570
- peak search range of data points, 373
- peak truncation in spectra plot, 414
- peak width of solvent resonances, 531
- peak, selecting, 550
- Peaks button, 433
- peaks.bin file, 322
- peak-to-peak noise, 372
- pens
 - maximum number to use, 352
 - on HP plotter, 513
 - selection for drawing, 402
- Performa I, II, III, 105
- Performa modules, 271
- PFG
 - absolute-value MQF COSY pulse sequence, 267
 - absolute-value ROESY pulse sequence, 276
 - amplifiers on/off control, 404
 - eddy current testing, 272
 - gradient calibration constant, 503
 - HMQC phase-sensitive pulse sequence, 261
 - HMQC pulse sequence, 260
 - HSQC pulse sequence, 261
 - NOESY parameter set, 267
 - selective excitation pulse sequence, 495
 - sequence for PFG testing, 382
 - TNNOESY pulse sequence, 276
- pge file, 405
- PGE pulse sequence
 - calibrate gradient strengths, 405, 406
 - extract data, 405
 - parameter conversion, 405
 - plot results, 406

- print results, 406
- processing of data, 406
- phantom for gradient calibration, 503
- phase angle display mode, 158, 383
- phase correction applied to interferogram, 126
- phase cycling type, 409
- phase encode
 - gradient levels, 505
 - image center position, 433
 - pulse length, 576
- phase encode dephasing gradient, EPI sequence, 270
- phase encode gradient increment multiplier, 270
- phase encoding gradient increment, 269
- phase encoding gradient pulse length, 576
- phase file
 - display in experiment, 132
- phase of first pulse, 409
- phase of peaks, 231, 408
- phase parameters
 - automatic calculation of, 55
- phase-correction angles, 330, 480, 481
- phased data display mode, 158
- phased spectra display mode, 383, 407, 408
- phase-encode axis, 332
- phasefiles, 298, 299, 300, 554
 - calculate 2D phasefiles, 298
 - copy stored phasefile, 483
 - transform and save images, 348
- phase-sensitive 2D transformation, 330
- phase-sensitive COSY pulse sequence, 113
- phase-sensitive data, 237, 240, 610, 612
- phasing
 - automatic, 55
 - control update region, 410
- phosphorus
 - acquisition, 382
 - processing, 382
 - spectrum plotting, 428
- phosphorus 1D experiment, getting parameters for, 70
- phosphorus spectrum, setting up parameters for, 411
- pi/3 shifted sinebell squared window function, 412
- pi/4 shifted sinebell squared window function, 412
- pixel size calculation, 467
- pl2dj macro, 425
- planes
 - extract from 3D spectral data, 255
- planes directory, 300, 483, 554
- planlock parameter, 416, 543
- planner lock, 417
- planning a target scan, 415
- Plot button, 415, 420
- Plot Design, joining, 308, 309
- plot parameters
 - adjusting, 309
- plot queue
 - show jobs in queue, 526
 - stop jobs and remove from queue, 311
- plots, 406
- plotter
 - characteristics, 513
 - device setup, 426
 - display mode, 428
 - Hewlett-Packard, 621
 - maximum number of pens, 352, 513
 - maximum width of plotting area, 605
 - plot contours, 400
 - reinitializing, 311
 - resolution of points drawn, 434
 - show plot queue, 526
 - stopping plot jobs, 311
 - submit plot and change plotter page, 385
 - write formatted text to plotter, 614
- plotter units
 - converted from Hz or ppm, 294
- Plotters color, 499
- plotting
 - 2D contour plots for 3D planes, 427
 - 2D displayed resolution, 61
 - 2D peak picking results, 424
 - 2D spectra in whitewash mode, 415
 - adjust plot parameters, 222
 - arrayed 1D spectra, 418
 - ATP-type spectra, 417
 - axis label units, 82
 - boxed parameters, 89
 - carbon spectrum, 418
 - color assignments, 100, 513
 - contour plot with colors, 400
 - contours display, 168, 169
 - COSY data set automatically, 384
 - COSY spectra, 419
 - deconvolution analysis, 420
 - DEPT analysis, 385
 - DEPT data, 419
 - display same as plot, 620
 - draw box, 88
 - exponential curves, 403
 - FIDs, 420
 - FIDs in whitewash mode, 404
 - files, 596
 - formatted text, 614
 - grid on 2D plot, 421
 - heteronuclear J-resolved 2D spectra, 421
 - homonuclear J-resolved 2D spectra, 422
 - horizontal LC axis, 394
 - limit to center of page, 96
 - line list, 424
 - NOESY spectra, 419
 - non-arrayed 1D spectra, 425
 - noninteractive gray scale image, 298
 - parameter list, 386, 432
 - parameter list on special paper, 291
 - parameters automatically, 55
 - peak frequencies over spectrum, 433
 - PGE calculated results, 406
 - phosphorus spectrum, 427
 - plot a title, 572
 - plotter characteristics, 513
 - polynomial curves, 403
 - proton spectrum, 421
 - pulse sequence, 435
 - scale below spectrum or FID, 443
 - set full page plot with room for traces, 247
 - set limits for full page plot, 246
 - spectra, 414
 - spectra automatically, 425
 - spectra in whitewash mode, 430

Index

- spectral expansion, 49
- start of plotting position, 490
- start of plotting position in second direction, 490
- store parameter style command, 392
- text file, 428
- X,H-correlation 2D spectrum, 423
- plotting area, see chart
- plotting scaling factor, 294
- pointer position, locating, 263
- polarization transfer experiments, 138
- polynomial curve, 207
- polynomial curves display, 209
- polynomial curves plot, 403
- polynomial fitting of baseline, 86
- Postscript printer, 621
- post-trigger delay, 288
- powder pattern
 - finding the center, 538
- power data display mode, 158
- power level calibration, 577
- power level for decoupler with deuterium decoupler, 175
- power level for decoupler with linear amplifier, 173, 174
- power level of transmitter, 576
- power spectra display mode, 451, 452
- power, setting, 513
- powers of 2 vertical scale adjustment, 600
- ppm calculations, 463
- ppm of solvent resonances, 531
- preacquisition delay, 384
- preamplifier signal level selection, 435
- precedence of parameter array, 60
- PRESAT sequence, 435
- pre-trigger delay, 460
- print queue
 - show jobs in queue, 526
 - stop print jobs and remove from queue, 312
- printcap entry, 597
- printer
 - device setup, 436
 - linewidth resolution, 434
 - maximum width of chart, 605
 - resolution in dots/mm, 434
 - send text to printer, 437
 - start print operation, 436
 - stopping print jobs, 312
 - type, 597
 - write formatted text on printer, 615
- printing
 - color assignments, 100, 513
 - parameters, 55
 - PGE calculated results, 406
 - probe file after autocalibration, 37
 - starting, 436
 - text file, 446
 - text files, 597
- printing area, see chart
- probe
 - phase glitch removal, 227
 - tuning, 456, 582
 - tuning frequencies, 582
 - tuning mode on GEMINI 2000, 583
 - type, 437
- probe directory, create new, 48
- probe file, 36, 46, 513
 - add parameter, 48
 - retrieve parameter, 255
 - set decoupler parameter values, 501
 - update, 587
- probe file, copying, 71
- probe file, create new, 48
- probe file, make copy, 71
- probe gcal calibration macros, 68
- probe protection control, 438
- probe, copying, 71
- probe, editing, 437, 438
- procdat file, 498
- processed-type parameter tree, 116
- processing
 - 1D carbon spectra, 92
 - 2D spectra, 440
 - 3D data processing information, 498
 - arrayed 1D spectra, 440, 545
 - create 2D parameters, 386
 - create 3D processing parameters, 387
 - DEPT spectra array, 138
 - FIDs automatically, 441
 - fluorine 1D, 212
 - generic automatic, 441
 - interleave FIDs, 296
 - phosphorus 1D spectra, 382
 - proton 1D, 279
 - select 1D experiment for processing, 252
 - selected 2D experiment, 253
 - simple 1D spectra, 439
 - solvent subtraction events, 389
 - using GLIDE windows, 263
- processing mode for 2D data, 430
- processing on FID, 438
- processing on the interferogram, 439
- processing parameters group, 144
- procpa file, 101, 116, 587
- procpa3d file, 498
- procpa3d parameter set, 366, 498
- programmable eddy current compensation file, 402
- programmable filters, 214, 404
- programmable pulse modulation, 160
- project 2D data onto axis, 442
- projection plane, 172
- protection mode of parameter, 514
- proton
 - acquisition, 281, 282
 - automatic acquisition, 279
 - homodecoupler power level, 153
 - pulse power level, 434
 - spectra processing, 279
 - spectra vertical scale adjustment, 601
 - spectrum plotting, 421, 428
- proton 1D experiment, set up parameters for, 68
- proton acquisition, 281, 282
- proton channel
 - tuning on GEMINI 2000, 294
- proton chemical shifts spectrum
 - calculating, 165
 - calculating and showing, 401
 - reducing to a list, 165

proton decoupler
 pulse calibration, 432
 proton decoupler calibrations, 69
 proton frequency configuration, 279
 Proton Frequency label, 103, 105, 106, 279
 proton gradient ratio calibration macros, 67
 proton observe calibration macros, 69
 proton parameter set, getting, 65, 66, 68, 69
 proton spectrum, setting up parameters for, 443
 Pseudo button, 444
 pseudo-2D, 590
 pseudo-2D dataset, 264
 pseudo-echo weighting, 444
 psg directory, 497
 PSG errors, 444
 PSG message, 35
 PSG object library compilation, 444
 psg.error file, 444
 psglib directory, 497
 PTS frequency synthesizer, 104, 447
 P-type diagonal, 230
 P-type double-quantum axis, 229
 pulse amplifier
 mode, 51
 phase glitch removal, 227
 pulse breakthrough effects, 480
 pulse calibration data file
 update and display, 448
 pulse interval time, 453
 pulse length of decoupler, 432
 pulse power for shaped pulse, 448
 pulse power level, 434
 parameter list, 453
 pulse sequence
 compiling, 497
 display diagram, 172
 initiate compilation, 497
 label for screen, 446
 name to be used, 496
 phase-sensitive COSY, 113
 plotting a picture of a sequence, 435
 set up parameters, 444
 setup macro, 269
 Pulse Sequence Controller board, 480
 pulse sequence generation, see PSG
 pulse sequences
 display templates, 518
 pulse template parameter list, 394
 pulse width in degrees, 381, 450
 pulse width length, 450
 pulse width of first pulse, 381
 pulse width optimum value, 203
 pulsecal file, 448, 502
 pulsed field gradient strength, 278
 Pulsed Field Gradients label, 105
 pulse-type parameter, 116
 pulsewidth, setting, 513
 pure absorptive display, 240
 pwwet pulse width, 608
 pwx1 parameter, 454

Q

quadratic fitting to data, 54, 208
 quadrature detection
 frequency shifted, 234
 quadrupole echo pulse sequence, 541
 question mark (?) notation, 456

R

ratio parameter, 479
 readout compensation gradient, 273
 readout field of view, 338
 readout gradient setting, 505
 readout gradient strength, 273
 readout image center position, 437
 readout position, 360
 real Fourier transform, 438, 439
 real number formatted into string, 231
 real scan repetition, 373
 real variable
 create real variable without value, 462
 format as string, 231
 real-time digital filter, 378
 real-time DSP (digital filtering), 187
 real-type parameter, 116
 real-value storage for macros, 458
 receiver
 channel imbalance, 113
 gain, 249, 582
 gating time, 118, 479, 480
 overflow warning, 42
 programmable filters, 404
 version in system, 459
 receiver option, 200-kHz, 168
 recording current window activity, 310
 recording keyboard entries, 462
 REDOR1 pulse sequence, 463
 reference deconvolution, 217
 reference frequency
 position, 464, 465
 reference line
 clear referencing, 117, 118
 frequency, 474
 position, 473
 reference frequency, 463
 set line, 477, 478
 reference peak, see reference line
 reference spectrum to TMS, 573
 refocus pulse width, 381
 refocusing gradient for slice selection, 275
 refocusing pulse shape, 451
 regions
 divide spectrum into regions, 465
 find tallest peak, 401
 frequency limits of specified region, 256
 in spectrum, 374
 plot expansions, 49
 selection, 291
 region-selective 3D processing, 447
 regression analysis data input, 477
 regression mode, 53
 regression mode curve fitting, 207
 regression scale limits, 490

Index

- regression.inp file, 209, 431, 477
 - relay switching version, 63
 - RELAY-COSY pulse sequence, 466
 - RELAYH pulse sequence, 466
 - release procedure, 362
 - remote file transfer to local host, 203
 - remote machine name, 40, 41
 - removing
 - dc offsets, 132
 - directories, 479
 - files, 478
 - user macro, 344
 - renaming a command, 285
 - renaming files, 363, 466
 - reset points for integrals, 121
 - Resets button, 256
 - resetting acquisition computer, 35
 - resolution enhancement function, 315
 - resolution enhancement parameters, 468
 - resolution equalization, 61
 - resolution on printers and plotters, 434
 - resonance offset frequency, 468
 - Results button, 526
 - retrieving
 - FIDs, 481
 - parameters, 482
 - Return button selection of menu, 314
 - reverse INEPT, 291
 - rf band in use, 469
 - rf channel selection, 470
 - rf channel type, 471
 - rf channels available, 375
 - rf channels frequencies, 502, 533
 - RF Control board, 63, 106
 - rf generation type, 475
 - rf power for desired flip angle, 502
 - rf pulse calibration identity, 472
 - rf pulse shape analysis, 449
 - rf pulses setup, 301
 - rf waveform generator, 476
 - ridges in FID display, 232
 - Right button, 476
 - right half of screen display limits, 476
 - right phase parameter, 481
 - right phase-correction angles, 480
 - ROESY button, 479
 - ROESY experiment, changing parameters for, 479
 - ROESY parameter set, 276
 - ROESY pulse sequence, 479
 - root-mean-square noise, 257, 372
 - rotating 2D data, 480
 - rotating frame NOE experiment, 574
 - rotating frame Overhauser experiment, 479
 - rotational echo double-resonance, 463
 - rotor speed display, 293
 - rotor synchronization, 293
 - configuration parameter, 480
 - spinning rate, 540
 - Rotor Synchronization label, 104, 293, 480
 - RS-232 cable, 42
 - running FDM program, 216
- ## S
- S2PUL button, 486
 - s2pul3rf parameter set, 387
 - sample
 - change for acquisition, 98
 - ejection from probe, 197, 200
 - insert in probe, 296, 304
 - location of samples in tray, 324
 - spin rate, 534
 - submit change sample experiment to
 - acquisition, 487
 - temperature, 384, 567
 - sample changer
 - automation data file prefix, 75
 - automation mode active, 70, 73
 - automation run preparation, 70
 - change sample experiment, 98
 - controlling, 71
 - controlling macro for automation, 71
 - errors, 43
 - last lock solvent used, 314
 - resume suspended automation run, 77
 - serial port, 103, 105
 - serial port connection, 530
 - starting automation run, 73
 - status window, 545
 - suspend automation run, 78
 - tray size, 580
 - Sample Changer label, 103, 105, 580
 - Sample Changer Serial Port label, 530
 - sample information for automation run, 200
 - Sample Management System serial port connection, 530
 - sample tray size, 103, 105
 - sampleinfo file, 71, 76
 - saving
 - data, 550
 - digitally filtered FIDs, 149
 - display parameters, 486
 - experiment data to subfile, 556
 - FID data in FDF format, 551
 - FIDs in current experiment, 551
 - files using a base name, 488
 - images as FDF files, 553, 555
 - images as ImageBrowser files, 552
 - images as phasefiles, 348
 - parameters from current experiment, 554
 - parameters to file, 234
 - phasefile in current experiment, 554
 - shim coil settings, 555
 - text file into another file, 449
 - scale below spectrum or FID, 182, 443
 - Scale button, 443
 - scale limits in regression, 490
 - scale spectral width, 491
 - scaling constant, 363
 - scaling factor for multipulse experiments, 490
 - scaling factor for plots, 294
 - scaling factors, 82
 - scan in progress, 120
 - scout experiment, 579
 - scout run, 516
 - scout scan repetitions, 373
 - scpos parameter, 579

- screen coordinates, translating, 580
- screen display set for center, 96
- screen distance, translating, 581
- SCSI errors, 44
- second decoupler
 - acquisition parameters, 145
 - adjust tip-angle resolution time, 157
 - decoupler mode, 154
 - decoupling sequence, 183
 - fine power attenuator, 176
 - frequency, 141
 - frequency offset array, 493
 - frequency offset control, 164
 - homodecoupling control, 290
 - linear modulator power, 176
 - modulation frequency, 156
 - modulation mode, 161
 - nucleus lookup, 162
 - power level with linear amplifier, 174
 - pulse sequence diagram, 173
 - set frequency to cursor position, 492
 - tip-angle resolution, 179
- second delay, 124
- selectable 4nuc (HCPF) experiment, 68
- selectable large-signal mode preamplifier, 435
- selected widths, setting, 518
- selective excitation experiment, continuing, 550
- selective excitation pulse sequence, 495
- selective frequencies, setting, 518
- selective inversion, setting up, 518
- send command to VNMR, 495
- Seq label on screen, 446, 496
- seqfil file, 173
- seqgenmake file, 497
- seqlib directory, 211, 497
- serial port connection, 530
- serial port for sample changer, 103, 105
- Set colors for Graphics Window, 499
- set colors for Plotter, 499
- Set Default button, 135
- Set Params button, 529, 530
- setup experiment, 548
- setup macros, 225
- sf wf button, 521, 609
- shape of an excitation pulse, 381
- shape of refocusing pulse, 451
- shaped gradients tests, 274
- shaped observe excitation sequence, 522
- shaped pulse analysis, 449
- shaped pulse calibration, 85, 448
- shapeinfo file, 85, 448
- shapelib directory, 85, 183, 448, 453
- shared amplifier type, 52
- shell on UNIX, 523
- Shifted Laminar Pulses (SLP), 316, 529
- shifted sinebell squared window function, 412
- shim coil settings, 462
 - retrieve from file, 483
 - save to file, 555
- shim gradient, 621, 623, 624, 626, 627, 628, 629, 630, 631
- shim method string creation, 365
- shim method string display, 185
- shim parameters group, 147
- shim set type, 524
- shim settings directory, 525
- shim supply, 524
- shim values comparison, 149
- shim values used, 324
- shimmap calculations, 278
- shimmethods directory, 185, 354
- shimming
 - automatic shimming conditions, 617
 - Autoshim method, 354
 - errors, 43
 - interactive, 38
 - Z1 hardware, 283
- shims
 - list of, for hardware shimming, 283
 - read all shims, 460
 - set all shims, 499
- shims directory, 483, 525, 555
- Shimset label, 103, 524
- Show Params button, 146
- Show Target button, 178, 416
- Show Time button, 572
- sidechain assignments, 281
- signal-to-noise ratio, 531, 568
 - estimate, 257
 - improvement, 221
 - maximum, 186
 - measurement, 186
- sine value of angle, 526
- sine window function values, 527
- Sinebell button, 527
- sinebell constant, 488, 489
- sinebell shift, 619
- sinebell shift constant, 489
- sinebell squared window function, 412
- sinebell time period, 619
- sinebell weighting parameters selection, 527
- sinebell-squared window function, 540
- sine-squared window function, 528
- SIS (12 bit) gradients, 105
- SISCO Imager console type, 107
- skyline projection, 442
- Slice button, 416
- slice gradient levels, 506
- slice parameters, 543
- slice parameters set for target slice, 529
- slice plane orientation, 377
- slice position, 446
- slice selection fractional refocusing, 275
- slice selection gradient level, 276
- slice selection gradient strength, 275
- slice selection refocusing gradient, 275
- slice thickness, 571
- slices to be acquired, 373
- slice-selective excitation pulse, 630
- Small button, 530
- small graphics window, 530
- software preparation date, 469
- software revision level, 469
- solids
 - adjust tau2 to current cursor position, 582
 - cross polarization spin-lock experiments, 54
 - cross-polarization spin-lock analysis, 208
 - echo pulse sequence, 541

Index

- f_1 spectral width scaling factor, 491
- first pulse phase, 409
- MREV8 multiple-pulse experiment, 361
- multiple-pulse line narrowing, 121
- rotor speed display, 293
- rotor synchronization module, 480
- scaling factor for multipulse experiments, 490
- solid-state echo pulse sequence, 541
- solid-state HETCOR sequence, 285
- spinning speed for MAS, 540
- VT controller in use, 352
- solvent resonances ppm and peak width, 531
- solvent selection, 519
- solvent subtraction
 - create parameters, 389
 - filter bandwidth for filtered FID, 542
 - order of polynomial to fit digital filtered FID, 543
 - parameter creation, 47
- solvent suppression, 239
- solvent table information, 532
- solvents file, 519, 531, 532
- solvent-suppressed region, 542
- sparse FID data points, 167
- spatial resolution calculation, 467
- spectra
 - 3D dc correction, 533
 - absolute value display mode, 80
 - add spectrum to add/subtract experiment, 532
 - APT plotting, 417
 - automatic 1D integrate, 305
 - automatic phase, 55
 - automatic phase adjustment, 55
 - center cursor, 96
 - data truncation limit, 121
 - deconvolution, 223
 - delete spectra from T_1 or T_2 analysis, 136
 - display calculated spectrum, 186
 - display scale, 182
 - display single spectrum, 180
 - divide spectrum into regions, 465
 - drift correction calculation, 127
 - drift correction parameters, 95
 - extract planes from 3D spectral data, 255
 - find gap in spectrum, 250
 - find peak heights or phases, 231
 - find tallest peak in region, 401
 - fold COSY-like correlation spectra, 230
 - fold J-resolved 2D spectrum, 230
 - frequency shift of spectrum, 340, 341
 - frequency-independent phase, 408
 - full display, 212
 - horizontal offset of each spectrum, 288
 - inset spectrum display, 304
 - integral amplitudes display, 170
 - integral amplitudes plot, 413
 - integral regions, 291
 - intensity of a spectrum at a point, 350
 - interactive display, 38
 - move cursor to center, 96
 - move spectral window according to cursors, 360
 - noise limit, 372
 - normalized integral amplitudes display, 170
 - normalized integral amplitudes plot, 413
 - normalized intensity mode, 369
 - number of regions, 374
 - offset of integral, 306
 - peak frequencies display, 169
 - peak height comparison, 50
 - peak search range of data points, 373
 - phase adjustment, 55
 - phase angle display mode, 383
 - phased display mode, 383, 407, 408
 - phosphorus processing, 382
 - phosphorus spectrum plot, 427
 - plot a scale under a spectrum, 443
 - plot arrayed 1D, 418
 - plot COSY automatically, 419
 - plot NOESY automatically, 419
 - plot one or more spectra, 414
 - plot peak frequencies, 433
 - plot spectra in whitewash mode, 430
 - power spectra display mode, 451, 452
 - processing simple 1D, 439
 - proton spectrum plotting, 421
 - reduce spectral width to minimum, 358
 - reference line frequency, 474
 - reference line position, 473
 - reference to TMS, 573
 - rotate 2D spectrum, 480
 - select spectrum without displaying, 494
 - signal-to-noise estimate, 257
 - signal-to-noise test, 568
 - solvent-suppressed region, 542
 - spectral integral display and plot, 305
 - stacked spectra display, 189, 190, 192, 193, 194
 - subtract spectrum from add/subtract experiment, 539
 - threshold for integrating 2D peaks, 570
 - total width to be acquired, 557
 - update region during phasing, 410
 - vertical offset in stacked display, 597
 - vertical position, 598
 - vertical scale, 599
 - whitewash mode display, 195
- spectra in 2D data set, rearrange, 565
- spectra in a 2D data set, rearrange, 565
- spectral expansion automatic plot, 49
- spectral subtraction, 539
- spectral width, 557, 558
 - Input board, 352
 - percentage for gradient shimming, 278
 - reduce to minimum, 358
 - set for given spectral window, 517
 - set in 2nd evolution dimension, 517
 - set in evolution dimension, 517
- spectrometer proton frequency, 279
- spectrometer system configuration, 559
- spectrum, plotting on side, 426
- spectrum, plotting on top, 427
- spectrum, plotting on top and side, 427
- spin automation hardware, 536
- spin hardware, 105, 106
- spin rate of sample, 534
- spin setup experiment, 534
- spin simulation

- clear line assignments, 98
- deconvolution start point, 588
- display group of parameters, 146
- index of experimental frequency, 100
- intensity threshold, 546
- linewidth, 529
- maximum frequency of any transition, 530
- measured line frequencies, 528
- measured line frequencies array, 535
- minimum frequency of any transition, 530
- number of iterations, 367
- parameter arrays, 151
- parameters to be iterated, 307
- perform spin simulation, 536
- set parameters to iterate, 302
- spin system entry, 539
- transition amplitude, 99
- transition frequency, 100
- transition number, 98
- using deconvolution as input, 223
- vertical scale, 555
- spin speed interlock, 301
- spin system
 - enter values for spin simulation, 539
 - restoring to before last run, 585
- spin-echo diffusion imaging sequence, 493
- spini.inpar file, 585
- spini.la file, 151
- spini.savela file, 585
- spin-lock contact time, 108
- Spinner Control window, 535
- spinner errors, 42
- spinner speed display, 39, 40
- spinning speed for MAS, 540
- spins.list file, 536
- spins.outdata file, 187
- Spinsight data, 482
- spinsys directory, 302
- spline fitting of baseline, 86
- split difference between two cursors, 538
- spoiler gradient level, 275
- spoiling time for gradient, 582
- square root image, 299
- square wave mode decoupling, 155, 160
- SSECHO button, 541
- SSECHO pulse sequence, 541
- SSECHO1 pulse sequence, 541
- stacked display width, 605
- stacked FIDs, 142, 143, 144
- stacked plot of 2D spectra, 181, 415
- stacked spectra display, 189, 190, 192, 193, 194
- stacked spectra horizontal offset, 288
- stacking control, 545
- stacking mode, 544
- standard 2-pulse sequence, 381, 486
- standard 2-pulse sequence in reverse, 486
- standard application mode, 58
- standard deviation of input, 81
- standard flip angle list, 227
- start of FID display, 521
- start of interferogram, 521, 522
- start of plot, 532
- starting
 - VNMR directly, 594
 - VNMR from UNIX, 595
- starting Plot Designer, 308
- startup macro, 90
- static binding, 497
- static magnet field value, 85
- stdpar directory, 519
- steady state pulses, 541
- Step Size label, 104, 575
- step size parameter value array, 392
- stimulated echo technique, 546
- stopping acquisition, 487
- stored FIDs, recalling, 79
- stored parameters, recalling, 79
- stored queue, 73
- streaming tape, 563
- strength of pulsed field gradient, 278
- string
 - format for output, 231
 - length in characters, 318
 - select substring, 549
 - text window display, 198
- string parameter values, 501
- string variable creation, 546
- string-type parameter, 116
- subfile, 114
- substring selected from a string, 549
- Subtract button, 539
- subtracting zero-frequency components, 389
- sum of input, 81
- sum of squares of input, 81
- sum/difference spectrum, 45
- summing projection, 442
- sum-to-memory error, 43
- Sun display, clear window, 99
- svsdfd command, 553
- swept-tune graphical tool, 456
- switchable probe caution, 148, 173, 174
- synchronous decoupler mode, 154
- Synthesizer label, 104, 447
- synthesizer value, 447
- syshelppath global parameter, 57
- sysmaclibpath global parameter, 57
- sysmaclibpath parameter, 205
- systemenulipath global parameter, 57
- system administrator, 102
- system configuration parameters, 102
- system console type, 107
- system macros
 - copy system macro to become user macro, 345
 - display in text window, 344
 - edit online description, 349
 - list system macro names, 345
 - online description, 349
 - remove system macro, 345
- system type configuration, 559
- System Type label, 103, 105, 106, 107, 475, 559
- systemglobal directory, 116
- systemglobal parameter tree, 103
- systemglobal-type parameter tree, 116

T

T_1 analysis, 208

Index

- delete spectra, 136
- plot curves, 403
- set up parameters, 167
- T₁ analysis, 54, 560
- t₁ dimension, 339
- T₂ analysis, 208
 - delete spectra, 136
 - plot curves, 403
- T₂ analysis, 54, 561
- t₂ dimension in, 339
- table conversion file, 565
 - read, sort, store, 565
- table conversion reformatting, 565
- table convert file
 - read, sort, and store, 566
- tablib directory, 562
- tangent value of angle, 563
- tapes
 - device selection, 564
 - display contents, 563
 - rewind tape, 563
- taps in digital filter, 542
- target experiment
 - move parameters, 579
- target scan planning, 415
- target slice parameters, 543
- target slices, 178
 - set slice parameters, 529
- target voxels, 178
- tau2 adjustment, 582
- Tcl script, 565
- tcrush parameter, 252
- tdelta parameter, 252
- tdiff parameter, 252
- temperature calculation curve, 567
- Temperature Control window, 566
- temperature display, 39, 40
- temperature interlock, 572
- temperature of sample, 567
- temperature regulation, 602
- terminating
 - abort function in macro, 35
 - calling macro, 35
- testing signal-to-noise of spectrum, 568
- text file
 - display for current experiment, 569
 - edit file, 569
 - edit with vi text editor, 592
 - editor, 199, 592
 - graphics window display, 195
 - plotting, 428
 - print text files, 446, 597
 - put into another file, 449
 - search for words and lines, 328
 - write file using a FID element, 616
- text output sent to printer, 437
- text window
 - changing the size, 227
 - display files, 94
 - display status, 569
 - display strings and parameter values, 198
 - display user macro, 342
 - list files, 149
- tflow parameter, 259
- third decoupler
 - adjust tip-angle resolution time, 157
 - decoupler mode, 154
 - decoupling sequence, 183
 - fine power attenuator, 176
 - frequency, 141
 - frequency offset array, 493
 - frequency offset control, 164
 - homodecoupling control, 290
 - linear modulator power, 176
 - modulation frequency, 156
 - modulation mode, 161
 - nucleus lookup, 162
 - power level with linear amplifier, 174
 - set frequency to cursor position, 492
 - tip-angle resolution, 179
- three-axis gradients, 117
- threshold for integrating peaks in 2D spectra, 570
- threshold for peak printout, 570
- threshold for printout of peak frequencies, 570
- thumbwheel switches on lock transmitter, 326
- tilted box, drawing a, 564
- time constant for lock, 327
- time constant for lock acquisition, 326
- time counter, 622
- time-domain cursor position, 117, 582
- time-domain cursors, 137, 573
- time-domain dc correction, 217
- time-domain solvent subtraction, 389
- time-shared decoupling, 289
- tip-angle resolution for decoupler, 179
- tip-angle resolution time for decoupler, 156, 157, 158
- TMS reference, 573
- TNCOSYPS sequence, 574
- TNDQCOSY sequence, 574
- TNMQCOSY sequence, 574
- TNNOESY parameter set, 276
- TNNOESY sequence, 574
- TNROESY sequence, 574
- TNTOCSY sequence, 574
- TOCSY button, 575
- TOCSY experiment, changing parameters for, 575
- TOCSY pulse sequence, 237, 242, 575, 611, 612
- TOCSY ID experiment, changing parameters for, 575
- tof parameter, 319
- total correlation (TOCSY) experiment, 575
- traces
 - find maximum intensity, 402
 - select trace without displaying, 494
- TRANSFER.par file, 579
- Transform button, 236
- transform images into phasefiles, 348
- transformed image array index, 200
- transformed image echo index, 198
- transients completed, 119
- transients setpoint action, 613, 614
- transients to be acquired, 373
- transition amplitude, 99
- transition calculation, 61
- transition frequency, 100, 530
- transition number calculation, 98
- transitions frequency, 530

- transmitter
 - fine power, 578
 - frequency of observe nucleus, 522
 - frequency offset for observe transmitter, 267, 575
 - linear modulator power, 578
 - local oscillator (L.O.) gate, 328
 - move transmitter offset, 361
 - nucleus of observe transmitter, 573
 - positioning, 575
 - power level with linear amps, 576
 - pulse sequence diagram, 173
 - transmitter frequency, 141
 - transmitter frequency, 141
 - transverse magnetization generation, 630
 - tray size on sample changer, 580
 - trigger pulses, 571
 - trigger signals to wait before acquisition, 374
 - trim gradient level, 276
 - triple-quantum filtered 2D MAS experiment, 126
 - TROESY pulse sequence, 581
 - truncation limit, 121
 - TUNE INTERFACE unit, 582
 - tuning
 - broadband channel on MERCURY and GEMINI 2000, 91
 - carbon channel on GEMINI 2000, 119
 - lock channel on GEMINI 2000, 196
 - mode, 583
 - proton channel on GEMINI 2000, 294
 - tuning the probe, 582
 - Type of Amplifier label, 52, 105
 - type of parameter, 518
 - Type of RF label, 104, 471, 475
- U**
- U+ H1 Only decouplers, 471
 - Ultra•nmr shim system, 462, 524
 - unit conversion for parameters, 585
 - UNITY console type, 107
 - UNITY^{INOVA} console type, 107
 - UNITY^{plus}
 - console type, 107
 - probe tuning, 582
 - UNIX shell startup, 523
 - UNIX text files conversion to VXR-style format, 586
 - unlocked experiment, 587
 - unshifted cosine-squared window function, 540
 - unshifted Gaussian window function, 250
 - unshifted sinebell-squared window function, 540
 - updating
 - gradient coil, 588
 - updating revision global file and parameters, 588
 - upper case format of string, 231
 - Upper Limit label, 104, 576
 - Use Console Data button, 102
 - use ll button, 535
 - user macros
 - copy file, 342
 - copy system macro to become user macro, 345
 - create without text editor, 115
 - delete, 135
 - display in text window, 342
 - edit with vi text editor, 346
 - library, 115
 - list user macro file names, 343
 - path to user macro directory, 342
 - remove user macro from directory, 344
 - user-defined weighting, 617, 618
 - users currently on the system, 604
 - user-selectable editor, 343
 - user-supplied modulation, 160
 - user-written weighting functions, 618
 - usrwt.o file, 618
- V**
- value of parameter in a tree, 257
 - variable temperature, see VT
 - Varian shim supply, 524
 - VAST accessory, 590
 - VAST data analysis, 101
 - VAST experiments, setting up initial parameters for, 590
 - VAST microtiter plate, 101
 - version of parameter set, 393
 - vertical offset, 597
 - vertical position
 - FID, 598
 - imaginary FID, 598
 - spectrum, 598
 - vertical projection of trace, 130
 - vertical scale adjustment, 599, 600, 601
 - vertical scale for 2D displays, 599
 - vertical scale for projections and traces, 601
 - vertical scale for simulated spectrum, 555
 - vertical scale for spectrum, 599
 - vertical scale of FID, 592
 - vi command (UNIX), 199, 592
 - vi text editor, 346, 354, 592, 594
 - VNMR
 - accounting program, 596
 - background processing, 591
 - error messages, 203
 - exiting, 206
 - exiting from system, 596
 - lines in error message display, 204
 - software preparation date, 469
 - software revision level, 469
 - start in windowing system, 595
 - start VNMR application directly, 594
 - style of stored data, 109, 114
 - system administrator, 102
 - system directory, 559
 - updating parameters and global file after install, 588
 - user directory, 589
 - write out memory buffers, 228
 - vnmr_textedit file, 388
 - vnmr_vi file, 388
 - vnmr1 user, 102
 - vnmraddr parameter, 233
 - vnmreditor variable, 199, 387
 - vnmrssystem variable, 559

Index

vnmruser variable, 589
volume localized spectroscopy sequence, 546
voxel
 dimensions, 597
 parameters, 598
 planning menu, 416, 598
 selection, 276
Voxel button, 416
voxel selection gradient levels, 506
voxel selection gradients setup, 301
VT Controller label, 103, 105, 106, 602
VT controller type, 602
VT cutoff point, 601
VT errors, 42
VT FAILURE message, 602
VT regulation light, 572
VT system in use, 352
VT wait time, 602
VXR-S console type, 107
VXR-style directory
 decompose to UNIX files, 134
VXR-style systems
 convert data to VNMR, 109, 114
 decompose files to UNIX files, 134
 read tapes, 563
 remote directory display, 200
VXR-style text files
 conversion to UNIX format, 602
 UNIX text file conversion to, 586

W

w command (UNIX), 604
walkup automation, 604
WALTZ decoupling present, 604
WALTZ decoupling sequence, 155
WALTZ-16 modulation, 160
warning error codes, 42
water suppression, 87, 310, 435, 574, 575
waveform generator, 476, 522
 decoupling, 179
 pulse interval time, 453
 test, 610
waveform generator decoupling, 179
Waveform Generator label, 104, 105, 476
weight and Fourier transform
 1D data, 610
 2D data, 611
 along f_2 for 2D data, 610
 phase-sensitive data, 610, 612
Weight, Transform button, 610
weight.h file, 618
weighting
 constant, 81
 interactive weighting, 619
 interactive weighting for 2D absorptive data, 619
 user defined, 617
weighting function compilation, 618
WET1D pulse sequence, 607
WETDQCOSY pulse sequence, 607
WETGCOSY pulse sequence, 607
WETGHMQCPS pulse sequence, 607

WETGHSQC pulse sequence, 607
WETNOESY pulse sequence, 607
WETPWXCAL pulse sequence, 608
WETTNTOCOSY pulse sequence, 608
wexp parameter, setting up, 78
WFG (waveform generator), 271
WFG + GCU gradients, 105
what you see is what you get, 620
whitespace in text file, 328
whitewash mode, 404, 415, 430
whitewash mode display, 144, 181, 182, 195
who is using system, 604
wideline systems data precision, 168
width of chart, 605
 maximum width, 605
 maximum width in second direction, 606
 second direction, 605
width of FID, 609
width of interferogram, 609
width of plot, 614
wildcard characters, 478
window
 current, 120
window activity, 310
windowing system, 595
windows
 clearing a window, 99
 flip between large and small, 227
word in text file, 328
word lookup in text file, 328
workspace for VNMR experiment, 97
wtlib directory, 618

X

X Axis, Y Axis, Z Axis label, 105, 270
X gradient strength, 277
X,H-correlation 2D spectrum, 423
X1 shim gradient, 621
X2Y2 shim gradient, 621
X3 shim gradient, 621
X4 shim gradient, 621
XL Interface board, 57
XL systems
 convert data to VNMR, 109
 convert files to UNIX format, 602
 decompose files to UNIX files, 134
 list contents of directory, 200
 read tape, 563
 UNIX text files conversion to, 586
XPOLAR button, 622
XPOLAR pulse sequence, 622
XPOLAR1 pulse sequence, 623
XY shim gradient, 623
XY32 decoupling sequence, 155, 160
XZ shim gradient, 623
XZ2 shim gradient, 623
x-zero position of Hewlett-Packard plotter, 621

Y

Y gradient strength, 277

Y1 shim gradient, 624
 Y3 shim gradient, 624
 Y4 shim gradient, 624
 YZ shim gradient, 624
 YZ2 shim gradient, 624
 y-zero position of Hewlett-Packard plotter, 624

Z

Z gradient strength, 277
 z0 calibration, automatic, 66, 68
 Z0 field position, 626
 Z0, automatic adjustment, 67
 Z1 shim gradient, 626
 Z1C shim gradient, 626
 Z2 shim gradient, 627
 Z2C shim gradient, 627
 Z2X2Y2 shim gradient, 627, 628
 Z2X3 shim gradient, 627
 Z2XY shim gradient, 627
 Z3 shim gradient, 627
 Z3C shim gradient, 628
 Z3X shim gradient, 628
 Z3X3 shim gradient, 628
 Z3XY shim gradient, 628
 Z3Y shim gradient, 628
 Z3Y3 shim gradient, 628
 Z4 shim gradient, 628
 Z4C shim gradient, 629
 Z4X shim gradient, 629
 Z4X2Y2 shim gradient, 629
 Z4XY shim gradient, 629
 Z4Y shim gradient, 629
 Z5 shim gradient, 629
 Z5X shim gradient, 629
 Z5Y shim gradient, 630
 Z6 shim gradient, 630
 Z7 shim gradient, 630
 Z8 shim gradient, 630
 z-axis shims used for gradient shimming, 278
 zero-filling, 228, 468
 zeroing phase, 118
 zero-order baseline correction, 341
 zero-order phase, 480, 481
 zero-order phasing constant, 410, 411, 481
 zero-order term automatic phase, 56
 Zeta plotter, 55
 zfs (zero-frequency suppression) option, 389
 ZX2Y2 shim gradient, 631
 ZX3 shim gradient, 631
 ZXY shim gradient, 631
 ZY3 shim gradient, 631

Index