Chapter 1

Single Dish

package for single dish processing

include "dish.g"

Description
The dish package contains modules needed for processing single dish data.

1.1 dish - Module

Single Dish Analysis Environment

Description
The single-dish environment within AIPS++is known as **dish**. See the Getting Results:Single Dish Analysis note for a complete description of the various graphical components and tools which together make up the **dish** module along with recipes for reduction.
dish is actually a set of nested tools with broad functionality. We organize them into several groups to facilitate finding the appropriate function for a given task, though all functions are available as a group from the command line.
There are additional functions available in dish through the AIPS++ plug-in utility. For example, the calibration routines are instrument-specific. Please see the chapters in the Getting Results Manual for details.

```
> glish -l dish.g
Glish version 2.7.
- const d := dish() # create my dish tool, calling it 'd'
Time to initialize DISH =  8.797364
- field_names(d)
```

`addop addtool busy clearrm dologging files filein gauss wsname fileout find doselect done nogui getscan gui history dhistory info listscans logcommand message mult open ops tools rm plotter plotcom plotscan plotxy rmadd restorestate savestate scanadd scansub scandiv scansrr scanmult scanscale scanbias scanscale2D scanbias2D summary statefile tofile type uniput uniget view_sdrec gaussian boxcar dumcal calibSM calibSP import fixpnt gaincorr tip1 tip tps1 fsw1 dclip aver base save select smooth stat`

For the following, we divide the core functionality of **dish** into 'user' and 'developer' sections. The 'developer' sections are designated by a 'dishd', though they are accessed in the same fashion as any other **dish** function, i.e., d.fn(...). We also present information on the functions available on tools contained within **dish**, namely the plotter and results manager (**rm**) tools.

Tools

dish	dish tool (user functions)
dishd	dish tool (developer functions)
msr	dish spectral line measure tool
plotter	dish plotter tool
rm	dish results manager
sdaverager	A tool for averaging single dish data
sditerator	A `tool` for interacting with single dish data.

Functions

dishdemodata	Set up demonstration data for Dish

1.1.1 dish - Tool

Package dish
Module dish

dish tool (user functions)
include "dish.g"

Constructors

dish	single dish analysis tool

Functions

accum	Add spectrum to the accumulator.
addstack	Append new scans to the stack.
ave	Average the contents of the accumulator.
aver	Average a group of scans or subscans.

avgpols	Average the polarizations in the globalscan1 spectrum
base	Perform a polynomial baseline fit, all in one function. An alternative step-by-step method is to use the nfit, setregion, bshape, and baseline functions
baseline	Perform a polynomial baseline fit to spectrum in *globalscan1*
bdrop	Set number of channels to drop from the beginning of a spectrum, used only in displaying.
bias	Add a bias to the scan in memory
boxcar	Perform a boxcar smoothing
bshape	Show the baseline fit
cal	Apply calibration to a scan (GBT only currently).
calib	Apply calibration to a scan (GBT only currently).
clearrm	Clear all variables from results manager
clip	Clip/Edit data (set flags)
copy	Copy a variable in the global record to another location
dcbase	Subtract a DC baseline (0th order poly) from the spectrum
delete	Delete a single entry from the stack.
divide	Divide globalscan1 by offscan1.
done	Exits dish and destroys the current dish tool.
edrop	Set number of channels to drop from the end of a spectrum, used only in displaying.
empty	Clear the stack.
filein	Sets the scan group used by scan access functions.
fileout	Sets the scan group that can be written to.
files	Prints the currently set values for filein and fileout.
find	Search all glish variables for a specified string.
flag	Flag data based on various specifications.
galactic	Provide coordinates of the globalscan1 spectrum in the Galactic reference frame.
gauss	Fit gaussian profiles.
gbtsum	Summarize scans in GBT projects from the /home/gbtdata area (GB only)
getc	Retrieve calibrated data (from the CORRECTED_DATA column)
getr	Retrieve raw data (from the FLOAT_DATA column)
getscan	Retrieve a scan (SDRecord) from a scan group.
getvf	Prints the velocity and frequency for a specified channel.
getvfarray	Return the velocities and frequencies for the globalscan1 spectrum.
gms	Summarize the scans in the currently opened data set.
gsflag	Flag some channels in the globalscan1 spectrum
gsget	Retrieves the globalscan1 record to a glish variable
gsput	Sends an sdrecord from glish to the globalscan1 record
gsread	Retrieve a spectrum saved to disk into the globalscan1 record
gswrite	Write the globalscan1 spectrum to disk
gui	Toggles mapping of DISH main gui (results manager).
hanning	Perform a hanning smooth
header	Print formatted header information to the command line window
help	Provide simple help on dish functions
history	Add a string or vector of strings to the history of a current SDRecord.
imagems	Make an image from all scans in a MeasurementSet.
imcal	Calibrate an imaging/mapping procedure
import	Import GBT data into an AIPS++ MeasurementSet
keep	Saves the current globalscan1 SDRecord to the designated fileout.

listscans	List the scan numbers from the active scan group.
logcommand	Writes command to scripter.
lsoutfile	Lists scans in the fileout scangroup
makeim	Make an image from scans in a MeasurementSet.
message	Writes text to the DISH GUI message line.
minus	Subtract offscan1 from globalscan1.
mscal	Calibrate a MeasurementSet.
mult	Scales (multiplies) a scan by a value.
multiply	Multiply globalscan1 by offscan1.
nfit	Set the order of the polynomial for baseline subtraction.
nogui	Dismisses the DISH main GUI.
nregion	Set the channel ranges for baseline subtraction.
open	Load a scan group from disk into DISH and perform a filein on it.
page	Clears the DISH plotter display.
peak	Obtain the peak amplitude, half-width and center in the displayed spectrum
plotc	Plot calibrated data (from the CORRECTED_DATA column)
plotr	Plot raw data (from the FLOAT_DATA column)
plotscan	Plot an SDRecord.
plotxy	Plot two vectors in the DISH plotter.
plus	Add offscan1 to globalscan1.
qscan	Query characteristics of scan.
reference	Designate the reference spectrum
restore	Retrieve the contents of the uni Record from disk.
restorestate	Restore the state of DISH's GUIs.
rmadd	Add an SDRecord to the results manager.
rms	Display statistics on baseline regions.
save	Save an SDRecord to file.
savestate	Save the current state of DISH.
saxis	Set the units of the x axis in the plotter
scale	Multiply the globalscan1 spectrum by a value.
scanadd	Add scan1 and scan2.
scanbias	Adds a bias level to a scan.
scandiv	Divide scan 1 by scan 2.
scanmult	Multiply scan 1 by scan 2.
scanscale	Multiply an sdrecord's spectrum by a value or array
scansrr	Performs a (signal-reference)/reference.
scansub	Subtract scan2 from scan1.
sclear	Clear the contents of the accumulator.
select	Select a subset of a scangroup.
setYUnit	Set the y-axis unit for a spectrum
setregion	Set the channel range for use in baseline subtraction
show	Display the globalscan1 SDRecord
show1	Display one polarization of the globalscan1 SDRecord
showref	Display the vref SDRecord
signal	Designate the signal spectrum
smooth	Smooth (hanning, boxcar or gaussian) an SDRecord.

stat	Generate statistics over a line region.
statefile	Changes file for state saving.
stats	Generate statistics over a region.
store	Store the contents of the uni Record to disk.
summary	List a summary for a Measurement Set
tellstack	Return contents of the stack.
upr	Print the current value of variables in the uni-record.
utable	Print channel and data columns to command window
writetofile	Write X,Y axes to an ascii file on disk.
zline	Draw a horizontal line at zero intensity on the plotter.

Description

Including "dish.g" loads the single-dish analysis environment and creates a default tool, **d**, for use.

dish knows how to read from and write to AIPS++tables which have been created from Single Dish FITS files (SDFITS). The documentation for dishdemodata provides more details. The Using Dish note has an appendix which describes the SDFITS convention.

You can load one of these AIPS++tables (for example, the sample data generated by dishdemodata) into **dish** by using one of the two options in the File/Open menu. The difference between "Read Only" and "Reading and writing" when opening a **dish** data set is whether you can save a result to this data set through the save operation. It is generally a good idea to save to a different data set from your raw data. The File/New menu can be used to create an empty AIPS++table suitable for storing **dish** results. Choosing any one of the menu options (Read Only, Reading and writing or New) will launch a file chooser window. Simply select one of the files (or type in a new file) and press "ok". The program is now ready to use the contents of that table.

The "Results Manager" is where all of the variables created by **dish** are shown. These are all global symbols which you can manipulate directly at the glish prompt. There are three types of variables which **dish** creates and manipulates.

- SDRecord - a glish record with a known structure. This holds the data plus header values for a single "scan". **dish** knows how to display SDRecords in the plotter. Selecting an SDRecord in the results manager window will always cause that SDRecord to be displayed on the plotter. SDRecords may also be browsed. In this context, pushing the browse button will launch a window which will allow you to browse the header values of this record. See the SDRecord information in the sditerator Tool.

- SDIterators - these are also called working sets in the **dish** GUI. They are collections of SDRecords. The disk files which **dish** reads from and writes to are SDIterators/working sets as are any selections which are made from these files. **Dish** knows how to browse these. Pushing the browse button will launch a window which will allow you to select any of the records in this collection. When selected, it will be plotted. The header values for each selection may also be browsed through this browser. See also the sditerator tool documentation.

- Anything else. **Dish** knows very little about these values. They can neither be plotter or browsed although they can be inspected.

The variables in the results manager come with a description which you can change by simply editing the description shown and pressing the return or enter key. Similarly, you can change the name of any variable by entering a new name in the entry in the results manager and pressing the return or enter key. The “Delete” button removes a variable completely from glish although it will not delete any files on disk.

To bring up the GUI for the operations, select one from the “Operations” menu. Due to the limited amount of screen real estate typically available, you will only be able to see a few operations at once. The operations each have a “Dismiss” button generally towards the lower left corner which will remove that frame from the **dish** window. This can also be achieved by re-selecting that operation from the “Operations” menu.

There are two types of operation: operations which operate on entire working sets and operations which operate on single SDRecords. Of the current operations, selection and averaging operate on entire working sets (selection produces a new working set and averaging produces a new record). The operations which operate on a single spectra always use the most recently viewed spectra (i.e. the one you see in the plotter). This is true even if the working set from which it came or the single record which produced that plot have been “Delete”d from the results manager. What you see is what **dish** operates on. When a new scan is produced, the focus of the plotter and the program is automatically set to that result (i.e. it becomes the most recently displayed SD Record).

Baseline regions are all set through the “Baselines” operation. Whether you enter the ranges by hand or by using the plotter, if the ”Plot ranges” option has been turned on, the ranges will be plotted on top of the data. The height of each range box on the plotter indicates the local RMS of the data and the center Y value of each range box indicates the local mean of the data. The statistics for all baseline regions are shown on the plotter in the statistics line.

A full example session, suitable for a demonstration, is given in AIPS++ Note 205.

dish - Constructor

Package	**dish**
Module	**dish**
Tool	**dish**

single dish analysis tool

Synopsis
dish()

Returns
tool

User functions

dish.accum - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Add spectrum to the accumulator.

Synopsis
accum(scans)

Description
This function is used in averaging scans. The spectrum stored in the globalscan1 spectrum is added to the accumulator for later averaging. Optionally, a scan number or vector of scan numbers can be supplied to this function, and those will be added to the accumulator.

Arguments

scans	scan numbers to add to accumulator Allowed:integer or vector of integers Default: globalscan1

Returns
Boolean

Example

```
d.sclear()           # clear the accumulator
d.getc(35)           # retrieve calibrated scan 35
d.accum()            # add the scan to the accumulator
d.getc(36)           # retrieve calibrated scan 36
d.accum()            # add the scan to the accumulator
d.ave()              # average scans in the accumulator
d.show()             # display the result on the plotter

Equivalently:

d.sclear()           # clear the accumulator
d.accum([35,36])     # add the scans to the accumulator
d.ave()              # average scans in the accumulator
d.show()             # display the result on the plotter
```

---

dish.addstack - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Append new scans to the stack.

Synopsis
addstack(beg, end, inc)

Description
This function is used to add new scan numbers to the “stack”. The stack is a construct (simply a vector of integers) which is used to organize a list of scans which will be operated on in some common way. Generally, it is used to assemble a list of scans which can then be averaged.

Arguments

beg	The beginning scan number Allowed:integer Default: F
end	The ending scan number Allowed:integer Default: beg
inc	the increment Allowed:integer Default: 1

Returns
Boolean

Example

```
d.addstack(10,50,2)       # append even scans from 10 through 50 to the stack
d.addstack(102)           # append scan 102 to the stack
d.addstack(105,108)       # append scans 105 through 108 to the stack
d.aver(d.tellstack())     # average all scans in the stack
```

---

dish.ave - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Average the contents of the accumulator.

Synopsis
ave()

Description
This function is used in averaging scans. It averages all contents of the accumulator and stores the result in the globalscan1 record.

Returns
Boolean

Example

```
d.sclear()    # clear the accumulator
d.getc(35)    # retrieve calibrated scan 35
d.accum()     # add the scan to the accumulator
d.getc(36)    # retrieve calibrated scan 36
d.accum()     # add the scan to the accumulator
d.ave()       # average scans in the accumulator
d.show()      # display the result on the plotter
```

---

dish.aver - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Average a group of scans or subscans.

Synopsis

aver(scanlist, subscanlist, weighting, alignment)

Description

This function permits flexible averaging and selection/averaging of scans. The scanlist should contain a vector of integers indicating which scans to average, and the subscanlist optionally specifies which records within each scan to include. The subscanlist can either be a vector of integers or a string identifier, as outlined here:

- subscanlist = [1,2,3] : this will select only records 1-3 for averaging from the specified scans.
- subscanlist = 'even' : this will select every even numbered record from the specified scans. (e.g., [2,4,6,8,...]).
- subscanlist = 'odd' : this will select every odd numbered record from the specified scans. (e.g., [1,3,5,7,...]).
- subscanlist = 'n/m' : this will select only record n from each group of m records. For example:
  - '1/4' will select records [1,5,9,13,...]
  - '3/4' will select records [3,7,11,15,...]

Arguments

scanlist	a vector of scan numbers Allowed:vector of integers
subscanlist	a vector of subscans or 'odd','even','n/m' Allowed:vector of integers Default: all
weighting	NONE, RMS, TSYS Allowed:string Default: RMS
alignment	NONE, VELOCITY, XAXIS Allowed:string Default: NONE

Returns
An SDRecord

Example

```
myavg := d.aver([2,3,4],[1,3]);  # myavg will contain the average of
                                 # subscans 1 and 3 in scans 2-4.
```

---

dish.avgpols - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Average the polarizations in the globalscan1 spectrum

Synopsis
avgpols()

Description
This function is used to average polarizations stored in the globalscan1 spectrum.
The result is stored in the globalscan1 array.

Returns

Boolean

Example

```
d.getc(35)
d.show()
d.avgpols()
d.show()
```

---

dish.base - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Perform a polynomial baseline fit, all in one function. An alternative step-by-step method is to use the nfit, setregion, bshape, and baseline functions

Synopsis
base(scanrec, order, range, action)

Description

Arguments

scanrec	SDRecord to fit Allowed:boolean Default: Currently displayed data
order	order of polynomial Allowed:integer Default: none
range	range for computing polynomial baseline Allowed:string Default: none
action	'subtract' or 'show' Allowed:string Default: 'subtract'

Returns
Boolean if action='show'
SDRecord if action='subtract'

Example

```
mybl := d.base(order=2,action='subtract',range='[50:400],[1500:2000]')
                                        # This will perform a 2nd order poly.
                                        # fit to the currently displayed scan.
```

---

dish.baseline - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Perform a polynomial baseline fit to spectrum in *globalscan1*

Synopsis
baseline()

Description
This function will use the values of *nfit* and *nregion* to fit and subtract a polynomial function from the spectrum in *globalscan1*. The result is then stored in the *globalscan1* spectrum. This operation can also be acheived using *d.base* with the action set to 'subtract'.

Returns
Boolean

Example

```
- d.nfit(2)
- d.setregion()
Use the left button to set location, right button to exit
region =  [23 152 300 644 807 1009]
- d.bshape()
- d.baseline()
```

---

dish.bdrop - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Set number of channels to drop from the beginning of a spectrum, used only in displaying.

Synopsis
bdrop(value)

Description
This function will restrict the viewing of channels to greater than bdrop when using the display function *show*. If no value is given, the current setting of bdrop is returned.

Arguments

value	The number of channels to drop Allowed:integer

Returns
Boolean

Example

```
- d.bdrop(100)
- d.edrop(100)
- d.show()
- d.bdrop()
100
```

---

dish.bias - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Add a bias to the scan in memory

Synopsis
bias(factor)

Description
This function will add a bias to the scan in memory. The bias should be a vector of the same shape as the scan's number of polarizations, or a single value to be applied to all polarizations.

Arguments

factor	The constant or vector to add Allowed:float or vector of floats Default: none

Returns
boolean (T if successful)

Description

Example

```
- d.plotc(159)
- d.bias([-2.0,-0.5]);
- d.show();
```

---

dish.boxcar - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Perform a boxcar smoothing

Synopsis

boxcar(smooth_width)

Description
This function will perform a boxcar smooth using the specified window in channels on the spectrum in *globalscan1*. This function automatically performs decimation. Use the *smooth* or *regrid* functions for more options.

Arguments

smooth_width	The smoothing width Allowed:integer Default: 3

Returns
Boolean

Example

```
- d.boxcar(5)
- d.show()
```

---

dish.bshape - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Show the baseline fit

Synopsis
bshape()

Description
This function will use the values of *nfit* and *nregion* to evaluate a polynomial fit on the spectrum in *globalscan1*. It will display this fit overlayed on the spectrum. This is effectively doing a *d.base* with the action set to 'show'. To subtract the fit after displaying it, use *d.baseline*.

Returns
Boolean

Example

```
- d.nfit(2)
- d.setregion()
Use the left button to set location, right button to exit
region =  [23 152 300 644 807 1009]
- d.bshape()
- d.baseline()
```

---

dish.cal - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Apply calibration to a scan (GBT only currently).

Synopsis
cal(scan, baseline, range, order, units, flipsr, fold, flipfold, proc)

Description
This function applies a calibration to the data based on information in the STATE table of the MS, in particular, it utilizes the OBSMODE column to determine a procedure type, switching signal, and switching state. In addition, information on the procedure size and procedure sequence number are used. This function is called by calib (with proc=T or proc=F).
The function applies a number of corrections:

1. determine elevation, frequency; solve for opacity,gain correction.

2. determine units used, solve for appropriate efficiency factor from table.

$$factor = \frac{e^{\tau(1/sin(elev))}}{\eta_x}$$

   where η_x is the efficiency factor determined by the user selected units.

3. based on procedure, calibrate data. Typically, this entails calibrating "cal on" and "cal off" phases of signal (on source) and reference (off source) observations, where the "cal" is a known noise diode. e.g.,

$$V_{sig}(\nu) = \frac{(V_{sig,on}(\nu)w_{sig,on} + V_{sig,off}(\nu)w_{sig,off})}{w_{sig,on} + w_{sig,off}}$$

Similarly for the reference observation, leading the difference spectrum:

$$T_{diff}(\nu) = < T_{sys,ref} > \frac{V_{sig}(\nu) - V_{ref}(\nu)}{V_{ref}(\nu)}$$

where, $< T_{sys,ref} >$ is:

$$< T_{sys,ref}(\nu) > = < \frac{T_{cal}(\nu)V_{ref}(\nu)}{V_{ref,on}(\nu) - V_{ref,off}(\nu)} >$$

Arguments

scan	scan number Allowed:integer Default: no default
baseline	toggle baseline subtraction on or off Allowed:boolean Default: F (off)
range	range for baseline Allowed:channel range vector Default: F (full range)
order	order of baseline fit Allowed:integer Default: 1
units	units for calibrated result; 1=TA*, 2=TMB, 3=S Allowed:integer Default: 1
flipsr	flip sense of signal and reference Allowed:boolean Default: F
fold	fold data - FSWITCH data only Allowed:boolean Default: F
flipfold	flip sense of fold - FSWITCH data only Allowed:boolean Default: F
proc	calibrate full procedure or single scan Allowed:boolean Default: T

Returns
T (if successful)

Example

```
- d.cal(9,proc=F); #calibrate only scan 9 regardless of whether it is part
                   #of a larger multi-scan procedure.
```

---

dish.calib - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Apply calibration to a scan (GBT only currently).

Synopsis
calib(scan, baseline, range, order, units, flipsr, fold, flipfold)

Description
This function applies a calibration to the data based on information in the STATE table of the MS, in particular, it utilizes the OBSMODE column to determine a procedure type, switching signal, and switching state. In addition, information on the procedure size and procedure sequence number are used.

This function calls cal.

Arguments

scan	scan number Allowed:integer Default: no default
baseline	toggle baseline subtraction on or off Allowed:boolean Default: F (off)
range	range for baseline Allowed:channel range vector Default: F (full range)
order	order of baseline fit Allowed:integer Default: 1
units	units for calibrated result; 1=TA*, 2=TMB, 3=S Allowed:integer Default: 1
flipsr	flip sense of signal and reference Allowed:boolean Default: F
fold	fold data - FSWITCH data only Allowed:boolean Default: F
flipfold	flip sense of fold - FSWITCH data only Allowed:boolean Default: F

Returns
T (if successful)

Example

```
- d.calib(9,baseline=T,order=5); #calibrate scan 9 and subtract a 5th order
                                 #polynomial
```

---

dish.clearrm - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Clear all variables from results manager

Synopsis
clearrm()

Description
This function removes all entries from the results manager.

Returns
T (if successful)

Example

```
- d.clearrm();
```

---

dish.clip - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Clip/Edit data (set flags)

Synopsis
clip(cliplevel, channelrange, clipdir)

Description
This function will flag all channels within the specified range, based on a threshhold value. Only channels in the given range will be considered for flagging.

Arguments

cliplevel	threshold for clipping Allowed:float Default: no default
channelrange	range of channels to flag Allowed:vector of integers Default: all
clipdir	clip direction – clip above ('high') or below ('low') the threshhold Allowed:string Default: high

Returns
Boolean

Example

```
- d.clip(0.4,[100:500]);
- d.clip(-.3,[100:300],'low');
```

---

dish.copy - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Copy a variable in the global record to another location

Synopsis
copy(fromhere, tohere)

Description
This function copies a named variable in the global record to another variable name. For example, this can be used to copy the contents of offscan1 into the globalscan1 record.

Arguments

fromhere	Current variable name Allowed:string Default: no default
tohere	Desired variable name Allowed:string Default: no default

Returns
Bool

Example

```
- d.uniput('echan',100);
- d.upr('echan')
echan  =  100
- d.copy('echan','emoment')
- d.upr('emoment')
emoment  =  100
```

---

dish.dcbase - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Subtract a DC baseline (0th order poly) from the spectrum

Synopsis
dcbase()

Description
This function will subtract a 0th order polynomial from the spectrum in *globalscan1*. Ther range of channels used for the baseline determination is set by either d.nregion or d.setregion.

Returns
Boolean

Example

```
- d.dcbase()
```

---

dish.delete - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Delete a single entry from the stack.

Synopsis
delete()

Description
This function is used to remove an entry from the "stack". The stack is a construct (simply a vector of integers) which is used to organize a list of scans which will be operated on in some common way. Generally, it is used to assemble a list of scans which can then be averaged.

Returns
Boolean

Example

```
- d.addstack(10,50,2)
- d.tellstack()
21  entries in the stack.
[10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50]
- d.delete(22)
- d.tellstack()
20  entries in the stack.
[10 12 14 16 18 20 24 26 28 30 32 34 36 38 40 42 44 46 48 50]
```

---

dish.divide - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Divide globalscan1 by offscan1.

Synopsis
divide()

Description
This function will divide the contents of globalscan1 by offscan1.

Example

```
- d.divide();
```

Returns
Boolean

---

dish.done - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Exits dish and destroys the current dish tool.

Synopsis
done()

Description

Example

```
- d.done()
```

Returns
Boolean

---

dish.edrop - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Set number of channels to drop from the end of a spectrum, used only in displaying.

Synopsis
edrop(value)

Description
This function will restrict the viewing of channels to less than edrop when using the display function *show*. If no value is given, the current setting of edrop is returned.

Arguments

value	The number of channels to drop Allowed:integer

Returns
Boolean

Example

```
- d.bdrop(100)
- d.edrop(100)
- d.show()
- d.edrop()
100
```

---

dish.empty - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Clear the stack.

Synopsis
empty()

Description
This function is used to clear the "stack". The stack is a construct (simply a vector of integers) which is used to organize a list of scans which will be operated on in some common way. Generally, it is used to assemble a list of scans which can then be averaged.

Returns
Boolean

Example

```
- d.addstack(10,50,2)
- d.tellstack()
21  entries in the stack.
[10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50]
- d.empty()
- d.tellstack()
No entries currently in the stack.
```

---

dish.filein - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Sets the scan group used by scan access functions.

Synopsis
filein(wsname)

Description
Several scan groups (Measurement sets or saved data) can be available in the

results manager at once. This function sets which of those available functions is used when scan access routines (getr, getc, getscan) are called. If a scan group is not in the results manager yet but has been filled and saved to disk, it can be opened with the d.open command.

Arguments

wsname	scan group name; must already be 'opened' (exist in the Result s Manager) Allowed:string Default: no default

Returns
T (if successful)

Example

```
- d.filein('dishdemo2');
```

---

dish.fileout - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Sets the scan group that can be written to.

Synopsis
fileout(sgname)

Description
This specifies a scan group to which spectra may be saved. If the scan group doesn't already exist in the results manager, fileout will create it. If the file doesn't exist in the results manager but is on disk, the user can load it with the appropriate permissions for writing as:
d.open(filename,access="w")
If the user wants to create a new blank scan group for subsequent storage, specify: d.fileout(filename) where filename is a unique disk file name.

Arguments

sgname	scan group name Allowed:string Default: no default	

Returns
T (if successful)

Example

```
- d.fileout('testfileout');
New file  testfileout  is created
- d.save(); # saves the currently viewed plot to testfileout
```

---

dish.files - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Prints the currently set values for filein and fileout.

Synopsis
files(quiet)

Description
Shows the current values for the filein and fileout settings. The filein setting indicates which data are accessed with getc, getr, and getscan. The fileout setting indicates which scan group is written to by the save and keep functions. If the parameter quiet is F, a glish record with this information is returned. If quiet=T, the information is printed to the screen.

Arguments

quiet	suppress printing Allowed:boolean Default: F	

Returns
a glish record with two values: filein and fileout (if quiet=F)

Example

```
- d.open('ohtest_SP')
- d.fileout('testout')
NORMAL: New file testout is created
- d.files()
Current filein is  :  ohtest_SP1
Current fileout is :  testout
T
```

---

dish.find - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Search all glish variables for a specified string.

Synopsis
find(myfoo)

Description

Arguments

myfoo	string to look for in all currently defined glish variables Allowed:string

Returns
T (if successful)

Example

```
- d.find('smo')    # returns any variables that have 'smo' in them
dishsmooth dishsmoothgui
```

---

dish.flag - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Flag data based on various specifications.

Synopsis
flag(scan, ints, polarization, channel, flag)

Description
This function allows a simple non-GUI means of flagging data; the viewer and msplot allow GUI flagging of data. If no scan is specified, it will flag all scans in the data according to the other parameters (e.g., all channels within a range can be flagged in this way). If no integration is specified, it will flag all integrations. If no polarization is specified, it will flag all polarizations. If no channel is specified, it will flag all channels. If you wish to unflag data, use flag=F.

Arguments

scan	scan number(s) to flag Allowed:integer or vector of ints Default: F
ints	integration number Allowed:integer or vector of ints Default: F
polarization	polarization number(s) to flag Allowed:integer or vector of ints Default: F
channel	channel (range) to flag Allowed:integer or vector of ints Default: F
flag	flag (T) or unflag (F) Allowed:boolean Default: T

Returns
T (if successful)

Example

```
d.open('stjul13SP');
d.gms();
d.plotc(36)
d.flag(36,channel=100:120)      #flag all subscans, polarizations of scan
                                #36 in the channel range of 100:120
d.plotc(36)                     #you should see the range masked
d.flag(36,flag=F);              #unset all flagging
d.plotc(36)                     #it's back!
d.flag(36,channel=500:550,polarization=1)
d.plotc(36)                     #you should see a gap just in the first
                                #polarization
d.flag(36,flag=F);
d.flag(36,1)                    #flag just the first integration
d.plotc(36)                     #should look okay though it doesn't
                                #include the first integration
d.plotc(36,1)                   #masked
d.plotc(36,2)                   #okay
```

---

dish.galactic - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Provide coordinates of the globalscan1 spectrum in the Galactic reference frame.

Synopsis
galactic(quiet)

Description

Arguments

quiet	indicate whether the results should be printed (quiet=F) or returned as a glish record (quiet=T) Allowed:boolean Default: F

Returns
T (if successful)

Example

```
- d.galactic();
1.91025363 deg 41.4198054 deg GALACTIC
```

---

dish.gauss - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Fit gaussian profiles.

Synopsis
gauss(ngauss, guess, prompt, pol, bchan, echan, plotfit, plotresid, ploteach, ret)

Description
This function is used to fit one or more gaussian curves to the spectrum in the globalscan1 array. The gaussian fit function must have initial guesses for the center, width and height of the features being fit. You can allow the function to guess these parameters on its own, you can enter initial guesses as a parameter, or you can enter guesses interactively using the mouse cursor. You can also restrict the range of the spectrum to be fit.
The *ngauss* parameter specifies the number of gaussians to be fit simultaneously. The *guess* parameter is an optional array specifying initial guesses for the height, center and width of each component to be fit. See the example below to get a feel for the format required to enter initial guesses using this parameter. The most convenient way to provide initial guesses for the gaussian parameters, though, is via the mouse cursor. To enter initial

guesses in this way, set the parameter *prompt* to T when calling the gauss function. You will then be prompted in the plotter message area to click on the plot to specify the range of values to be fit and the locations of the features.

If your spectrum has more than one polarization shown, use the *pol* parameter to specify which polarization to fit. Only one is fit at a time. The *bchan* and *echan* parameters are used to restrict the range of data used for the fit. These are useful for fitting components which are not dominant in a spectrum. The units for the *bchan* and *echan* values must be specified in pixels, but the plot should not be shown in pixels when the function is called. Range limits can alternatively be specified interactively by setting the parameter *prompt* to T.

The remaining parameters are flags. Set *plotfit* to T (the default) if you want to see the fit plotted on the data. Set *plotresid* to T to see residuals. Set *ploteach* to T if you have a multicomponent fit (that is, *ngauss* ¿ 1) and you want to see each component's fit plotted individually. Finally, set *ret* to T if you want the function to return a glish record containing the results of the fit.

Arguments

ngauss	number of gaussians to fit Allowed:integer Default: no default
guess	array with the guesses for height,center, and width Allowed:float Default: function will guess if none given
prompt	prompt for mouse entry of limits and initial guesses Allowed:boolean Default: F
pol	polarization number to use Allowed:integer Default: 1
bchan	beginning channel number (pixel) of range to be fit Allowed:integer Default: 1
echan	ending channel number (pixel) of range to be fit Allowed:integer Default: full range
plotfit	flag to show fit on plot Allowed:boolean Default: T
plotresid	flag to show residuals on plot Allowed:boolean Default: F
ploteach	flag to show plots of individual components on plot Allowed:boolean Default: F
ret	return fit components as a glish array? Allowed:boolean Default: F

Returns

boolean (if ret=F) or a glish record (if ret=T)

Example

```
- d.gauss(1); # fit a single gaussian using auto-guess values
```

Example

```
- # fit a gaussian using an initial guess and restricting the channel range
- myguess:=array(0,2,3)   # initialize an array which will contain the guesses
```

```
- myguess[1,1]:= 1.2      # height of component 1
- myguess[1,2]:= 110.2    # center of component 1
- myguess[1,3]:= 0.2      # width of component 1
- myguess[2,1]:= 0.6      # height of component 2
- myguess[2,2]:= 114.8    # center of component 2
- myguess[2,3]:= 0.2      # width of component 2
- d.gauss(1,myguess,bchan=1100,echan=1300);
```

Example

```
- d.gauss(1,prompt=T); # prompt for interactive entry of range limits and
-                      # initial guesses
```

Example

```
- result := d.gauss(2,ret=T)
Gauss:  1
Center: 2.216355e+01   Height: 1.116988e-02    Width: 1.457546e-04
C-err : 1.172879e-06   H-err : 1.840365e-04    W-err: 2.926720e-06
Gauss:  2
Center: 2.216288e+01   Height: 3.880797e-03    Width: 7.887906e-04
C-err : 8.028110e-06   H-err : 8.039411e-05    W-err: 2.059391e-05
- print result.h
[0.011169876 0.00388079719]
- print result.c
[22.1635453 22.1628813]
- print result.w
[0.000145754649 0.000788790639]
- print result.herr
[0.000184036544 8.03941075e-05]
- print result.cerr
[1.17287918e-06 8.02811046e-06]
- print result.werr
[2.92671975e-06 2.05939141e-05]
```

---

dish.gbtsum - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Summarize scans in GBT projects from the /home/gbtdata area (GB only)

Synopsis
gbtsum(projname, source, bscan, escan, verbose, filename)

Description
This function is used to list a scan summary of data on disk without the need to fill that data first. It works only in Green Bank, for GBT data. The directory prefix /home/gbtdata/ is not necessary in the project name. Output can be sent to an ASCII file if desired.

Arguments

projname	the project name Allowed:string Default: no default
source	source name Allowed:string Default: all
bscan	beginning scan to list Allowed:int Default: first
escan	end scan to list Allowed:int Default: last
verbose	list extra information? Allowed:boolean Default: F
filename	optional name of ASCII file to write Allowed:string Default: none

Returns
Boolean

Example

```
- d.gbtsum('standards_jul13')
```

---

dish.getc - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Retrieve calibrated data (from the CORRECTED_DATA column)

Synopsis
getc(scan, int, pol, nif)

Description
This provides data access directly to the CORRECTED_DATA column. If the scan is specified and no integrations, the data is averaged with a TSYS weighting scheme (weighting is equal to the integration time divided by the square of the system temperature).

Arguments

scan	the scan number Allowed:integer Default: no default
int	the integration number Allowed:integer Default: All integrations
pol	the polarization number Allowed:boolean Default: All polarizations
nif	the IF number Allowed:integer Default: 1

Returns
An SDRecord

Example

```
- myrec := d.getc(9); #will average all of the calibrated data from scan 9
```

---

dish.getr - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Retrieve raw data (from the FLOAT_DATA column)

Synopsis
getr(scan, phase, int, pol, nif)

Description
This provides data access directly to the FLOAT_DATA column. If the scan is specified and no integrations, the data is averaged with a TSYS weighting scheme (weighting is equal to the integration time divided by the square of the system temperature).

Arguments

scan	the scan number Allowed:integer Default: no default
phase	the phase number Allowed:integer Default: no default
int	the integration number Allowed:integer Default: All integrations
pol	the polarization number Allowed:boolean Default: All polarizations
nif	the IF number Allowed:integer Default: 1

Returns
An SDRecord

Example

```
- myrec := d.getr(9,1); #will average all of the phase 1 data from scan 9
```

dish.getscan - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Retrieve a scan (SDRecord) from a scan group.

Synopsis
getscan(scannumber, subscan, resync, setgs)

Description
This is a data access utility for retrieving a single record from the group. It is useful for exploring uncalibrated data in detail. The getc function should be used instead for general scan retrieval of calibrated scans.

Arguments

scannumber	the scan number Allowed:integer Default: no default
subscan	the subscan number Allowed:integer Default: 1
resync	update the cache Allowed:boolean Default: F
setgs	copy the result to the globalscan array? Allowed:boolean Default: T

Returns
An SDRecord

Example

```
- myrec := d.getscan(422,2);
```

---

dish.getvf - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Prints the velocity and frequency for a specified channel.

Synopsis
getvf(chan, dop, refframe)

Description
This function will print the value of the velocity and frequency for a specified channel (units of km/s and Hz) of the currently displayed spectrum.

Arguments

chan	channel of interest Allowed:integer Default: no default
dop	Doppler convention Allowed:RADIO, OPTICAL or TRUE Default: RADIO
refframe	Reference Frame Allowed:LSRK, LSRD, BARY, GEO, TOPO or GALACTO Default: LSRK

Returns
Boolean

Example

```
- d.getvf(1)
Channel  1     RADIO LSRK 1.40984264e+09 46301.7957
T
- d.getvf(1,'OPTICAL','BARY')
Channel  1     OPTICAL BARY 1.40992869e+09 54737.4952
T
```

---

dish.getvfarray - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Return the velocities and frequencies for the globalscan1 spectrum.

Synopsis
getvfarray(dop, refframe)

Description
This function will return vectors of the velocities and frequencies for the currently viewed spectrum.

Arguments

dop	Doppler convention Allowed:RADIO, OPTICAL or TRUE Default: RADIO
refframe	Reference Frame Allowed:LSRK, LSRD, BARY, GEO, TOPO or GALACTO Default: LSRK

Returns
Record with four fields: v (array of velocities), f (array of frequencies), refframe, and doppler

Example

```
- a := d.getvfarray('OPTICAL','BARY')
- a.f[1]
1.40992869e+09
- a.v[1]
54737.4952
- a.refframe
BARY
- a.doppler
OPTICAL
```

---

dish.gms - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Summarize the scans in the currently opened data set.

Synopsis
gms(verbose)

Description
This will provide information on the MeasurementSet opened in DISH. By default, it will display only the first in a sequence of scans for a given procedure. The verbose flag allows the user to specify viewing of all scans.

Arguments

verbose	indicate level of output Allowed:boolean Default: F

Returns
boolean

Example

```
- d.gms();
Scan     Object   Proctype    SWState   SWtchsig   Procseqn   Procsize
   5       S171      OffOn PSWITCHOFF     TPWCAL          1          2
   7       S171      OffOn PSWITCHOFF     TPWCAL          1          2
   9       S171      Track    FSWITCH      FSW01          1          1
  15     OrionA  RALongMap    FSWITCH      FSW01          1         31
T
```

---

dish.gsflag - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Flag some channels in the globalscan1 spectrum

Synopsis
gsflag(pol, bchan, echan, flagval)

Description
This function is used to flag a channel range in the globalscan1 spectrum. Channels can be specified with the mouse cursor or with a parameter in the function call line. Also see the *clip* function for other flagging options.
If no value is passed for the bchan and echan parameters, the function will prompt you for input from the mouse to specify a flagging range.

Arguments

pol	polarization to flag; 0 means flag all Allowed:integer Default: 0
bchan	beginning channel in range Allowed:integer Default: F
echan	ending channel in range Allowed:integer Default: bchan
flagval	flag (T) or unflag (F) Allowed:boolean Default: T

Returns
boolean

Example

```
- d.gsflag()   # interactive flagging of all displayed polarizations
```

Example

```
- d.gsflag(pol=1,bchan=123)   # flag a single channel in a single polarization
```

Example

```
- d.gsflag(bchan=100,echan=200,flagval=F)   # unflag the specified range
```

---

dish.gsget - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Retrieves the globalscan1 record to a glish variable

Synopsis
gsget()

Description
This function is used to retrieve the globalscan1 record into a glish variable. It is useful to more advanced DISH users who wish to access and modify the sdrecord directly.

Example

```
- xx := d.gsget()             # get the globalscan1 record
- dd.data.arr[1,10:20] := 0   # modify the data array
- d.gsput(xx)                 # return the modified data to globalscan1
- d.show()                    # see the result
```

Returns
sdrecord

---

dish.gsput - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Sends an sdrecord from glish to the globalscan1 record

Synopsis
gsput(sdrec)

Description
This function is used to place a glish value into the globalscan1 record. The entry must be a valid sdrecord. This function is useful to more advanced DISH users who wish to access and modify the sdrecord directly.

Arguments

sdrec	sdrecord to place into globalscan1 Allowed:sdrecord

Example

```
- xx := d.gsget()             # get the globalscan1 record
- dd.data.arr[1,10:20] := 0   # modify the data array
- d.gsput(xx)                 # return the modified data to globalscan1
- d.show()                    # see the result
```

Returns
boolean

---

dish.gsread - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Retrieve a spectrum saved to disk into the globalscan1 record

Synopsis
gsread(file)

Description
Use this function to retrieve a globalscan1 record from a disk file. The globalscan1 value must have been saved previously using d.gswrite. This function may be useful for storing and organizing individual spectra. Also see the *save* function and its associated retrieval functions.

Arguments

file	filename Allowed:string	

Example

```
- d.show()   # You should already have some spectrum available for this example
- d.gswrite('ngc1068_before_baseline')  # save the preliminary result
- d.baseline()                          # operate on the globalscan1 data
- d.show()
- d.gsread('ngc1068_before_baseline')   # retrieve the preliminary result
- d.show()
```

Returns
sdrecord

---

dish.gswrite - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Write the globalscan1 spectrum to disk

Synopsis
gswrite(file)

Description
Use this function to write the globalscan1 to disk. It can be retrieved at a later time using the *gsread* function. This function may be useful for storing and organizing individual spectra. Also see the *save* function.

Arguments

file	filename Allowed:string	

Example

```
- d.show()   # You should already have some spectrum available for this example
- d.gswrite('ngc1068_before_baseline')  # save the preliminary result
- d.baseline()                          # operate on the globalscan1 data
- d.show()
- d.gsread('ngc1068_before_baseline')   # retrieve the preliminary result
- d.show()
```

Returns
boolean

---

dish.gui - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Toggles mapping of DISH main gui (results manager).

Synopsis
gui()

Description

Example

Returns
Bool

---

dish.hanning - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Perform a hanning smooth

Synopsis
hanning()

Description
This function will perform a hanning smooth on the globalscan1 spectrum. *hanning* utilizes the underlying *smooth* and *regrid* functions. Please see these functions for more options.

Returns
Boolean

Example

```
- d.hanning()
- d.show()
```

---

dish.header - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Print formatted header information to the command line window

Synopsis
header() ;

Description
This function provides basic header information on the globalscan1 spectrum.

Arguments

Returns
T (if successful)

Example

```
- d.header()
Proj : A40_3            Src  : S6               Proc : Track
Obs  : Lockman          RA   : 15:31:33.712     PType: SIMPLE
Scan : 303              Dec  : -002.25.23.477   OType: LINE
Seq  : 1/1              Epoch: J2000            Swtch: FSWITCH
Date : 2002-07-03       Az   : -179.087         Swsig: FSW01
Time : 02:09:41.983 UT  El   : 49.320           Ints : 2

Tsys : 1.41             Trx  : 7.46             Tcal : 1.62
Tsys : 1.41             Trx  : 13.76            Tcal : 1.66

BW   : 4.995    (MHz)   Res  : 4.883    (kHz)
Expos: 29.914           Durat: 119.931
T
```

---

dish.help - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Provide simple help on dish functions

Synopsis
help(dishfn, driveweb)

Description
This function provides simple help on the specified **dish** function. In addition, it will optionally drive the web browser to the location of the more extended help.

Arguments

dishfn	Name of dish function Allowed:String Default: no default
driveweb	Drive the browser to the URM help Allowed:Boolean Default: F

Returns
T (if successful)

Example

```
- d.help('calib')
dish.dish.dish.calib.function

 calib -- Function -- dish.dish.dish
  Apply calibration to a scan (GBT only currently).
 Useage:  calib(scan, baseline, range, order, units)

You may find more information in the on-line documentation available
via your web browser.  Type the command

   web()

to view more about dish.dish.dish.calib.function.

T
```

---

dish.history - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Add a string or vector of strings to the history of a current SDRecord.

Synopsis
history(scanrec, history)

Description

Arguments

scanrec	SDRecord to add history to Allowed:SDRecord Default: no default
history	text to add to history Allowed:string Default: no default

Returns
T (if successful)

Example

```
- d.history(myrec,'1.5*(on-off)/off');
```

---

dish.imagems - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Make an image from all scans in a MeasurementSet.

Synopsis
imagems(msname, start, stop, step, imname, spwid, nx, ny, cellx, celly, gridfn, convsupport)

Description
This command is a more general facility than its complement **makeim**. Where **makeim** will operate only on the scans for a given procedure, **imagems** will operate on all the scans found within a MeasurementSet; this is particularly useful for creating a data cube from several mapping procedures around the same region. Like **makeim**, it will construct a data cube (position-position-frequency). It uses the **imager** tool to achieve this. The user selects the MeasurementSet into which all of the mapping scans have

been filled, specifying the channel range over which to produce the cube (start, stop). An optional averaging is offered through the step argument. The cell size can also be optionally specified with the number and units; both cellx and celly must be specified to over-ride the default calculated value (e.g., cellx='4arcmin',celly='4arcmin'). The default gridding function is the boxcar (BOX); PB (primary beam) and SF (prolate spheroidal) may also be specified. For the BOX and SF gridding functions, the size of function can be specified through the convsupport argument. The units of convsupport are pixels in the resulting image and this specifies a radial size of the convolving function. For BOX, a convsupport of 0 (the default) corresponds to nearest cell gridding. A convsupport of 1 is boxcar of width 3 (-1, 0, and +1). For SF gridding, the default size is 1 a radius of 1 pixel. Larger values may produce images with better noise characteristics at the expense of a loss of spatial resolution. There is no gridding along the frequency axis (data are mearly averaged into the nearest frequency channel as they are gridded on to the image). The default value is chosen when convsupport has a boolean or non-numeric value.

Arguments

msname	MeasurementSet name Allowed:string Default: no default
start	starting channel Allowed:integer Default: no default
stop	stopping channel Allowed:integer Default: no default
step	number of channels to average over for final cube Allowed:integer Default: 1
imname	name of output image Allowed:string Default: scanimage
spwid	spectral window id number Allowed:integer Default: F
nx	number of cells in x Allowed:integer Default: F
ny	number of cells in y Allowed:integer Default: F
cellx	cell size in x Allowed:string Default: F
celly	cell size in y Allowed:string Default: F
gridfn	gridfunction to use; BOX, PB or SF Allowed:string Default: BOX
convsupport	convolution support radius for BOX or SF Allowed:integer Default: F

Returns

T (if successful)

Example

```
- d.imagems('cloud_SP',520,1000,imname='cloud.im');
```

---

dish.imcal - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Calibrate an imaging/mapping procedure

Synopsis
imcal(scan, baseline, range, order, units)

Description
This uses the calib function to calibrate the individual scans in an imaging or mapping procedure (currently RALongMap or DECLatMap).

Arguments

scan	scan number Allowed:integer Default: no default
baseline	toggle baseline subtraction on or off Allowed:boolean Default: F
range	range for baseline Allowed:string Default: F(full range)
order	order of baseline fit Allowed:integer Default: 1
units	units for calibrated result Allowed:integer Default: 1=TA*,2=TMB,3=S

Returns
T (if successful)

Example

```
- d.imcal(15,baseline=T,order=5);#calibrate all scans in the mapping sequence
 #beginning with scan 15;subtract a 5th order
                                #polynomial
```

---

dish.import - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Import GBT data into an AIPS++ MeasurementSet

Synopsis
import(projdir, outms, outmsdir, startscan, stopscan, backend, calflag)

Description
This function writes an AIPS++ MeasurementSet from a GBT FITS directory tree, and loads the MS into DISH (via open). (NOTE: This function is exclusive to GBT data). The MeasurementSet created on disk will default to the input project name with an appended '_' followed by a backend specifier. If an 'outms' argument is supplied, it will be this string with the appended underscore and backend specified.

Arguments

projdir	project directory Allowed:string Default: no default
outms	output MeasurementSet Allowed:string Default: name derived from projdir
outmsdir	directory for output MS Allowed:string Default: current directory
startscan	starting scan to begin filling Allowed:integer Default: first scan in projdir
stopscan	ending scan for filling Allowed:integer Default: last scan in projdir
backend	which backend to fill Allowed:string Default: all
calflag	toggle to permit automated calibration upon filling Allowed:boolean Default: F

Returns
T (if successful)

Example

```
- d.import('standards\_apr27',outms='demo',startscan=5,stopscan=45);
T
```

This creates a file 'demo_SP' on disk for spectral processor data, or 'oh_SPECTROMETER_A' for the spectrometer (the additional _A designates which bank of the spectrometer was used - if only one bank, it is 'A').

---

dish.keep - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Saves the current globalscan1 SDRecord to the designated fileout.

Synopsis
keep()

Description
Stores the current *globalscan1* record in the output file. This is similar to *d.save*, however the scan to be saved does not to be specified explicitly. *d.keep* saves whatever is currently in *globalscan1*).

Arguments

Returns
boolean (T, if successful)

Example

```
- d.files();
Current filein is  :  jaytestSP
Current fileout is :  testfileout
- d.plotc(303)
- d.keep()
globalscan saved to  testfileout
```

---

dish.listscans - Function

Package	**dish**
Module	**dish**
Tool	**dish**

List the scan numbers from the active scan group.

Synopsis
listscans()

Description
Lists scan numbers. Equivalent to d.lscans().

Returns
a vector of scan numbers

Example

```
- d.listscans();
[398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417
 418 419 420 421 422 423 426 429 432 435 437 440 443 446 449 452 455 458 463]
```

---

dish.logcommand - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Writes command to scripter.

Synopsis
logcommand()

Description
Writes a command to the scripter. logcommand uses the same syntax as history but the information is logged to the scripted rather than to the record's history.

Example

Returns
Bool

---

dish.lsoutfile - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Lists scans in the fileout scangroup

Synopsis
lsoutfile()

Description
This simply lists the scans contained in the fileout data set.

Returns
vector of scan numers

Example

```
- d.files();
Current filein is  :  jaytestSP
Current fileout is :  testfileout
T
- d.plotc(303)
- d.keep()
- d.lsoutfile()
303
```

---

dish.makeim - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Make an image from scans in a MeasurementSet.

Synopsis
makeim(scan, start, stop, step, imname, spwid, nx, ny, cellx, celly, gridfn)

Description
This command will construct a data cube (position-position-frequency) from a mapping procedure. It uses the **imager** tool to achieve this. The user selects the starting scan of a mapping procedure and the channel range over which to

produce the cube (start, stop). An optional averaging is offered through the step argument. The cell size can also be optionally specified with the number and units; both cellx and celly must be specified to over-ride the default calculated value (e.g., cellx='4arcmin',celly='4arcmin'). The default gridding function is the boxcar (BOX); PB (primary beam) and SF (polate spheroidal) may also be specified.For the BOX and SF gridding functions, the size of function can be specified through the convsupport argument. The units of convsupport are pixels in the resulting image and this specifies a radial size of the convolving function. For BOX, a convsupport of 0 (the default) corresponds to nearest cell gridding. A convsupport of 1 is boxcar of width 3 (-1, 0, and +1). For SF gridding, the default size is 1 a radius of 1 pixel. Larger values may produce images with better noise characteristics at the expense of a loss of spatial resolution. There is no gridding along the frequency axis (data are mearly averaged into the nearest frequency channel as they are gridded on to the image). The default value is chosen when convsupport has a boolean or non-numeric value.

Arguments

scan	scan number Allowed:integer Default: no default
start	starting channel Allowed:integer Default: no default
stop	stopping channel Allowed:integer Default: no default
step	number of channels to average over for final cube Allowed:integer Default: 1
imname	name of output image Allowed:string Default: scanimage
spwid	spectral window id number Allowed:integer Default: F
nx	number of cells in x Allowed:integer Default: F
ny	number of cells in y Allowed:integer Default: F
cellx	cell size in x Allowed:string Default: F
celly	cell size in y Allowed:string Default: F
gridfn	gridding function Allowed:'BOX', 'PB', or 'SF' Default: BOX

Returns
T (if successful)

Example

```
- d.makeim(15,250,750);
```

---

dish.message - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Writes text to the DISH GUI message line.

Synopsis
message(message)

Description
This function allows the user to write messages to the DISH message line, found on the main DISH GUI just below the Results Manager. This is mostly useful for custom user applications, to prompt or proffer information.

Example

```
- d.message('This is a test message');
```

Arguments

message	The message to send to DISH message line Allowed:string Default: no default

Returns
Boolean

---

dish.minus - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Subtract offscan1 from globalscan1.

Synopsis

minus()

Description
This function will subtract the contents of offscan1 from globalscan1. The contents are stored in globalscan1.

Example
- d.minus();

Returns
Boolean

---

dish.mscal - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Calibrate a MeasurementSet.

Synopsis
mscal(baseline, range, order, units)

Description
This function will use the **calib** function to calibrate each individual *known* procedure within a MeasurementSet. It will use the MS currently pointed to/opened in DISH (use d.files() to see what this is).

Arguments

baseline	toggle to permit baselines to be subtracted Allowed:boolean Default: F
range	the range for the baseline Allowed:string Default: F(full range)
order	the polynomial order for the baseline Allowed:integer Default: 1
units	the units of the calibrated output Allowed:integer Default: 1(TA*)

Returns
T (if successful)

Example

```
- d.mscal();
```

---

dish.mult - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Scales (multiplies) a scan by a value.

Synopsis
mult(factor)

Description

Arguments

factor	The factor to multiply by Allowed:float Default: no default

Example

```
- newrec := d.mult(2.5);
```

Returns
SDRecord

---

dish.multiply - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Multiply globalscan1 by offscan1.

Synopsis
multiply()

Description
This function will multiply the contents of offscan1 from globalscan1 and store the result in globalscan1.

Example
- d.multiply();

Returns
Boolean

---

dish.nfit - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Set the order of the polynomial for baseline subtraction.

Synopsis

nfit(value)

Description
This function sets the order of the polynomial for baseline subtraction. It is used by the *bshape* and *baseline* functions. Use d.nfit() with no arguments to query the current value.

Arguments

value	The order of the polynomial Allowed:integer Default: none

Example

```
- d.nfit(3);
- d.upr('nfit')
nfit  =  3
- d.nfit()
3
```

Returns
Boolean

---

dish.nogui - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Dismisses the DISH main GUI.

Synopsis
nogui()

Description
This function dismisses the main DISH GUI. It can be turned on again with d.gui().

Example
- d.nogui()

Returns
Boolean

---

dish.nregion - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Set the channel ranges for baseline subtraction.

Synopsis
nregion(...)

Description
This function sets the channel ranges for fitting polynomials in baseline subtraction. It is used by the *bshape* and *baseline* functions. It offers a direct (non-interactive) means of setting these ranges. *setregion* provides an alternative, interactive way of setting these ranges using the mouse cursor. *nregion* accepts a variable number of arguments. The arguments list the first and last channel of each segment of the range to be used in the polynomial fit. The total number of arguments should thus be a multiple of 2.

Arguments

...	channel numbers to demark a range Allowed:integers

Example

```
- d.nfit(3);
- d.nregion(10,100,400,512)          # use channels 10-100 and 400-512 for the fit
- d.bshape()
- d.nregion(10,100,120,130,400,512)  # use channels 10-100, 120-130 and 400-512 for the fit
- d.bshape()
```

Returns
Boolean

---

dish.open - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Load a scan group from disk into DISH and perform a filein on it.

Synopsis
open(file, access, new, corrdata, filein)

Description
This command loads pre-existing datasets on disk into the results manager for viewing/manipulation. It can also be used to create a new, blank dataset for storing results (d.open('filename',new=T)).

Arguments

file	File on disk to be opened. This file becomes a glish variable in the Results Manager Allowed:string Default: no default
access	permissions for file ('r' readonly or 'w' for writable Allowed:string Default: 'r'
new	Should it create an empty file on disk (for saving to) Allowed:boolean Default: F
corrdata	look at CORRECTED_DATA column? Allowed:boolean Default: T
filein	should this dataset be the default filein Allowed:boolean Default: T

Returns
T (if successful)

Example

```
- d.open('dishdemo2');
```

---

dish.page - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Clears the DISH plotter display.

Synopsis
page()

Description
This function is a shortcut to clear the plotter display.

Example

```
- d.page();
```

Returns
Boolean

---

dish.peak - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Obtain the peak amplitude, half-width and center in the displayed spectrum

Synopsis
peak(quiet)

Description
This function measures the peak amplitude and associated half-width and line center for the globalscan1 spectrum. If quiet=T that results are stored in internal variables appropriate for use with the gaussian fitter. If quiet=F (default) the values are displayed to the screen.

Arguments

quiet	Display the output to command window Allowed:boolean Default: F	

Example

```
- d.peak()
Center= 26 Hwidth= 2 Height= 41.6090736
Center= 26 Hwidth= 2 Height= 46.4903967
T
```

Returns
Boolean

---

dish.plotc - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Plot calibrated data (from the CORRECTED_DATA column)

Synopsis
plotc(scan, int, pol, nif)

Description
This provides quick plotting of the CORRECTED_DATA column. If the scan is specified and integrations are not, the data is averaged with a TSYS weighting scheme (weighting is equal to the integration time divided by the square of the system temperature).

Arguments

scan	the scan number Allowed:integer Default: no default	
int	the integration number Allowed:integer Default: All integrations	
pol	the polarization number Allowed:boolean Default: All polarizations	
nif	the IF number Allowed:integer Default: 1	

Returns

boolean

Example

```
- d.plotc(9)   #will average all calibrated data from scan 9 and display it
```

---

dish.plotr - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Plot raw data (from the FLOAT_DATA column)

Synopsis

plotr(scan, phase, int, pol, nif)

Description

This provides quick plotting of the FLOAT_DATA column. If the scan is specified and integrations are not, the data is averaged with a TSYS weighting scheme (weighting is equal to the integration time divided by the square of the system temperature).

Arguments

scan	the scan number Allowed:integer Default: no default
phase	the phase number Allowed:integer Default: no default
int	the integration number Allowed:integer Default: All integrations
pol	the polarization number Allowed:boolean Default: All polarizations
nif	the IF number Allowed:integer Default: 1

Returns
boolean

Example

```
- d.plotr(9,1)
```

---

dish.plotscan - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Plot an SDRecord.

Synopsis
plotscan(scanrec, overlay)

Description

Arguments

scanrec	An SDRecord to be plotted Allowed:SDRecord Default: no default
overlay	overlay (T) or don't overlay(F) Allowed:boolean Default: F

Returns
T (if successful)

Example

```
- myspec := d.getc(2)
- d.plotscan(myspec);
- d.plotscan(d.getscan(3));
```

---

dish.plotxy - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Plot two vectors in the DISH plotter.

Synopsis
plotxy(xarray, yarray, newplot)

Description
A simple tool for making plots in the dish plotter. More versatile access to the plotter, with full pgplot functions, is available using the d.plotter functionality, e.g. d.plotter.plotxy(1:100,sin(1:100)). Run d.fields(d.plotter) for a complete listing of functions.

Arguments

xarray	vector of values for the x axis Allowed:vector of floats Default: no default
yarray	vector of values for the y axis Allowed:vector of floats Default: no default
newplot	Create a new plot(T) or overlay this(F) Allowed:boolean Default: T

Returns
T (if successful)

Example

```
- x:=1:100
- y:=sin(x)
- d.plotxy(x,y);
```

---

dish.plus - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Add offscan1 to globalscan1.

Synopsis
plus()

Description
This function will add the contents of offscan1 from globalscan1 and store the result in globalscan1.

Example
- d.plus()

Returns
Boolean

---

dish.qscan - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Query characteristics of scan.

Synopsis
qscan(scan)

Description
Capability to obtain more detailed information on a particular scan. The function returns a record with the following fields:

```
rows - number of rows in the MeasurementSet
phases - number of distinct phases in the scan
ifs     - number of ifs in the scan
fields - number of fields in the scan
ints - number of integrations in the scan (sequences of all phases,ifs, etc).
rns - the row numbers in the MAIN table of the MeasurementSet
```

Arguments

scan	scan number Allowed:integer Default: no default

Returns
a Glish record

Example

```
- d.qscan(9);
[rows=24, phases=4, if=2, field=0, ints=6, rownumbers=[29 30 31 32 33 34 35
36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52] ]
```

---

dish.reference - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Designate the reference spectrum

Synopsis
reference()

Description
This function can be used to designate a reference spectrum explicitly. For standard observing and calibration, d.TPcal or d.calib can be used instead. However, this function allows the option of calibrating a ref and sig scan that were not observed as part of the same observing procedure.

Returns
boolean (T if successful)

Example

```
- d.reference(10)
- d.signal(11)
- d.temp()
- d.show()
```

---

dish.restore - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Retrieve the contents of the uni Record from disk.

Synopsis
restore()

Description
This function will retrieve all of the contents of the uni record from disk file /.uni2params. This record includes vairables used by dish, such as the globalscan1 array, nregion settings, and so on. A complete list of the variables can be seen from dish by d.upr('all'). The record can be saved using the d.store() function.

Returns
boolean

Example

```
- d.nregion(1,100,400,512)
- d.nregion()
nregion currently set to:  [1,100][400,512]
To modify these values, use this function as follows:
nregion(1,256,850,1023)
- d.store()
- d.nregion(1,2,3,4)
- d.nregion()
nregion currently set to:  [1,2][3,4]
To modify these values, use this function as follows:
nregion(1,256,850,1023)
- d.restore()
- d.nregion()
nregion currently set to:  [1,100][400,512]
To modify these values, use this function as follows:
nregion(1,256,850,1023)
```

---

dish.restorestate - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Restore the state of DISH's GUIs.

Synopsis
restorestate()

Description
This function restored GUI paramters. Also see d.restore() for other state saving options.

Example

```
- d.restorestate()
```

Returns
Boolean

---

dish.rmadd - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Add an SDRecord to the results manager.

Synopsis
rmadd(scanrec, name, desc)

Description

Arguments

scanrec	an SDRecord Allowed:boolean Default: globalscan1
name	the name of the SDRecord in the Results Manager Allowed:string Default: auto-generated name
desc	the description of the SDRecord Allowed:string Default: none

Returns
T (if successful)

Example

```
- ok := d.rmadd(myspec)
```

---

dish.rms - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Display statistics on baseline regions.

Synopsis
rms()

Description
This function will calculate statistics on the regions set via the *setregion* command or *nregion* command for the spectrum stored in *globalscan1*.

Example

```
- d.rms()
Using 1023 for calculating baseline stats.
== Feed  1  ===
RMS   =  3.24550665    Mean =  0.000737983237    Num points =  1023
max   =  48.9361229   min =  -12.9025164
nregion =  [1 1023]
== Feed  2  ===
RMS   =  3.29074012    Mean =  0.351124813    Num points =  1023
max   =  50.4430733   min =  -14.0051394
nregion =  [1 1023]
T
```

Returns
Boolean

---

dish.saxis - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Set the units of the x axis in the plotter

Synopsis

saxis(strval)

Description

This function allows command line selection of available x-axis units on the plotter display.

Arguments

strval	units to be displayed Allowed:GHz, MHz, kHz, Hz, km/s, m/s, pix,index Default: none

Returns

boolean (T if successful)

Example

```
- d.saxis('MHz')
```

---

dish.save - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Save an SDRecord to file.

Synopsis

save(sdrec, outfile)

Description
This command, along with d.keep(), is the mechanism used to store results to disk. *save* allows the user to specify the SDrecord to be written, while *keep* simply writes the spectrum in *globalscan1*.

Arguments

sdrec	SDRecord to be saved Allowed:SDRecord Default: currently viewed
outfile	outfile (scan group) to be written to Allowed:string Default: fileout, as listed by d.files()

Returns
T (if successful)

Example

```
- d.open('mydata')
- d.fileout('newfile')
- sdrec := d.getc(10)
- d.save(sdrec);
```

---

dish.savestate - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Save the current state of DISH.

Synopsis
savestate()

Description

Saves the state of the GUIs and plotter to a file. The default file is in the home aips++/dishstate directory called default. This may be altered with the statefile function. The state of GUIs can be recovered with the d.restorestate() function.

Example

```
- d.savestate()
```

Returns
Boolean

---

dish.scale - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Multiply the globalscan1 spectrum by a value.

Synopsis
scale(factor)

Description

Arguments

factor	value to scale (multiply) by Allowed:float Default: no default

Returns
Boolean

Example

```
- d.plotc(10)
- d.scale(2.0)
- d.show()
```

---

dish.scanadd - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Add scan1 and scan2.

Synopsis
scanadd(scan1, scan2)

Description
Add the spectra from two scans.

Arguments

scan1	first scan Allowed:SDRecord Default: no default
scan2	second scan Allowed:SDRecord Default: no default

Returns
An SDRecord

Example

```
- on := d.getc(2)
- off := d.getc(3)
- addscan := d.scanadd(on,off);
```

---

dish.scanbias - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Adds a bias level to a scan.

Synopsis
scanbias(scan, offset)

Description
Adds a bias offset to a spectrum. Also see d.bias() to add a value to the globalscan1 record directly.
The bias value can be a single value, in which case the entire data array in the sdrecord is offset by one value. It can also be an array of values if the user wishes to apply different offsets to each polarization. The length of the offset array in this case must equal the number of polarizations in the sdrecord.

Arguments

scan	scan Allowed:SDRecord Default: no default
offset	bias level to add Allowed:float or array of floats Default: no default

Returns
An SDRecord

Example

```
- scan := d.getc(2)
- biasedscan := d.scanbias(scan,0.5);
- d.plotscan(biasedscan)
- biasedscan_b := d.scanbias(scan,[0.5,-0.5]);
- d.plotscan(biasedscan_b)
```

---

dish.scandiv - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Divide scan 1 by scan 2.

Synopsis
scandiv(scan1, scan2)

Description
Scan division.

Arguments

scan1	numerator Allowed:SDRecord Default: no default
scan2	denominator Allowed:SDRecord Default: no default

Returns
An SDRecord

Example

```
- on := d.getc(2)
- off := d.getc(3)
- divscan := d.scandiv(on,off)
```

---

dish.scanmult - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Multiply scan 1 by scan 2.

Synopsis
scanmult(scan1, scan2)

Description
Scan multiplication.

Arguments

scan1	numerator Allowed:SDRecord Default: no default	
scan2	denominator Allowed:SDRecord Default: no default	

Returns
An SDRecord

Example

```
- sc1 := d.getc(2)
- sc2 := d.getc(3)
- multscan := d.scanmult(sc1,sc2)
```

---

dish.scanscale - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Multiply an sdrecord's spectrum by a value or array

Synopsis
scanscale(scan1, sfactor)

Description

This function scales an sdrecord and returns the result as an sdrecord. Also see the function d.scale() to apply scaling to the globalscan1 record directly. The factor can be a single value, in which case the entire data array in the sdrecord is scaled by one value. The factor can also be an array of values if the user wishes to scale each polarization by a different value. The length of the sfactor array in this case must equal the number of polarizations in the sdrecord to be scaled.

Arguments

scan1	SDRecord to be scaled Allowed:SDRecord Default: no default
sfactor	multiplicative factor Allowed:float or array of floats Default: no default

Returns

Boolean

Example

```
- sc := d.getc(10)
- scaled := d.scanscale(sc, 2.0)
- d.plotscan(scaled)
- scaled_b := d.scanscale(sc, [2.0,4.5])
- d.plotscan(scaled_b)
```

---

dish.scansrr - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Performs a (signal-reference)/reference.

Synopsis

scansrr(scan1, scan2)

Description
This function does (S-R)/R calibration in an explicit way. Also see the d.calib() function for a more automated way to handle calibration.

Arguments

scan1	signal scan Allowed:SDRecord Default: no default
scan2	reference scan Allowed:SDRecord Default: no default

Returns
An SDRecord

Example

```
- on := d.getc(2)
- off := d.getc(3)
- Tant := d.scansrr(on,off)
- d.plotscan(Tant)
```

---

dish.scansub - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Subtract scan2 from scan1.

Synopsis
scansub(scan1, scan2)

Description
Substract one spectrum from another.

Arguments

scan1	scan to be subtracted from Allowed:SDRecord Default: no default
scan2	scan to subtract Allowed:SDRecord Default: no default

Returns
An SDRecord

Example

```
- on := d.getc(2)
- off := d.getc(3)
- subscan := d.scansub(on,off);
```

---

dish.sclear - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Clear the contents of the accumulator.

Synopsis
sclear()

Description
This function is used in averaging scans. It clears the contents of the accumulator, and should be used prior to using the accumulator for an averaging operation.

Returns
Boolean

Example

```
d.sclear()    # clear the accumulator
d.getc(35)    # retrieve calibrated scan 35
d.accum()     # add the scan to the accumulator
d.getc(36)    # retrieve calibrated scan 36
d.accum()     # add the scan to the accumulator
d.ave()       # average scans in the accumulator
d.show()      # display the result on the plotter
```

---

dish.select - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Select a subset of a scangroup.

Synopsis
select(scans, rows, source, restfrequency, date, ut)

Description
This will produce a new scan group in the Results Manager labeled working_groupx, where x is some integer value. Source names should be a single name (in single quotes) or a vector of source names (in double quotes or as a glish vector using brackets (['source A','source B',...]). A "*" is the source name wild card and will match any characters.

Arguments

scans	scan numbers Allowed:string - e.g., '[4,7,9]' Default: all
rows	row numbers in the MS Allowed:string - e.g., '[4:8],10' Default: all
source	source names Allowed:string Default: all
restfrequency	the restfrequency Allowed:vector of doubles Default: all
date	date of the observation Allowed:string Default: all
ut	seconds of UT time since the observations started Allowed:vector of floats Default: all

Returns
T (if successful)

Example

```
- ok := d.select(rows='[1:100]',scans='[4:6]');
```

---

dish.setregion - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Set the channel range for use in baseline subtraction

Synopsis
setregion()

Description
This function provides an interactive means of obtaining channel ranges for

baseline subtraction. The function changes the displayed units to channels (pixels) and then sets the regions through use of the mouse buttons. The regions are then displayed for viewing. These regions will be used by the *bshape*, *rms* and *baseline* routines.

Returns
boolean (T if successful)

Example

```
- d.setregion()
Use the left button to set location, right button to exit

region =  [21 156 309 644 803 1011]
```

---

dish.setYUnit - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Set the y-axis unit for a spectrum

Synopsis
setYUnit(unit)

Description
This function will set or override the y-axis unit for the globalscan1 spectrum. It does not affect the data values in the spectrum or calibrate in any way, it only changes the label on subsequent displays and the header entry for Y-units.

Arguments

unit	The units Allowed:string Default: no default

Returns
boolean (T if successful)

Example

```
- d.setYUnit('Ta*');
- d.show()
```

---

dish.show - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Display the globalscan1 SDRecord

Synopsis
show()

Description
This function will plot the spectrum currently contained in the *globalscan1* SDRecord.

Returns
boolean (T if successful)

Example

```
- d.show()
```

---

dish.show1 - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Display one polarization of the globalscan1 SDRecord

Synopsis
show1(ifeed)

Description
This function will plot the spectrum currently contained in the *globalscan1* SDRecord for the specified polarization/receptor. The other receptors will be zeroed out in the display (but the actual data are not affected). The principal display functions plotr and plotc also have selections on the polarization and can be used to display individual polarizations. In addition, *getpol* will return a single polarization from data to an SDRecord.

Arguments

ifeed	numeric designation for polarization Allowed:no default Default: integer

Returns
boolean (T if successful)

Example

```
- d.show1(1)
```

---

dish.showref - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Display the vref SDRecord

Synopsis
showref()

Description
This function will plot the spectrum currently contained in the *vref* SDRecord.

Returns
boolean (T if successful)

Example

```
- d.showref()
```

---

dish.signal - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Designate the signal spectrum

Synopsis
signal()

Description
This function can be used to designate a signal spectrum explicitly. For standard observing and calibration, d.TPcal or d.calib can be used instead. However, this function allows the option of calibrating a ref and sig scan that were not observed as part of the same observing procedure.

Returns
boolean (T if successful)

Example

```
- d.reference(10)
- d.signal(11)
- d.temp()
- d.show()
```

---

dish.smooth - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Smooth (hanning, boxcar or gaussian) an SDRecord.

Synopsis
smooth(scanrec, type, width, units, decimate)

Description
This function allows access to several smoothing functions.

Arguments

scanrec	an SDRecord Allowed:boolean Default: currently displayed
type	type of smoothing function to be applied Allowed:'HANNING', 'BOXCAR', or 'GAUSSIAN' Default: HANNING
width	width of the boxcar or gaussian Allowed:float Default: 3
units	units for the width Allowed:string Default: 'channels' or 'native'–only relevant for GAUSSIAN
decimate	decimate when smoothing (for HANNING) Allowed:boolean Default: F

Returns
An SDRecord

Description

Example

```
- mysmooth:=d.smooth(type='BOXCAR',width=5);
```

This does a 5 channel wide boxcar smooth.

---

dish.stat - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Generate statistics over a line region.

Synopsis
stat(scanrec, range, units)

Description

Arguments

scanrec	an SDRecord Allowed:boolean Default: currently displayed
range	the range over which the statistics will be generated Allowed:string Default: no default
units	the units used for the range Allowed:string Default: 'channels'

Returns
A record of the results (peak,area,min,rms,scan,centroid,vpeak)

Example

```
- mystat := d.stat{range='[50:100]');
```

---

dish.statefile - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Changes file for state saving.

Synopsis
statefile()

Description
Changes the location that the state of Dish's GUIs and plotter will be saved to. The full pathname is required.

Example

```
d.statefile('mydefaultstate');
d.savestate();                    # saves to mydefaultstate in the
                                       # current directory
```

Returns
Boolean

---

dish.stats - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Generate statistics over a region.

Synopsis
stats(quiet, feed, bchan, echan)

Description
This function provides basic statistics over a region. It is used mainly for obtaining information on line-free regions. The default behavior will write information on all polarizations to the command window.

Arguments

quiet	Suppress write of information to screen Allowed:boolean Default: F
feed	Limit information to a single polarization Allowed:integer Default: all
bchan	Beginning channel to calculate statistics Allowed:integer Default: 1
echan	End channel for statistics Allowed:integer Default: last

Returns
boolean (T if successful)

Example

```
- d.stats()
Feed : 1       bchan: 1       rms  : 3.382537      min  : -12.625508
Npts : 1023    echan: 1023    mean : 0.381799      max  : 51.412025

Feed : 2       bchan: 1       rms  : 3.264693      min  : -14.260262
Npts : 1023    echan: 1023    mean : -0.004182     max  : 49.291866
```

---

dish.store - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Store the contents of the uni Record to disk.

Synopsis
store()

Description
This function will store the uni record to disk file ~/.uni2params. This record includes vairables used by dish, such as the globalscan1 array, nregion settings,

and so on. A complete list of the variables can be seen from dish by d.upr('all'). The record can be restored using the d.restore() function.

Returns
boolean

Example

```
- d.nregion(1,100,400,512)
- d.nregion()
nregion currently set to:  [1,100][400,512]
To modify these values, use this function as follows:
nregion(1,256,850,1023)
- d.store()
- d.nregion(1,2,3,4)
- d.nregion()
nregion currently set to:  [1,2][3,4]
To modify these values, use this function as follows:
nregion(1,256,850,1023)
- d.restore()
- d.nregion()
nregion currently set to:  [1,100][400,512]
To modify these values, use this function as follows:
nregion(1,256,850,1023)
```

---

dish.summary - Function

Package	**dish**
Module	**dish**
Tool	**dish**

List a summary for a Measurement Set

Synopsis
summary()

Description
Summarizes a measurement set to the logger.

Returns
T (if successful)

Example

```
- d.summary()
```

---

dish.tellstack - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Return contents of the stack.

Synopsis
tellstack()

Description
This function is used to return the contents of the "stack". The stack is a construct (simply a vector of integers) which is used to organize a list of scans which will be operated on in some common way. Generally, it is used to assemble a list of scans which can then be averaged.

Returns
array of ints

Example

```
- d.addstack(10,50,2)
- d.delete(22)
- d.tellstack()
20  entries in the stack.
[10 12 14 16 18 20 24 26 28 30 32 34 36 38 40 42 44 46 48 50]
- avscan := d.aver(d.tellstack())
```

---

dish.upr - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Print the current value of variables in the uni-record.

Synopsis
upr(...)

Description
This function will print the current value of the specified variable in the uni-record. To display all variables, pass no arguments.

Arguments

...	list of variables Allowed:strings Default: all

Returns
boolean

Example

```
- d.upr('nfit','nregion')
nfit  =  0
nregion  =  [1,100000]

- d.upr()
tau  =  F
factor  =  1
astack  =  F
acount  =  0
edrop  =  0
bdrop  =  0
echan  =  0
bchan  =  0
emoment  =  0
```

```
bmoment  =  0
nfit  =  0
numaccum  =  0
vref  =  F
vsig  =  F
globalscan1  =  F
nregion  =  [1,100000]
nregionArr  =  [1 100000]
```

---

dish.utable - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Print channel and data columns to command window

Synopsis
utable()

Description
This function provides a shortcut to display the channel number and data from the current *globalscan1* SDRecord (typically the currently viewed data). It uses the bdrop and edrop variables to limit the range printed.

Example

```
d.utable()
- d.utable()
1 [0.042550344 -0.135733142]
2 [-0.0302197617 -0.149972931]
3 [0.0953259096 -0.0912254006]
4 [-0.0387661941 -0.0562298447]
5 [-0.0511925854 -0.0671290383]
6 [0.134882167 -0.204534769]
7 [-0.0294003487 -0.168623194]
...
```

Returns
Boolean

---

dish.writetofile - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Write X,Y axes to an ascii file on disk.

Synopsis
writetofile(outfile, scanrec)

Description
This will produce a file on disk with the user specified name containing the X and Y data for the specified spectrum (default=currently displayed) in 2+ columns (the first column is the currently displayed x-axis, the subsequent columns will be the data in each polarization of the data).

Arguments

outfile	the name of the file on disk where the data will be written Allowed:string Default: no default
scanrec	an SDRecord Allowed:boolean Default: currently displayed

Returns
T (if successful)

Example

```
ok := d.writetofile('junk.dat');
```

---

dish.zline - Function

Package	**dish**
Module	**dish**
Tool	**dish**

Draw a horizontal line at zero intensity on the plotter.

Synopsis
zline(torF)

Description
This function allows the option of plotting a line at the zero intensity as a reference. After using this function, subsequent plots will display it.

Arguments

torF	T indicates draw the line Allowed:boolean Default: no default

Returns
T (if successful)

Example

```
d.zline(T);
d.plotc(35); #see the zero line
```

1.1.2 dishd - Tool

Package dish
Module dish

dish tool (developer functions)

Constructors

Functions

addop	Add new operation to DISH tool
dologging	Switch to determine whether commands from the GUI are logged to the scripter
doselect	Toggle selection between last plotted and RM selection

history	Writes message to history of SDRecord.
logcommand	Writes command to scripter.
message	Writes a command to the DISH GUI message entry.
type	Returns the type of the DISH tool.

dishd.addop - Function

Package	**dish**
Module	**dish**
Tool	**dishd**

Add new operation to DISH tool

Synopsis
addop()

Description

Example

```
d.addop(include_file, operation_name);
```

Returns
Bool

---

dishd.dologging - Function

Package	**dish**
Module	**dish**
Tool	**dishd**

Switch to determine whether commands from the GUI are logged to the scripter

Synopsis
dologging()

Description

Example

```
d.dologging(T or F);
```

Returns
Bool

---

dishd.doselect - Function

Package	**dish**
Module	**dish**
Tool	**dishd**

Toggle selection between last plotted and RM selection

Synopsis
doselect()

Description
A toggle switch which determines whether commands operations will operate on the last viewed (the default behavior) or upon the selected item in the results manager. This gives the ability of performing bulk processing on SDIterators. Selecting will also change the GUI state of the Option 'select from rm (for SDITs)'.

Example

```
d.doselect(T or F);
```

Returns
Bool

---

dishd.history - Function

Package	**dish**
Module	**dish**
Tool	**dishd**

Writes message to history of SDRecord.

Synopsis
history()

Description
Writes a message to the history line of the current SDRecord. It takes two arguments: 1) a string of the command executed , 2) a record of the inputs to that command.

Example

```
- d.history('dish.ops().smooth.settype',[type='HANNING']);
```

Returns
Bool

---

dishd.logcommand - Function

Package	**dish**
Module	**dish**
Tool	**dishd**

Writes command to scripter.

Synopsis
logcommand()

Description
Writes a command to the scripter. logcommand uses the same syntax as history but the information is logged to the scripted rather than to the record's history.

Example

Returns
Bool

---

dishd.message - Function

Package	**dish**
Module	**dish**
Tool	**dishd**

Writes a command to the DISH GUI message entry.

Synopsis
message()

Description

Example

Returns
Bool

---

dishd.type - Function

Package	**dish**
Module	**dish**
Tool	**dishd**

Returns the type of the DISH tool.

Synopsis
type()

Description
A mandatory function required by the toolmanager. The type of any dish tool is 'dish'.

Example

Returns
Bool

1.1.3 msr - Tool
Package dish
Module dish

dish spectral line measure tool

Constructors

Functions

areape	Derives profile characteristic values.
areapf	Derives profile characteristic values.
mean	Derives characteristic values based on the mean.
peak	Derives characteristic values based on the peak.
twopeak	Derives characteristic values based on the peaks.

Description
The **dish** msr tool duplicates the functionality of the GMEASURE methods in Unipops (modes 1-5). Each method derives three characteristic values for the flux over the range of channels:

- flux integral (i.e. flux*dv)
- characteristic velocity (i.e. estimate of systemic velocity)

- velocity width

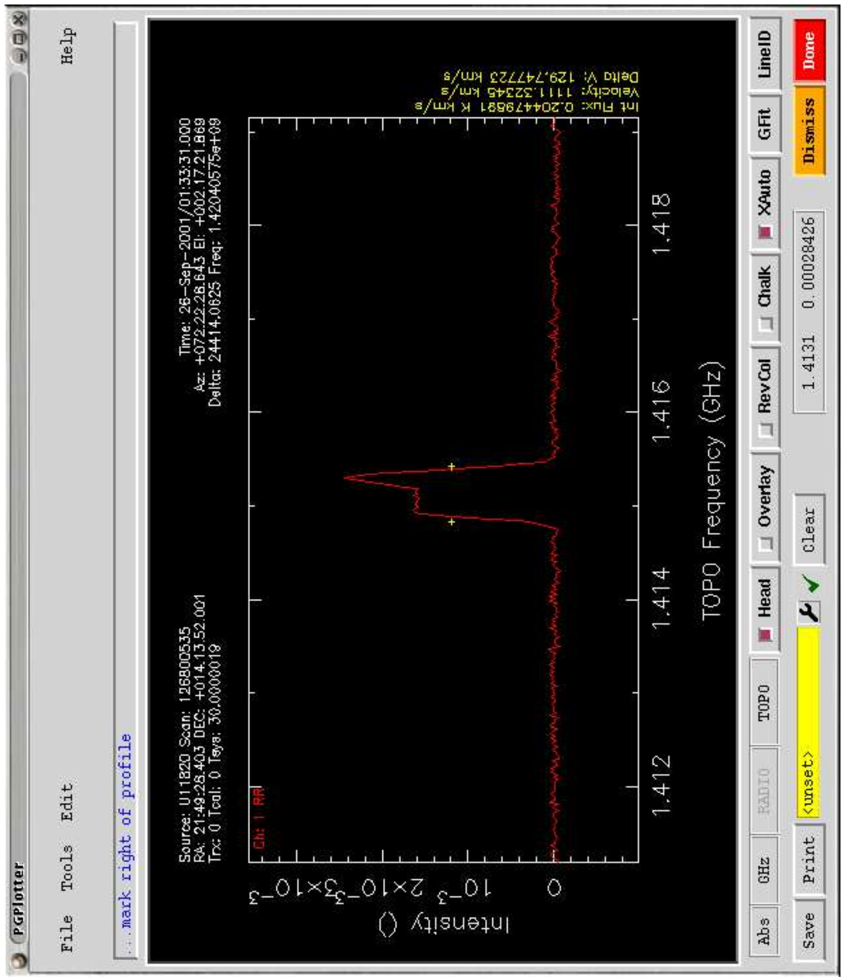

Figure 1.1: View of plotter after msr mean fit

msr.areape - Function

Package	**dish**
Module	**dish**
Tool	**msr**

Derives profile characteristic values.

Synopsis
areape(lv, first, last, levs, rms, usecurs, norder)

Description
This function calculates the width and average velocity of a galaxy spectrum. The two halves of the double horned profile are treated separately. For each half, the peak is determined (it is assumed that the two peaks are on either side of the center of the user specified limits first and LAST). Then on each side of the profile a polynomial (by default 1st order) is fit to a user specified portion of the profile. If the fit is poor, (i.e. (chisqr - deg.of.freedom) ¿ 3 times sqrt(2*deg.of.freedom)), the user is given the option of fitting a 2nd degree polynomial and/or changing the levels between which to fit. The FR points (where FR is a user specified fraction of the peak flux) on either side of the profile are located by interpolation between the two data points that bracket the FR point. The interpolaton is done by channel number and then the velocity corresponding to the interpolated channel number is computed. The average velocity is taken to be the mean of these two velocities and the width is taken to be the difference between these two velocities. The uncertainity in the velocity measurment is then calculated from the rms in the baseline and the coefficients of the fitted polynomial. Finally the area between the channels first and last is computed as in the other methods.

Example

```
- d.msr().areape(rms=0.005,levs=[0.8,0.2,0.4],usecurs=T)
Upper and lower profile points of left peak coincide!
Poor fit to profile: the normalized chisqr are nan14.2039694
Number of points used to fit left and right halves are 23
[type=areape, lchans=[329 328 329] , rchans=[351 353 351] , badfit=T,
vwidth=116.518171, vel=1111.61462, vrms=0.109011172, badpeak=F,
limits=[320.401001 358.420601] , fluxint=98.4090931]
```

Arguments

lv	an SDRecord Allowed:SDRecord - Not enabled Default: F
first	first channel number in range to integrate Allowed:integer Default: F
last	last channel number in range to integrate Allowed:integer Default: F
levs	define levels relative to each peak flux Allowed:vector Default: [0.8,0.2,0.4]
rms	rms of the spectrum (non-line) Allowed:float Default: 0.0
usecurs	optionally set ranges with cursor Allowed:boolean Default: F
norder	order of the fit Allowed:integer Default: 1

Returns
Record (fluxint, vel, vwidth, level, limits, type, mean)

---

msr.areapf - Function

Package	**dish**
Module	**dish**
Tool	**msr**

Derives profile characteristic values.

Synopsis
areapf(lv, first, last, levs, rms, usecurs, norder)

Description
This subroutine calculates the width and average velocity of a galaxy spectrum. The two halves of the double horned profile are treated separately. For each half, the peak is determined (it is assumed that the two peaks are on either side of the center of the user specified limits first and LAST). Then each

on each side of the profile a polynomial (by default 1st order) is fit to a user specified portion of the profile. If the fit is poor, (i.e. (chisqr - deg.of.freedom) ¿ 3 times sqrt(2*deg.of.freedom)), the user is given the option of fitting a 2nd degree polynomial and/or changing the levels between which to fit. Finaly using the fitted polynomial, the fr points (where fr is a user specified fraction of the peak flux) on either side of the profile is located. The interpolaton is done by channel number and then the velocity corresponding to the interpolated channel number is computed. The average velocity is taken to be the mean of these two velocities and the width is taken to be the difference bewteen these two velocities. The uncertainity in the velocity measurment is then calculated from the covariance matrix of the coefficients of the fitted polynomial. Finally the area of the profile between the limits first and last is computed as summation (x(i)*dv(i)), where dv(i) is the velocity difference between channels i and i-1.

Example

```
- d.msr().areapf(rms=0.005,levs=[0.8,0.2,0.4],usecurs=T)
Upper and lower profile points of left peak coincide!
Poor fit to profile: the normalized chisqr are nan 14.2039694
Number of points used to fit left and right halves are 23
[type=areapf, lchans=[323.402033 322.402033 322] , rchans=[351 353 351] ,
badfit=T, vwidth=150.814539, vel=1094.01721, vrms=0.184532409, badpeak=F,
limits=[322.402033 356.419569] , fluxint=99.7206675]
```

Arguments

lv	an SDRecord Allowed:SDRecord - Not enabled Default: F
first	first channel number in range to integrate Allowed:integer Default: F
last	last channel number in range to integrate Allowed:integer Default: F
levs	define levels relative to each peak flux Allowed:vector Default: [0.8,0.2,0.4]
rms	rms of the spectrum (non-line) Allowed:float Default: 0.0
usecurs	optionally set ranges with cursor Allowed:boolean Default: F
norder	order of the fit Allowed:integer Default: 1

Returns
Record (fluxint, vel, vwidth, level, limits, type, mean)

---

msr.mean - Function

Package	**dish**
Module	**dish**
Tool	**msr**

Derives characteristic values based on the mean.

Synopsis
mean(lv, first, last, flevel, usecurs)

Description
The flevel (flux level) is multiplied by the mean over the channel range. Starting at the edges of the range, channels are found where the flux first rises above the derived level. The integrated flux is taken between these channels. A linear interpolation is done to find the velocity corresponding to where the

flux equals that level. These two velocities (one at each end) are used to determine the characteristic velocity (the average of the two). The velocity width is the absolute value of their difference.

Example

```
- d.msr().mean(first=324,last=360,flevel=0.2)
[fluxint=99.0127912, vel=1109.67758, vwidth=130.143466, level=0.103916899,
limits=[328 352] , type=mean, mean=0.519584496]
- d.msr().mean(flevel=0.3,usecurs=T)
[fluxint=0.204479891, vel=1111.32345, vwidth=129.747723, level=0.000357433806,
limits=[329.167913 353.229436] , type=mean, mean=0.00119144602]
```

Arguments

lv	an SDRecord Allowed:SDRecord - Not enabled Default: F
first	first channel number in range to integrate Allowed:integer Default: F
last	last channel number in range to integrate Allowed:integer Default: F
flevel	fractional flux level Allowed:float Default: 0.3
usecurs	optionally set ranges with cursor Allowed:boolean Default: F

Returns
Record (fluxint, vel, vwidth, level, limits, type, mean)

---

msr.peak - Function

Package	**dish**
Module	**dish**
Tool	**msr**

Derives characteristic values based on the peak.

Synopsis
peak(lv, first, last, flevel, rms, usecurs)

Description
Identical to mean except that the cutoff level is the peak value times flevel. Also, you can optionally supply a noise estimation (the rms argument). This is subtracted from the peak value before it is multiplied by flevel. The returned fields here are nearly identical to those in mean. The only difference is that there is a peak field here instead of mean. That is the actual peak, not the value reduced by rms.

Example

```
- d.msr().peak(flevel=0.5,usecurs=T)
[fluxint=91.9571165, vel=1108.78946, vwidth=105.650325, level=0.610705972,
limits=[330.404096 350.418538] , type=peak, peak=1.22141194]
```

Arguments

lv	an SDRecord Allowed:SDRecord - Not enabled Default: F
first	first channel number in range to integrate Allowed:integer Default: F
last	last channel number in range to integrate Allowed:integer Default: F
flevel	fractional flux level Allowed:float Default: 0.3
rms	rms of spectrum Allowed:float Default: 0.
usecurs	optionally set ranges with cursor Allowed:boolean Default: F

Returns
Record (fluxint, vel, vwidth, level, limits, type, peak)

---

msr.twopeak - Function

Package	**dish**
Module	**dish**
Tool	**msr**

Derives characteristic values based on the peaks.

Synopsis
twopeak(lv, first, last, peak1, peak2, flevel, rms, usecurs)

Description
Similar to peak except that there are two cutoff levels determined for the data. Peaks are found within 10 channels from each guessed peak location (peak1 and peak2), the cuttoff levels are then set to be flevel times each peak. The value from the peak near first is used to set the lower channel limit and the value from the peak near last is used to set the upper channel limit in the same way that peak and mean set those channel limits. As with peak, you optionally supply a noise estimate (the rms argument) which is subtracted from the peak values before they are multiplied by flevel. The return value fields are nearly the same as in peak. Here, the level and peak fields have 2 elements. There is also a peakchan field which contains the refined locations of the two peaks.

Example

```
- d.msr().twopeak(flevel=0.5,usecurs=T)
[fluxint=94.6122732, vel=1110.71511, vwidth=109.501611, level=[0.610705972
0.489752263] , limits=[330 351] , type=twopeak, peak=[1.22141194 0.979504526] ,
peakchan=[334 336] ]
```

Arguments

lv	an SDRecord Allowed:SDRecord - Not enabled Default: F
first	first channel number in range to integrate Allowed:integer Default: F
last	last channel number in range to integrate Allowed:integer Default: F
peak1	guess of first peak channel number Allowed:integer Default: F
peak2	guess of second peak channel number Allowed:integer Default: F
flevel	fractional flux level Allowed:float Default: 0.3
rms	rms of spectrum Allowed:float Default: 0.
usecurs	optionally set ranges with cursor Allowed:boolean Default: F

Returns
Record (fluxint, vel, vwidth, level, limits, type, peak, peakchan)

1.1.4 plotter - Tool

Package dish
Module dish

dish plotter tool

Constructors

Functions

ips	Image profile support.
plotrec	Plot an SDRecord.

putranges	This allows setting of range regions for use by future operations.
ranges	Retrieves any active ranges.

Description
The **dish** plotter is a superposition of numerous plotting capabilities with additional functionality added. In particular, the plotter has all of the functions of the **pgplotter** tool (as described in the GLISH Manual section on Glish/PGPLOT) and the AIPS++ **pgplotwidget** tool. We dont' describe this functionality as it is detailed in the above sections.

```
displaylist pgplotwidget
setcallback pgplotwidget
deactivatecallback pgplotwidget
activatecallback pgplotwidget
record pgplotwidget
canplay pgplotwidget
play pgplotwidget
postscript pgplotwidget
plotfile pgplotwidget
refresh pgplotwidget
restore pgplotwidget
done pgplotwidget
plotxy pgplotwidget
maskline pgplotwidget
clear pgplotwidget
message pgplotwidget
settings pgplotwidget
cursor pgplotwidget
lastchange pgplotwidget
curs pgplotwidget
size pgplotwidget
addredrawfunction pgplotwidget

arro - wnad pgplotter

plotxy1 pgplotter
ploty pgplotter
ploty2 pgplotter
plotxy2 pgplotter
timey pgplotter
timey2 pgplotter
setxscale pgplotter
setx2scale pgplotter
setyscale pgplotter
```

```
sety2scale pgplotter
setxaxisgrid pgplotter
setyaxisgrid pgplotter
sety2axisgrid pgplotter
setcolor pgplotter
setplottitle pgplotter
setxaxislabel pgplotter
setyaxislabel pgplotter
psprint pgplotter
psprinttofile pgplotter
unzoom pgplotter
addtool pgplotter
tool pgplotter
typ pgplotter
editgui pgplotter
userframe pgplotter
demo pgplotter
gui                     pgplotter
screen                  pgplotter
busy                    pgplotter
shortnametofullname     pgplotter
resetplotnumber         pgplotter

is_active dishd dishd indicates developer only function
pf dishd
index dishd
header dishd
overlay dishd
reverse dishd
chalkboard dishd
gfit dishd
lineid dishd
setranges dishd
compute_statistics dishd
active dishd
register_range dishd
assert_range_cursor dishd
deassert_range_cursor dishd
create dishd
lineid dishd
debug dishd

ips dish
plotrec dish
putranges dish
ranges dish
```

```
uzoom dish
```

In the following, we describe the additional functionality available within the **dish** plotter.

plotter.ips - Function

Package	**dish**
Module	**dish**
Tool	**plotter**

Image profile support.

Synopsis
ips()

Description
ips or Image Profile Support is actually a tool within the **images** Module. All data in **dish** are plotted as an image (1-d, 2-d, or 3-d). Numerous function exist within the **ips** tool for getting data and plotting data (e.g. getordinate, getabcissaunit, etc.).

Example
The following will change the displayed units on the plotter to 'Hz'.

```
- d.plotter.ips.setabcissaunit('Hz')
T
```

Returns
Bool

---

plotter.plotrec - Function

Package	**dish**
Module	**dish**
Tool	**plotter**

Plot an SDRecord.

Synopsis

plotrec()

Description
This is the basic function within the plotter, allowing an SDRecord to be plotted from the command line. A shortcut to this command is available as d.plotscan. The plotrec command has an optional 'overlay' argument which when 'T', will overlay the record on the existing plot.

Example

```
- d.plotter.plotrec(myrec);
```

Returns
Bool

---

plotter.putranges - Function

Package	**dish**
Module	**dish**
Tool	**plotter**

This allows setting of range regions for use by future operations.

Synopsis
putranges()

Description

Example

Returns
Bool

---

plotter.ranges - Function

Package	**dish**
Module	**dish**
Tool	**plotter**

Retrieves any active ranges.

Synopsis
ranges()

Description
Any ranges that have been set by the baseline or statistics operations can be obtained at the command line through this command.

Example

```
- d.plotter.ranges()
[1.40517056e+09:1.41934387e+09]
```

Returns
Vector

1.1.5 rm - Tool

Package	**dish**
Module	**dish**

dish results manager

Description
The "Results Manager" is where all of the variables created by **dish** are shown. These are all global symbols which you can manipulate directly at the glish prompt.
In the following examples, we assume that there is an existing **dish** tool called 'd'.

Constructors

Functions

add	Add an SDRecord to the Results Manager.
copy	Copies currently viewed to the clipboard.
delete	Deletes the Results Manager Variable with the specified index.
done	Deletes and closes down the results manager tool (not recommended!).
getdescriptions	Gets descriptions from an index value in the results manager.
getlastviewed	Returns the SDRecord of the last viewed item in the plotter.
getnames	Gets name from an index value in the results manager.
getselectiondescriptions	Gets the descriptions of all selected items in the Results Manager.
getselectionnames	Gets the names of all selected items in the Results Manager.
getselectionvalues	Gets the values of all selected items in the Results Manager.
getvalues	Returns values of selected in results manager.
gui	Opens the GUI for the results manager if it isn't already viewed.
nogui	Specifies whether the GUI for the Results Manager should be viewed.
paste	Pastes from the clipboard into the results manager.
rename	Renames a selected Results Manager variable.
select	Selects an item in the results manager.
selectbyname	Selects an item in the results manager by name rather than by index number.
selectionsize	Gives the number of items selected in the results manager.
setdescription	Sets description to a specified string.
setlastviewed	For a specified index value in the results manager, it will set this as the last viewed item.
size	Gives the number of items in the results manager.
wsname	Retrieves name of MS that will be used for any selections.

rm.add - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Add an SDRecord to the Results Manager.

Synopsis
add()

Description

Example

```
- d.rm().add(name,description,value,type,decorate)
```

```
For example, using myrec:
- d.rm().add('myrec','something I did', myrec, type='SDRECORD');
```

A shortcut to this function is provided through:

```
- d.rmadd(myrec,'myrec','something else I did');
```

Returns
Bool

---

rm.copy - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Copies currently viewed to the clipboard.

Synopsis
copy()

Description
The clipboard is a virtual workspace for storing any information. This command writes the currently viewed SDRecord to the clipboard. It can later be *paste*d into the results manager.

Example

```
- d.rm().copy()
T
- d.rm().paste() # the SDRecord should appear in the Results Manager
```

Returns
Bool

---

rm.delete - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Deletes the Results Manager Variable with the specified index.

Synopsis
delete()

Description

Example

```
- d.rm().delete(2)
T
```

Returns
Bool

---

rm.done - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Deletes and closes down the results manager tool (not recommended!).

Synopsis
done()

Description
This is a developer function. This should not be used by astronomers. The keys to closing elements of **dish** are contained at a higher level, in the user functions.

Example

Returns
Bool

---

rm.getdescriptions - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Gets descriptions from an index value in the results manager.

Synopsis
getdescriptions()

Description

Example

```
- d.rm().getdescriptions(1)
SD dataset from file
```

Returns
String

---

rm.getlastviewed - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Returns the SDRecord of the last viewed item in the plotter.

Synopsis
getlastviewed()

Description

Example

```
- lastviewed := d.rm().getlastviewed();
```

Returns
SDRecord

---

rm.getnames - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Gets name from an index value in the results manager.

Synopsis
getnames()

Description

Example

```
- d.rm().getnames(1)
dishdemo2
```

Returns
String

---

rm.getselectiondescriptions - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Gets the descriptions of all selected items in the Results Manager.

Synopsis
getselectiondescriptions()

Description

Example

```
- mydesc:=d.rm().getselectiondescriptions()
- mydesc[1]
SD dataset from file
- mydesc[2]
something I did
```

Returns
Vector of Strings

---

rm.getselectionnames - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Gets the names of all selected items in the Results Manager.

Synopsis
getselectionnames()

Description

Example

```
- d.rm().getselectionnames()
naic_ms1 mydum1
```

Returns
Vector of Strings

---

rm.getselectionvalues - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Gets the values of all selected items in the Results Manager.

Synopsis
getselectionvalues()

Description

Example

Returns
Vector of SDRecords

---

rm.getvalues - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Returns values of selected in results manager.

Synopsis
getvalues()

Description

Example

```
- myaverscan := d.rm().getvalues(2)
- field_names(myaverscan)
data header hist other
- #the variable myaverscan is an SDRecord!
```

Returns
SDRecord

---

rm.gui - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Opens the GUI for the results manager if it isn't already viewed.

Synopsis
gui()

Description
This is a developer function. This should not be used by astronomers. The keys to mapping and unmapping GUI elements of **dish** are contained at a higher level, in the user functions.

Example

Returns
Bool

---

rm.nogui - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Specifies whether the GUI for the Results Manager should be viewed.

Synopsis
nogui()

Description
This is a developer function. This should not be used by astronomers. The keys to mapping and unmapping GUI elements of **dish** are contained at a higher level, in the user functions.

Example

Returns
Bool

---

rm.paste - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Pastes from the clipboard into the results manager.

Synopsis
paste()

Description

Example

Returns
Bool

rm.rename - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Renames a selected Results Manager variable.

Synopsis
rename()

Description
This provides a command line function to rename a selected results manager variable.

Example

```
- d.rm().rename('orion_scan1')
T
```

Returns
Bool

---

rm.select - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Selects an item in the results manager.

Synopsis
select()

Description
For a specified index value, it will select (and consequently display) the item in

the Results Manager.

Example

```
- d.rm().select(2)
F
```

Returns
Bool

---

rm.selectbyname - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Selects an item in the results manager by name rather than by index number.

Synopsis
selectbyname()

Description

Example

```
- d.rm().selectbyname('naicspec1');
F
- # the spectrum called 'naicspec1' in the Results Manager is displayed.
```

Returns
Bool

---

rm.selectionsize - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Gives the number of items selected in the results manager.

Synopsis
selectionsize()

Description

Example

```
- d.rm().selectionsize()
1
```

Returns
Integer

---

rm.setdescription - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Sets description to a specified string.

Synopsis
setdescription()

Description

Example

```
- d.rm().setdescription(1,'my description is now different');
F
- d.rm().getdescriptions(1)
my description is now different
```

Returns
Bool

---

rm.setlastviewed - Function

Package	**dish**
Module	**dish**
Tool	**rm**

For a specified index value in the results manager, it will set this as the last viewed item.

Synopsis
setlastviewed()

Description

Example

Returns
Bool

---

rm.size - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Gives the number of items in the results manager.

Synopsis
size()

Description

Example

```
- d.rm().size();
32
```

Returns
Integer

---

rm.wsname - Function

Package	**dish**
Module	**dish**
Tool	**rm**

Retrieves name of MS that will be used for any selections.

Synopsis
wsname()

Description
This is a developer function. This should not be generally used.

Example

Returns
Bool

1.1.6 dishdemodata - Function

Package	**dish**
Module	**dish**

Set up demonstration data for Dish

Synopsis
dishdemodata()

Description
This function converts the sample SDFITS data files (dishdemo1.fits, dishdemo2.fits, dishmopra.fits, dishparkes.fits, and dishspecproc.fits) into AIPS++tables using tablefromfits. If these tables already exist in your current working directory, then this command does nothing. Status messages from this conversion process are sent to the AIPS++logger.
Use the dishdemodata function to get a copy of the demonstration data, as in this example:

Example

```
- include 'dishdemodata.g'
T
- dishdemodata()
T
```

The following text appears in the AIPS++ logger:

```
dishdemodata starts
Creating demo table : dishdemo1
FITS file found : /aips++/data/demo/dishdemo/dishdemo1.fits
Begin converting it to an aips++ table ...
Demo table created : dishdemo1
Creating demo table : dishdemo2
FITS file found : /aips++/data/demo/dishdemo/dishdemo2.fits
Begin converting it to an aips++ table ...
Demo table created : dishdemo2
Creating demo table : dishmopra
FITS file found : /aips++/data/demo/dishdemo/dishmopra.fits
Begin converting it to an aips++ table ...
Demo table created : dishmopra
Creating demo table : dishparkes
FITS file found : /aips++//data/demo/dishdemo/dishparkes.fits
Begin converting it to an aips++ table ...
Demo table created : dishparkes
Creating demo table : dishspecproc
FITS file found : /aips++/data/demo/dishdemo/dishspecproc.fits
Begin converting it to an aips++ table ...
```

```
Demo table created : dishspecproc
The single dish demo tables are: dishdemo1 dishdemo2 dishmopra dishparkes dishspecproc
```

Other SDFITS files (and binary tables) can be converted using tablefromfits.

```
tab := tablefromfits('mydemo','/aips++/data/dishdemo/dishparkes.fits')
```

A new table named 'mydemo' is created and filled using the input fits file (in this case, dishparkes.fits). This table is then opened and the returned value is a table tool that can be used immediately. Once the table has been converted, there is no need to run tablefromfits on that same fits file again. You can use the normal table function to re-open that table as necessary.

```
tab := table('mydemo')
```

1.1.7 sdaverager - Tool

Package **dish**
Module **dish**

A tool for averaging single dish data
include "sdaverager.g"

Constructors

sdaverager	An sdaverager

Functions

accumiterator	Add an entire iterator to the internal accumulation
accumulate	Add the data array to the internal accumulation.
average	Get the current internal accumulation
clear	Clear the on-going average.
dorestshift	Do we align by rest frequency?
getalignment	Return the current alignment option.
getweighting	Return the current weighting option.
restshiftstate	Are we aligning by rest frequency?
setalignment	Set the alignment option.
setweighting	Set the weighting option.

Description
This `tool` is intended to be used with **dish**, the single dish environment. It

averages single dish data including the contents of an Sditerator `tool` with a single function call.

Example

```
sdit := sditerator('dishdemo1')
avger := sdaverager()
avger.clear()
avger.setalignment('VELOCITY')
avger.setweighting('TSYS')
avger.dorestshift(T)
ok := avger.accumiterator(sdit)
ok := avger.average(newarr, newtsys, newcrpix, newcrval,
                    newcdelt, newcunit, newctype, newveldef,
                    newrestfreq, newexposure, newduration)
```

The first line makes an sditerator from a file. The second line makes the averager tool. The averager is then cleared before the accumulation starts. Various options are set. The entire sditerator is then averaged. The resulting averaged array, system temperature, x-axis description, rest frequency, exposure, and duration are then requested from the averager. At this point, additional sdrecords from other iterators or other sources could continue to be added to this on-going average. If a new average is to be started, the averager must first be cleared.

See Also
dish reference manual Using Dish sditerator tool

sdaverager - Constructor

Package	**dish**
Module	**dish**
Tool	**sdaverager**

An sdaverager

Synopsis
sdaverager()

Returns
object

sdaverager.clear - Function

Package	**dish**
Module	**dish**
Tool	**sdaverager**

Clear the on-going average.

Synopsis
clear()

Returns
boolean

sdaverager.setweighting - Function

Package	**dish**
Module	**dish**
Tool	**sdaverager**

Set the weighting option.

Synopsis
setweighting(option)

Arguments

option	The weighting option Allowed:'NONE','RMS','SIGMA'

Returns
boolean

Description
A weighting of `NONE` means that each data value has equal weight. The `RMS` weighting means that each data value in a spectra is weighted by the RMS

across the entire spectra. In the future it will be possible to specify a region over which this RMS is calculated. The `SIGMA` weighting option means that each data value in a spectra is weighted by $t_{exposure}/T_{sys}^2$.

---

sdaverager.getweighting - Function

Package	**dish**
Module	**dish**
Tool	**sdaverager**

Return the current weighting option.

Synopsis
getweighting()

Returns
string

---

sdaverager.setalignment - Function

Package	**dish**
Module	**dish**
Tool	**sdaverager**

Set the alignment option.

Synopsis
setalignment(option)

Arguments

option	The alignment option Allowed:'NONE','VELOCITY'

Returns
boolean

Description

This function sets how the spectra are aligned during the averaging. If this is set to `NONE` there is no alignment. Channels are added in the order they occur starting with the first channel and proceding until there are no more channels. If the alignment options is `VELOCITY` then averaging is done after first aligning the data in by velocity. The first scan in the average determines the velocity definition and reference frame to use throughout the average. Subsequent scans first have their x-axes converted to velocity using that definition and reference frame. If the fractional channel alignment is > 0.01 channels then a fourier transform is used to do that portion of the alignment. Velocity alignment always happens after any rest frequency adjustment indicated by the restshiftstate.

---

sdaverager.getalignment - Function

Package	**dish**
Module	**dish**
Tool	**sdaverager**

Return the current alignment option.

Synopsis
getalignment()

Returns
string

---

sdaverager.dorestshift - Function

Package	**dish**
Module	**dish**
Tool	**sdaverager**

Do we align by rest frequency?

Synopsis
dorestshift(torf)

Arguments

torf	Do we align by rest frequency? Allowed:boolean

Returns
boolean

Description
This option is useful if more than one specification of the rest frequency of the same line is present in the data. If this option is set to T, the x-axis is adjusted before alignment so that it is as if the data were observed at the rest frequency of the first sdrecord in the current accumulation. This option is only active when `VELOCITY` alignment is in effect.

---

sdaverager.restshiftstate - Function

Package	**dish**
Module	**dish**
Tool	**sdaverager**

Are we aligning by rest frequency?

Synopsis
restshiftstate()

Returns
boolean

---

sdaverager.accumulate - Function

Package	**dish**
Module	**dish**
Tool	**sdaverager**

Add the data array to the internal accumulation.

Synopsis
accumulate(data, tsys, crpix, crval, cdelt, cunit, ctype, veldef, restfrequency, exposure, duration)

Arguments

data	The data array Allowed:Float matrix
tsys	The system temp Allowed:Float
crpix	X axis reference pixel Allowed:Double
crval	X axis value at crpix Allowed:Double
cdelt	X axis increment per pixel Allowed:Double
cunit	X axis units Allowed:string
ctype	Type of X axis Allowed:string
veldef	Velocity definition Allowed:string
restfrequency	The rest frequency Allowed:Double
exposure	The effective integration time Allowed:Float
duration	The total integration time Allowed:Float

Returns
boolean

---

sdaverager.accumiterator - Function

Package	**dish**
Module	**dish**
Tool	**sdaverager**

Add an entire iterator to the internal accumulation

Synopsis
accumiterator(sditerator)

Arguments

sditerator	The sditerator to averager Allowed:sditerator

Returns
boolean

---

sdaverager.average - Function

Package	**dish**
Module	**dish**
Tool	**sdaverager**

Get the current internal accumulation

Synopsis
average(data, tsys, crpix, crval, cdelt, cunit, ctype, veldef, restfrequency, exposure, duration)

Arguments

data	The average data array
tsys	The average system temperature
crpix	X axis reference pixel of the average
crval	X axis value at the reference pixel
cdelt	X axis increment per pixel
cunit	X axis units
ctype	X axis type
veldef	Velocity definition
restfrequency	The rest frequency
exposure	The effective integration time
duration	The total integration time

Returns
boolean

1.1.8 sditerator - Tool

Package	**dish**
Module	**dish**

A `tool` for interacting with single dish data.
include "sditerator.g"

Constructors

newsditerator	Make a new, empty sditerator.
sditerator	An sditerator from an existing file on disk.
sditfromsdit	An sditerator from another sditerator.

Functions

appendHistory	Add a history element to the global sditerator history.
appendrec	Put an sdrecord at the end of the sditerator.
correcteddata	Is the CORRECTED_DATA column being used?.
deepcopy	Make a true copy of this data set.
deleterec	Delete the sdrecord at the current location pointer.
done	Close this sditerator and disconnect it from the data set.
flush	Flush the contents to disk.
get	Get an sdrecord.
getdata	Get the data part of the sdrecord.
getdesc	Get the desc part of the field of the sdrecord.
getdescvector	Get a vector of values from the desc record.
getempty	Get an empty sdrecord.
getheader	Get the header part of the sdrecord.
getheadervector	Get a vector of values from the header record.
getother	Get the other part of the sdrecord.
getvectors	Get a vector of values for the fields in the template.
history	Get the global history associated with all sdrecords
id	Return the ID record of this sditerator.
iswritable	Can sdrecords be stored (written) to this sditerator?
length	Get the total number of sdrecords in this sditerator.
location	Get the value of the location pointer.
lock	Set the appropriate lock on all associated tables.
more	Are there more sdrecords after this location?
name	What is the name of the underlying data on disk.
next	Set the location pointer to the next record.
origin	Set the location pointer to the first record.
previous	Set the location pointer to the previous record.
put	Put an sdrecord at the current location pointer.
reselect	Re-apply the previous selection to the parent data set.
resync	Resynchronize with the data on disk.
sel2string	Convert a selection record to a string.
select	Make a new sditerator by applying a selection criteria record
setlocation	Set the location pointer to the indicated record.
stringfields	Returns a record indicating which fields in the sditerator are strings.
type	Return the type of the underlying data.

unlock	Release the lock on all associated tables.
usecorrecteddata	Use the CORRECTED_DATA column?.

Description

This `tool` is intended to be used with the dish `tool` . It provides access to single dish data sets. One can step (iterate) through a data set with this `tool` . Two types of single dish data sets are supported: MeasurementSet and a flat table modeled on the SDFITS convention. Data is copied to and from an SDRecord, which is a *Glish* record structure with a set of fixed fields plus a few optional fields which vary depending on the data source. Data can only be saved back to an sditerator using a flat table as the underlying data source. Selection is supported either through a constructor or through the select function.

The SDRecord structure

The SDRecord is the fundamental data atom of this `tool` as well as of dish. An SDRecord is a *Glish* record having the following fields:

data A *Glish* record containing the data and things which describe that data (axes, units, flags, weights, etc.). The following fields are found in **data**:

- **arr** The data array. Its dimensions are (nstokes, nchan) where nchan is the number of frequency channels and nstokes is the number of stokes pixels (polarizations). The type may be real or complex.
- **flag** A boolean array with the same shape as **arr**. A value of `T` (true) for any pixel here means the corresponding pixel in **arr** has been flagged as bad.
- **sigma** An array of floats having the same shape as **arr**.
- **weight** An array of floats having the same shape as **arr**.
- **desc** A *Glish* record describing **arr** (axes and units). The following fields are found in **desc**:
  - **chan_freq** A vector of quantities giving each frequency value along the first axis of **arr**. See the quanta `tool` for a description of how to extract the values from a quantity as well as how to work with quantities.
  - **refframe** A string indicate the frequency measures reference frame for the **chan_freq** values.
  - **reffrequency** A reference frequency for the frequency axis. This has the same units as the bf chan_freq values do.
  - **chan_width** The channel width at **reffrequency** for use in selection. This has the same units as the bf chan_freq values do.
  - **restfrequency** A rest frequency associated with this data. This has the same units as the bf chan_freq values do.

corr_type A vector of strings giving the polarization type for each pixel of the stokes axis in **arr**.

units The units of **arr**.

header A *Glish* record containing associated header information common to all sditerators. The following fields are found in **header**:

time An epoch measure giving the midpoint of this integration. See the measures `tool` for a description of how to extract the values from a measure as well as how to work with measures.

scan_number An integer.

source_name

direction A direction measure showing where the telescope was pointed during this integration.

refdirection A direction measure for any reference (off) direction if position switched data has been calibrated.

veldef The velocity definition appropriate to this data.

transition A string describing the transition, if available.

exposure The effective integration time, in seconds.

duration The total time, including any blanking time, spent taking this data, in seconds.

observer

project

resolution The channel resolution at the **reffrequency** in Hz.

bandwidth The total bandwith in Hz.

tcal A vector of CAL temperatures - one for each pixel on the stokes axis, in K.

trx A vector of receiver temperatures - one for each pixel on the stokes axis, in K.

tsys A vector of system temperatures - one for each pixel on the stokes axis, in K.

telescope

telescope_position A position measure holding the location of the telescope.

pressure The atmospheric pressure in hPa.

dewpoint The dewpoint in K.

tambient The ambient temperature in K.

wind_dir The wind direction in radians.

wind_speed The wind speed in m/s.

azel A direction measure holding the azimuth and elevation for this integration (at **time**).

hist A vector of strings holding the history of operations on this data.

other A *Glish* record containing other information from the data set not found elsewhere in this structure. The contents of this field depend on the data source. If the source is a MeasurementSet, this record contains sub-records holding the appropriate values for the various sub-tables in that MeasurementSet. There will be one sub-record for each sub-table and the fields in each sub-record are the same as the column names in that sub-table. If the source is a flat table following the SDFITS convention, then any field not used in the **header** or **data** records is copied to an **sdfits** sub-record of the **other** record.

Example

```
sdit := sditerator('dishdemo2')
nrecords := sdit.length()
scans := array(0,nrecords)
sources := array("",nrecords)
count := 1
rec := sdit.get()
scans[count] := rec.header.scan_number
sources[count] := rec.header.source_name
while (sdit.more()) {
   sdit.next();
   rec := sdit.get()
   count +:= 1
   scans[count] := rec.header.scan_number
   sources[count] := rec.header.source_name
}
# or alternatively ...
vec := sdit.getheadervector("scan_number source_name")
scans := vec.scan_number
sources := vec.source_name
```

The first line makes an sditerator `tool` from a file. The next several lines show how the scan numbers and source names for each sdrecord in that iterator can be retrieved by iterating through the entire sditerator. The final three lines show how all of that could be done using the getheadervector function.

See Also

dish reference manual Using Dish

sditerator - Constructor

Package	**dish**
Module	**dish**
Tool	**sditerator**

An sditerator from an existing file on disk.

Synopsis
sditerator(filename, readonly, selection, lockoptions, correcteddata)

Description
This constructor is used to open an already existing single dish data set. The type of data set (flat-table or MeasurementSet) is automatically determined. MeasurementSets can only be opened read-only. The lockoptions are the same as those appropriate for the table `tool` . The selection record follows the same structure as that used by the select function. The CORRECTED_DATA column is only available for MeasurementSet-based sditerators. When CORRECTED_DATA is requested, the type of the **arr** field in the sdrecord will be complex.

Arguments

filename	in	Name of SDFITS file on disk Allowed:String
readonly	in	Open Read-only? Allowed:Bool Default: T
selection	in	Record describing any selections to do on opening Allowed:*Glish* record Default: [=]
lockoptions	in	locking to be used Allowed:String Default: auto
correcteddata	in	Use the CORRECTED_DATA column? Allowed:Bool Default: F

Returns
sditerator `tool`

Example

```
sdit := sditerator("a302_SP",correcteddata=T);
r := sdit.get();
```

The first line opens an existing data set, using the CORRECTED_DATA column to supply the values for the **arr** field in the returned sdrecord. The second line fetches the first sdrecord found in that sditerator. The type of r.data.arr is complex.

---

newsditerator - Constructor

Package	**dish**
Module	**dish**
Tool	**sditerator**

Make a new, empty sditerator.

Synopsis
newsditerator(filename, type, lockoptions)

Description
Create a new, empty data set of the indicated type on disk and attach an sditerator to it. Currently, only flat-tables can be created. In the future, other types will be available.

Arguments

filename	in	Name of new data set on disk. Allowed:String
type	in	The type of data set. Allowed:String Default: 'Table'
lockoptions	in	locking to be used Allowed:String Default: auto

Returns
sditerator `tool`

Example

```
sdit := sditerator("a302_SP",correcteddata=T);
newit := newsditerator("a302_SP.copy");
for (i in 1:sdit.length()) {
   newit.appendrec(sdit.get());
   sdit.next();
}
```

This example shows how a new sditerator is created and sdrecords from a existing sditerator are copied to it. The first line opens the existing data set. The second line creates a new data set. The loop copies each record from the existing sditerator to the new one.

---

sditfromsdit - Constructor

Package	**dish**
Module	**dish**
Tool	**sditerator**

An sditerator from another sditerator.

Synopsis

sditfromsdit(sdit, selection)

Description

This constructor is equivalent to the select function.

Arguments

sdit	in	An sditerator Allowed:sditerator `tool`
selection	in	Record describing any selections to do on opening Allowed: *Glish* record Default: [=]

Returns

sditerator `tool`

Example

```
sdit := sditerator("a302_SP");
selrec := [=];
selrec.header := [=];
selrec.header.source_name := "My source";
sditSource := sditfromsdit(sdit,selrec)
```

The first line opens an existing data set. The rest make another sditerator by making and applying a selection record to the first iterator. In this example, sources matching the indicated name are selected. The resulting iterator will only have sdrecords which match that selection criteria.

---

sditerator.appendHistory - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Add a history element to the global sditerator history.

Synopsis
appendHistory(newHistory)

Description
This adds the argument to the global history associated with this sditerator. That global history is returned by the history function.

Arguments

newHistory	in	The history string to append. Allowed:String

Returns
Bool

---

sditerator.appendrec - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Put an sdrecord at the end of the sditerator.

Synopsis
appendrec(rec)

Description
Appends this sdrecord to the end of the sditerator. The length of the sditerator is increased by 1 as a result of this operation. The sditerator must be writable.

Arguments

rec	in	The sdrecord to append to this sditerator. Allowed:sdrecord

Returns
Bool

---

sditerator.correcteddata - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Is the CORRECTED_DATA column being used?.

Synopsis
correcteddata()

Description
Determine if the CORRECTED_DATA or FLOAT_DATA column is being used. When the return value is T, the CORRECTED_DATA are being copied to the **arr** field. Otherwise, the FLOAT_DATA values are being copied to the **arr** field. This only works with MeaurementSet based data sets.

Returns
Bool

---

sditerator.deepcopy - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Make a true copy of this data set.

Synopsis
deepcopy(newname)

Description
This functions copies the underlying table to a new table on disk. The select function only returns a reference to the selected portion of the data set (no copy is made). This function can be used to make a copy of that selected portion of the data set. An sditerator `tool` will then need to be constructed using this new data set.

Example

```
# sdit is an already existing data set
sdit.deepcopy("newCopy");
copyOfSdit := sditerator("newCopy");
```

Arguments

newname	in	The name of the disk file to hold the new copy Allowed:String

Returns
Bool

---

sditerator.deleterec - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Delete the sdrecord at the current location pointer.

Synopsis
deleterec()

Description
The sdrecord at the current location pointer is removed from the data set. The length of the sditerator is descreased by one. The sditerator must be writable.

Returns
Bool

---

sditerator.done - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Close this sditerator and disconnect it from the data set.

Synopsis
done()

Description
When invoked, any unwritten data is flushed to disk, the connection with the underlying data set is closed and the `tool` is left unusable.

Returns
Bool

---

sditerator.get - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Get an sdrecord.

Synopsis

get()

Description
Get the sdrecord at the location of the current location pointer.

Returns
sdrecord

---

sditerator.getempty - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Get an empty sdrecord.

Synopsis
getempty(rec, nchan, nstokes)

Description
Get an empty sdrecord with default values for all of the fields and the shapes of the array fields consistent with the indicated shape.

Arguments

rec	out	the requested empty sdrecord Allowed:sdrecord
nchan	in	The number of channels Allowed:Integer
nstokes	in	The number of pixels along the stokes axis Allowed:Integer

Returns
Bool

---

sditerator.getdata - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Get the data part of the sdrecord.

Synopsis
getdata()

Description
Get the **data** part of the sdrecord at the location of the current location pointer. If all you want is information from the **data** record, this may be faster than getting the full record and picking out that information.

Returns
sdrecord

---

sditerator.getdesc - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Get the desc part of the field of the sdrecord.

Synopsis
getdesc()

Description
Get the **desc** part of the **data** field of the sdrecord at the location of the current location pointer. If all you want is information from the **desc** record, this may be faster than getting the full record and picking out that information.

Returns
sdrecord

---

sditerator.getdescvector - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Get a vector of values from the desc record.

Synopsis
getdescvector(fieldNames)

Description
This allows you to get all of the values for any scalar fields in the **desc** portion of an sdrecord in the sditerator with a single function call. This uses the getvectors function but is simpler to use if all of the fields are from the desc record. Here, you only give a vector of field names to extract from the sditerator. The returned value is a record where the field names are the requested fields.

Arguments

fieldNames	in	A vector of desc fields to fetch. Allowed:string vector

Returns
record

---

sditerator.getheader - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Get the header part of the sdrecord.

Synopsis
getheader()

Description
Get the **header** part of the sdrecord at the location of the current location pointer. If all you want is information from the **header** record, this may be faster than getting the full record and picking out that information.

Returns
sdrecord

---

sditerator.getheadervector - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Get a vector of values from the header record.

Synopsis
getheadervector(fieldNames)

Description
This allows you to get all of the values for any scalar fields in the **header** portion of an sdrecord in the sditerator with a single function call. This uses the getvectors function but is simpler to use if all of fields are from the **header** record. Here, you only give a vector of field names to extract from the sditerator. The returned value is a record where the field names are the requested fields.

Example

```
vec := sdit.getheadervectors("scan_number source_name");
scans := vec.scan_number;
sources := vec.source_name;
```

Arguments

fieldNames	in	A vector of header fields to fetch. Allowed:string vector

Returns
record

---

sditerator.getother - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Get the other part of the sdrecord.

Synopsis
getother()

Description
Get the **other** part of the sdrecord at the location of the current location pointer. If all you want is information from the **other** record, this may be faster than getting the full record and picking out that information.

Returns
sdrecord

---

sditerator.getvectors - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Get a vector of values for the fields in the template.

Synopsis
getvectors(template)

Description
This allows you to get all of the values for any scalar fields in an sditerator with a single function call. The template has the same structure as an sdrecord except that you only supply those fields that you want extracted. The returned value has the same structure as the template, but the field values are filled in with a vector containing each value from each sdrecord in the sditerator, in order. This function is tuned for a few fields so that they can be retrieved relatively quickly if they are the only fields in the template. These tuned fields are those used in the dish selection GUI: **source_name**, **scan_number**, **time**, **chan_width**, and **restfrequency**.

Example

```
# tediously extract the individual values
```

```
sdit.origin();
nrecords := sdit.length();
scans := array(0,nrecords);
sources := array("",nrecords);
restfqs := array(0.0,nrecords);
count := 1;
rec := sdit.get();
scans[count] := rec.header.scan_number;
sources[count] := rec.header.source_name;
restfqs[count] := rec.data.desc.restfrequency;
while (sdit.more()) {
   sdit.next();
   rec := sdit.get()
   count +:= 1
   scans[count] := rec.header.scan_number;
   sources[count] := rec.header.source_name;
   restfqs[count] := rec.data.desc.restfrequency;
}

# or alternatively using getvectors
template := [data=[desc=[restfrequency=0.0]],
             header=[scan_number=0,source_name=""]];
vec := sdit.getvectors(template);
scans := vec.header.scan_number;
sources := vec.header.source_name;
restfqs := vec.data.desc.restfrequency;
```

Arguments

template	in	The template indicating which fields to fetch. Allowed:record

Returns
record

---

sditerator.history - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Get the global history associated with all sdrecords

Synopsis
history()

Description
This history describes the creation of this sditerator and any selection and other global operations done on this sditerator. This history is appended to the sdrecord **hist** field when each sdrecord is fetched.

Returns
Bool

---

sditerator.id - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Return the ID record of this sditerator.

Synopsis
id()

Description
Used by the sdaverager tool to determine which sditerator is being used.

Returns
Record

---

sditerator.iswritable - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Can sdrecords be stored (written) to this sditerator?

Synopsis

iswritable()

Description
This returns the value of the readonly argument used at construction for this sditerator. Currently, only flat-table based sditerators can be written to.

Returns
Bool

---

sditerator.length - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Get the total number of sdrecords in this sditerator.

Synopsis
length()

Description
Returns the total number of sdrecords that are found in this sditerator. Note that for the flat-table data set, this may require examining each row of the table to see if there is more than one sdrecord stored in that row hence this may be an expensive operation for that type of table. Using more and next is always a faster way to step through (iterate) an sditerator (rather than using length and setlocation in a for loop).
ahreturnsInteger

---

sditerator.location - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Get the value of the location pointer.

Synopsis

location()

Returns
Integer

---

sditerator.more - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Are there more sdrecords after this location?

Synopsis
more()

Description
This function returns T if the location pointer is pointing at the last sdrecord in this sditerator.

Returns
Bool

---

sditerator.name - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

What is the name of the underlying data on disk.

Synopsis
name()

Returns
String

---

sditerator.next - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Set the location pointer to the next record.

Synopsis
next()

Description
This function advances the location pointer by one unless the pointer is already at the end. A subsequent use of the get function would return the sdrecord at that location.

Returns
Bool

---

sditerator.unlock - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Release the lock on all associated tables.

Synopsis
unlock()

Description
This function releases any table locks that may exist on the tables that are associated with this sditerator. Normally this isn't necessary but if you are trying to work directly with the associated tables through a table tool using unlock and lock should make the interaction happen faster since the the auto-locking mechanism won't need to periodically check to see if a lock needs to be released.

Returns

Bool

---

sditerator.lock - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Set the appropriate lock on all associated tables.

Synopsis
lock(nattempts)

Description
This function set table locks on the tables that are associated with this sditerator. Normally this isn't necessary but if you are trying to work directly with the associated tables through a table tool using unlock and lock should make the interaction happen faster since the the auto-locking mechanism won't need to periodically check to see if a lock needs to be set. The nattempts argument is used to indicate how many additional attempts to acquire the locks the tool should make before giving up. Each additional attempt happens after about a 1 second pause. If nattempts is 0, only the initial attempt will be made. If the locks can't be set, the return value will be false (F).

Arguments

nattempts	in	The number of times to attempt to set the lock. Allowed:Integer

Returns
Bool

---

sditerator.origin - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Set the location pointer to the first record.

Synopsis
origin()

Description
This function resets the location pointer to the first record in the sditerator. A subsequent use of the get function would return the sdrecord at that location.

Returns
Bool

---

sditerator.previous - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Set the location pointer to the previous record.

Synopsis
previous()

Description
This function decrements the location pointer by one unless the pointer is already at the origin. A subsequent use of the get function would return the sdrecord at that location.

Returns
Bool

---

sditerator.put - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Put an sdrecord at the current location pointer.

Synopsis
put(rec)

Description
Replaces the sdrecord at the current location pointer with this sdrecord. The sditerator must be writable.

Arguments

rec	in	The replacement sdrecord Allowed:sdrecord

Returns
Bool

---

sditerator.reselect - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Re-apply the previous selection to the parent data set.

Synopsis
reselect()

Description
This function re-applies the previous selection record to the parent data set of this iterator. If there is no previous selection, this function does nothing. This function is useful if the data set has changed since the last selection.

Returns
Bool

---

sditerator.resync - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Resynchronize with the data on disk.

Synopsis
resync()

Description
This function resynchronizes the internals of the sditerator to the data on disk. The default locking mode does that automatically and so this normally will not need to be used. Some locking modes do not do that. In that case, another process (e.g. an on-line filler) can write to the data on disk. In that case, the sditerator will not see those changes until this function is called.

Returns
Bool

---

sditerator.flush - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Flush the contents to disk.

Synopsis
flush()

Description
This function flushes the internals of the sditerator to the data on disk. The default locking mode does that automatically and so this normally will not need to be used. Some locking modes do not do that. In that case, another process (e.g. an the table client) may read data written through the sditerator. In that case, the data in the sditerator may not be available to that other process until it has been explicitly flushed to disk.

Returns
Bool

---

sditerator.select - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Make a new sditerator by applying a selection criteria record

Synopsis

select(selection)

Description

A new sditerator is constructed by applying the indicated selection criteria record to this sditerator. The returned sditerator is an sditerator which references this sditerator. Only sdrecords which match the criteria are found in the returned sditerator. The returned sditerator may be empty. The structure of the selection criteria record follows the structure of an sdrecord. Only fields which are to be selected on need be included in the selection criteria record. Only scalar fields (strings and numeric fields) in the sdrecord can be part of the criteria. Fields which are measures or quanta records in the sdrecord can be selected if they are single valued. The values used in the selection record should be in the same units as the first sdrecord in this sditerator. For string fields, the selection criteria value for that field should be a vector of values to match during the selection. A simple wildcard can be indicated using an asterisk. For numeric fields, the selection criteria should be a 2-dimensional array of values having dimensions (2, nranges). In other words, this is a set of range pairs. Values in this sditerator for that field which fall within that range or are equal to the range endpoints are considered to match this criteria.

Arguments

selection	in	Selection criteria record Allowed: *Glish* record Default: [=]

Returns

sditerator `tool`

Example

```
sdit := sditerator('a302_SP');
sel := [header=[scan_number=array(0,2,3),source_name="UGC1*"]];
sel.header.scan_number[,1] := [-1, 10];
sel.header.scan_number[,2] := [15,15];
sel.header.scan_number[,3] := [50,9999999];
s2 := sdit.select(sel);
```

In this example, scan numbers which are less than or equal to 10 or equal to 15 or greater than or equal to 50 are selected, so long as the source name matches the one given in the example (a wild card is used here).

---

sditerator.sel2string - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Convert a selection record to a string.

Synopsis
sel2string(selrec)

Description
This function converts a selection record to a string so that it can be stored in the history record associate with that iterator. This is used by the constructors and will likely not be used by the average user.

Arguments

selrec	in	Selection criteria record
		Allowed: *Glish* record

Returns
String

---

sditerator.setlocation - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Set the location pointer to the indicated record.

Synopsis
setlocation(location)

Description
This function sets the location pointer to the indicated record number. Valid record numbers run from 1 to the length of the sditerator. If the location argument is outside this range, the location pointer is set to the appropriate end point. A subsequent use of the getfunction would return the sdrecord at that location.

Arguments

location	in	The desired record number. Allowed:Integer

Returns
Bool

---

sditerator.stringfields - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Returns a record indicating which fields in the sditerator are strings.

Synopsis
stringfields()

Description
This is used in various places as a way of ensuring that the dish selection mechanism does the appropriate thing.

Returns
Record

---

sditerator.type - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Return the type of the underlying data.

Synopsis
type()

Description
Returns either "MeasurementSet" or "Table" to indicate the type of data underlying this sditerator.

Returns
String

---

sditerator.usecorrecteddata - Function

Package	**dish**
Module	**dish**
Tool	**sditerator**

Use the CORRECTED_DATA column?.

Synopsis
usecorrecteddata(correcteddata)

Description
Toggle the use of the CORRECTED_DATA or FLOAT_DATA column. When T, the CORRECTED_DATA are copied to the **arr** field. Otherwise, the FLOAT_DATA values are copied to the **arr** field. This only works with MeaurementSet-based data sets.

Arguments

correcteddata	in	Use the CORRECTED_DATA column? Allowed:Bool

Returns
Bool