NOTE 225 – Using Dish : The AIPS++ Single Dish Analysis Environment

R. W. Garwood, J. P. McMullin

14 September 1999

Contents

1 Introduction **2**

2 Getting Data Into Dish **3**
2.1 UniPOPS SDD -> AIPS++ table 3
2.2 SDFITS -> AIPS++ table 4

3 The Dish Graphical User Interface **4**
3.1 Results Manager 4
3.1.1 Working sets, SDRECORDs, and everything else . . . 5
3.2 Menubar and Message Line 5
3.3 Browsing 6
3.4 Inspecting 6
3.5 Operations 6
3.5.1 Averaging 6
3.5.2 Baselines 8
3.5.3 Calculator 8
3.5.4 Function on Data 11
3.5.5 Multi-Operation 11
3.5.6 Re-gridding 13
3.5.7 Saving 13
3.5.8 Selection 13
3.5.9 Smoothing 13
3.5.10 Statistics 13
3.5.11 Write to File 16

4 Saving/Restoring State **18**

5 Dish at the *Glish* Command Line **19**

6 The Dish Plotter **19**

7 SDImager **25**
7.1 lookmap 26
7.2 spectramap 26
7.3 contourmap 29

8 Recipes **32**
8.1 Recipe 1: Reduce an ON/OFF Total Power scan 32
8.2 Recipe 2: Perform a gaussian fit on a spectral line. 33
8.3 Recipe 3: Add a function to DISH (or fun with extensibility) 34

9 Development Plan **36**

A The sdrecord **36**

B sditerator **40**

C SDFITS **40**
C.1 EXTNAME keyword 41
C.2 Virtual columns 41
C.3 The DATA column and the DATA axes 41
C.4 CORE keywords and columns 44
C.5 SHARED keywords and columns 45
C.6 Other columns 47
C.7 Multiple SDFITS tables in a single file 47

1 Introduction

DISH is a collection of *Glish* scripts and clients which provide an environment within AIPS++ intended to be used for single dish radioastronomy data analysis. Its initial aim is to be a worthy replacement for traditional single dish analysis programs such as UniPOPS. Eventually it will provide access to more advanced data calibration and imaging utilities which will share the same basic design as the sythesis calibration and imaging utilities in AIPS++.

To start DISH, you must type the following at the glish command line after starting AIPS++:

```
include "dish.g"
```

In DISH, wherever practical, what you see is what you are operating on. So, for example, when you press the "Apply" button in the "Baselines" graphical user interface (GUI) frame of DISH, a baseline will be fit to whatever is currently plotted in the DISH plotter. Results appear in the results manager and, when appropriate, are immediately displayed on the plotter. Results can be moved between the results manager, the calculator, and the browser through the use of a menu which appears when the user presses the right mouse button in one of these frames.

The current state of DISH (the contents of the results manager, the settings of all of the operations, the contents of the calculator, and any active browsers) can be saved at any time either to the default state file or to a location of the user's choosing. DISH can be restored to a previously saved state. By default, DISH saves its state upon exit from glish and restores itself to that same state when DISH is started in a new *Glish* session.

In this release DISH only uses data in AIPS++ tables which have been filled from FITS binary tables which follow the single dish FITS convention (SDFITS) described in Appendix C. Future versions will interact directly with AIPS++ MeasurementSets containing single dish data.

2 Getting Data Into Dish

2.1 UniPOPS SDD -> AIPS++ table

Converting from UniPOPS SDD data to an AIPS++ table readable by DISH currently requires three steps:

1. Convert UniPOPS SDD data to UniPOPS FITS binary table.

   Use the UniPOPS utility *uni2fits* to convert the SDD data to FITS binary format. To use the program, you must be logged onto a Sun workstation where UniPOPS has previously been installed and have your path include:

   ```
   export popsdir=~unipops/test/
   export PATH=$popsdir/sunbin:$popsdir/utilities:$PATH
   ```

2. Convert UniPOPS FITS to standard SDFITS.

   Use the AIPS++ utility *uni2sdfits* to convert the data file. The syntax is:

```
uni2sdfits input=unipops_fits_file output=sdfits_file
```

3. Convert the SDFITS file to an AIPS++ table. See the next section for the syntax of this step.

2.2 SDFITS -> AIPS++ table

Use the AIPS++ utility *fits2table* to convert from a binary table following the SDFITS convention to an AIPS++ table. The syntax is:

```
fits2table input=sdfits_file output=aips2_table [which_hdu=#]
```

where "sdfits_file" is the name of the SDFITS file and "aips2_table" is the desired name of the new AIPS++ table created by this step. The third argument, "which_hdu", is optional. Use this to specify a specific FITS header data unit (hdu) other than the first one after the primary hdu. For example, if you have a FITS file which has three tables following the primary hdu and the second table is a valid SDFITS table then you would use `which_hdu=2` to indicate that that hdu is the one to be converted to the indicated output AIPS++ table.

3 The Dish Graphical User Interface

The following sections detail the major graphical compenents of DISH.

3.1 Results Manager

The Results Manager is the heart of DISH. All results which DISH creates in response to user actions are stored in the results manager and listed in the *Results* window. These results are all available at the *Glish* command line as glish variables of the same name for the user to interact with. DISH immediately plots the currently selected result if it can be plotted. A description is associated with each result. The user can change the description or the name of the result if they choose to do so. Certain results can also be browsed. Since each result is also a glish variable of the same name, new names are limited to be a valid glish variable name.

Results can be copied to and from the clipboard or to the calculator. This Copy/Paste facility is activated by pressing the right mouse button when the mouse pointer is over the *Results* window. A popup menu appears with the three possible options. The current selection is copied to the chosen destination when a copy option is selected. The contents of the cliboard is pasted into the results manager when that option is selected.

3.1.1 Working sets, SDRECORDs, and everything else

Results are either working sets (also known as sditerators), SDRECORD, and anything else.

A working set is a collection of SDRECORDs. Specifically, it is an sditerator tool. When you open a data set through the *File* menu (either a new one or an existing one) the result is a working set which will be shown in the results manager. The other way to create a working set is to use the "Apply" button in the "Selection" operation. When you make a selection this way the result is also a working set. It does not make a copy of the selected data, rather it makes a reference to the selected data in the underlying data. This means that if you change the underlying data, the reference in the selected data is also changed. Working sets can be browsed. See the Reference Manual for more information on the Sditerator `tool`.

An SDRECORD is a *Glish* record which has the structure described in Appendix A. It is data (generally spectra at this point in DISH development) plus some standard header values, any non-standard header values, and a history of what has been done to that particular SDRECORD. SDRECORDs can be browsed and plotted.

Anything may appear in the results manager. If it isn't a working set or an SDRECORD it can not be browsed or plotted.

3.2 Menubar and Message Line

New data files are created, existing data files are opened, the state of DISH is saved, a previously saved state is restored, and DISH is reset to its default initial state through the *File* menu. The GUI panels for each available operation are enabled and dismissed through the *Operations* menu. The *Options* menu has two options which can be turned on or off. The "Write to Script" option, when turned on, causes DISH to echo the underlying glish commands to the AIPS++ defaultscripter (ds) tool when a GUI operation occurs in DISH. This can be a way to learn more about what DISH is doing behind the scenes and write glish scripts to do operations not yet available in DISH. When the "Save when done" option is turned on, the state of DISH is saved to the current state file when the "done" function is invoked (e.g. `dish.done()`). The "done" function is always invoked whenever you exit glish. Because DISH can not be restarted after "done" has been called it is best not to do that unless you are about to exit glish. This deficiency will be corrected in the next release.

The Message Line is the text immediately below the Results Manager.

These messages are also echoed to the AIPS++ logger for longer term storage.

3.3 Browsing

A working set within the results manager can be 'browsed' or examined by hitting the 'Browse' button. This brings up a frame which lists all of the scan numbers and objects, record by record, within the working set. Each SDRECORD can also be browsed to examine all of the contents (e.g., all of the header information and data).

When browsing a working set, individual records can be copied to the clipboard or to the results manager. This copy facility is activated by pressing the right mouse button when the mouse pointer is over the browser window. A popup menu appears with the three possible options. The current selection is copied to the chosen destination when one of the copy options is selected.

3.4 Inspecting

Inspecting is another way of viewing the values of the results within the results manager. It lists all of the contents of the selected result. These may be accessed via the *Glish* command line by the names shown while inspecting.

3.5 Operations

Currently there are eleven operations pre-defined within Dish. These operations are general utilities available from the basic Dish Gui. All operations have two basic buttons: a green Apply button which will execute the given operation and a yellow Dismiss button which will close the operation frame but retain all information in that frame for subsequent use.

3.5.1 Averaging

The Averaging operation offers several options for performing the averaging (Figure 1). Alignment which may be either None, By Velocity, or By X-Axis, 2) Rest Frequency, which allows shifting to match the first in the average (for example, for averaging spectra from transients), 3) Weighting which may be None, Tsys and Time or by RMS, and 4) which will use any selections chosen from the Selection Operation prior to averaging.

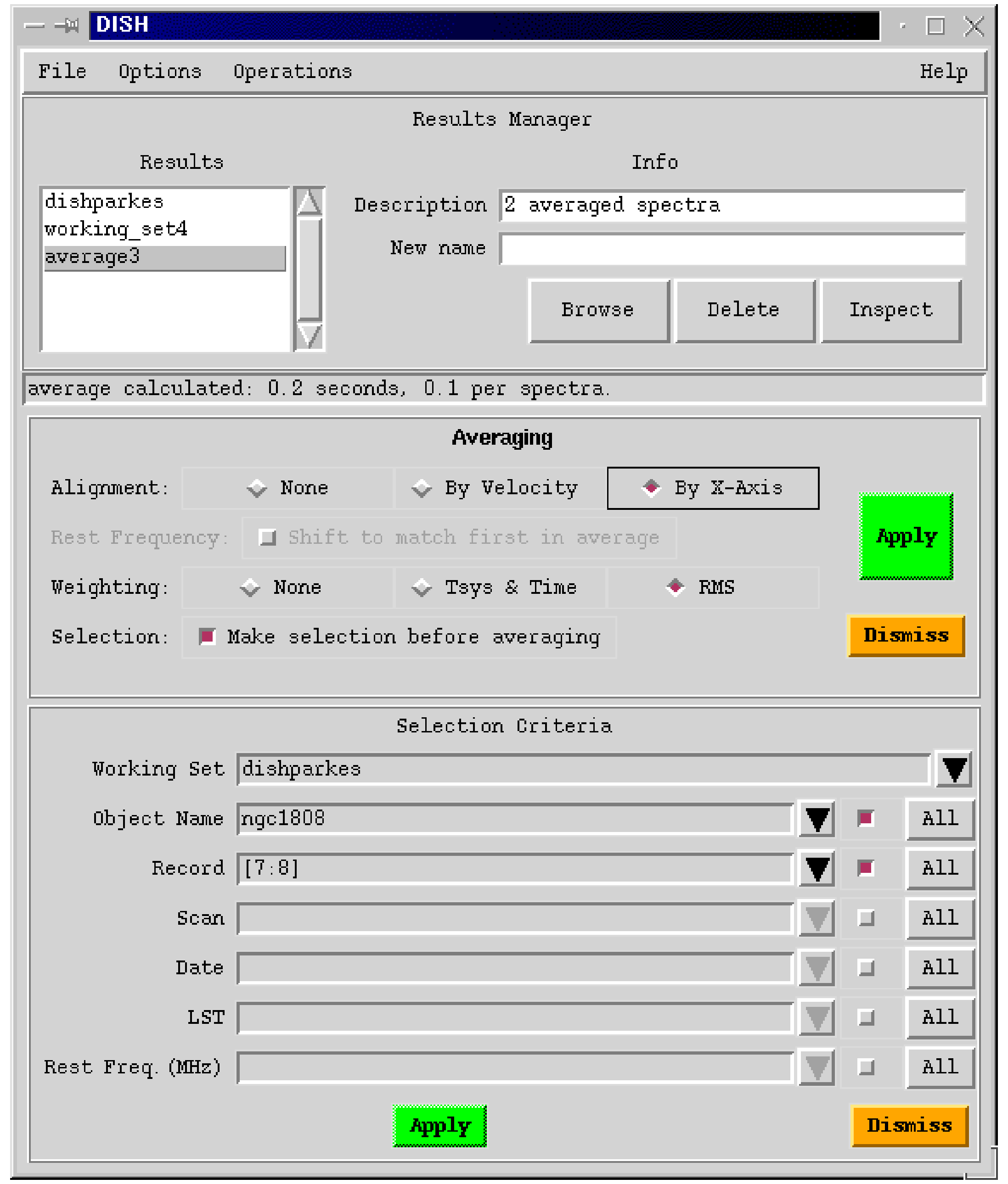

Figure 1: Example window showing basic Dish frame with the Averaging and Selection Operation frames open.

3.5.2 Baselines

The Baseline operation allows either a polynomial fit or a sinusoid fit. The baseline regions may be typed into the regions entry box at the bottom of the frame, or selected via cursor (the Plot Ranges and Cursor Active buttons should both be checked). The baseline operation frame can be seen in Figure 2.

The default action is to calculate the baseline and show it. Once the fit seems satisfactory, the 'Subtract' button must be checked and the fit 'Apply'd. This will remove the baseline and display the resultant spectrum. The baseline fit may be viewed from the results manager. The current ranges selected will be displayed in the entry field. A history of these is preserved within a cache which can be viewed by clicking on the down-arrow button. The RMS of the fit is displayed to the right of the frame. Units for ranges or fit parameters (sinusoid only) may be converted between channels and X-Axis units using the Convert button.

3.5.3 Calculator

The Calculator operation is a general tool for manipulating spectra in ways not facilitated by the basic set of operations. Spectra may be transferred into the calculator stack through the use of the Copy/Paste to Clipboard facility.

The Copy/Paste facility is activated by the right mouse button. A popup menu appears with a two copy options and a paste option. "Copy to clibpoard" will copy whatever is selected in the Calculator stack listbox to the Clipboard; the Clipboard is simply a convenient virtual storage area. "Copy to results manager" copy whatever is selected in the Calculator stack listbox to the results manager. "Paste from clipboard" retrieves whatever was most recently sent to the clipboard. For example, in Figure 3, the Results Manager spectrum "average3" was copied into the Calculator stack. An "_1" designation was added to the label to distinguish it from the same value in the Results Manager. The spectrum was then multiplied by a factor, producing "average3_1_2". The additional id tag is an underscore and the stack number.

The available functions may be seen in the figure. In addition, it works as any standard Reverse Polish Notation calculator. Numbers, vectors and arrays may be typed into the entry field. The calculator will only work with SDRECORDs, numbers, vectors and arrays.

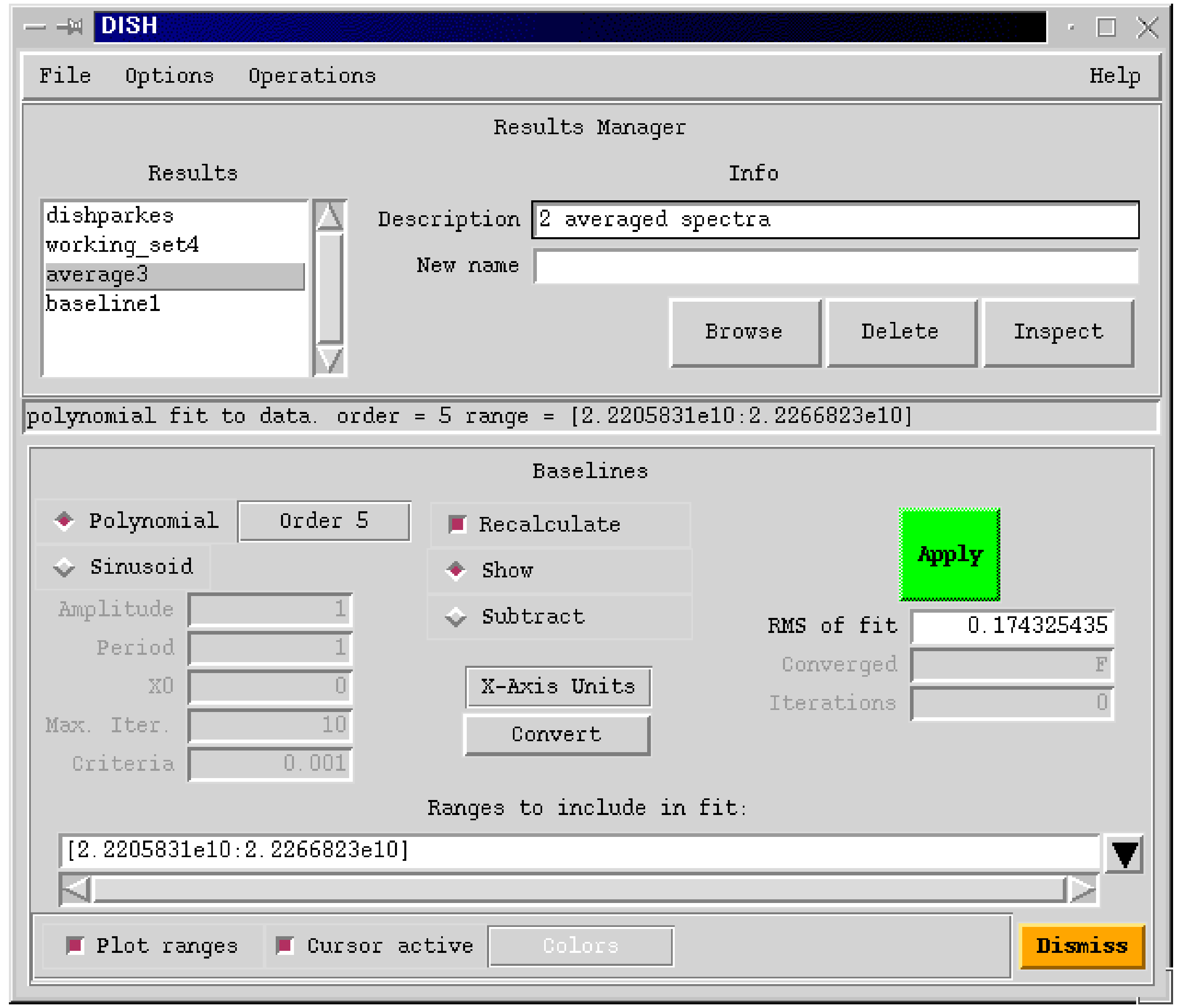

Figure 2: Example window showing basic Dish frame with the Baseline Operation frame open. In this example, one baseline region is selected and a 5th order polynomial will be shown overlayed on top of the spectrum.

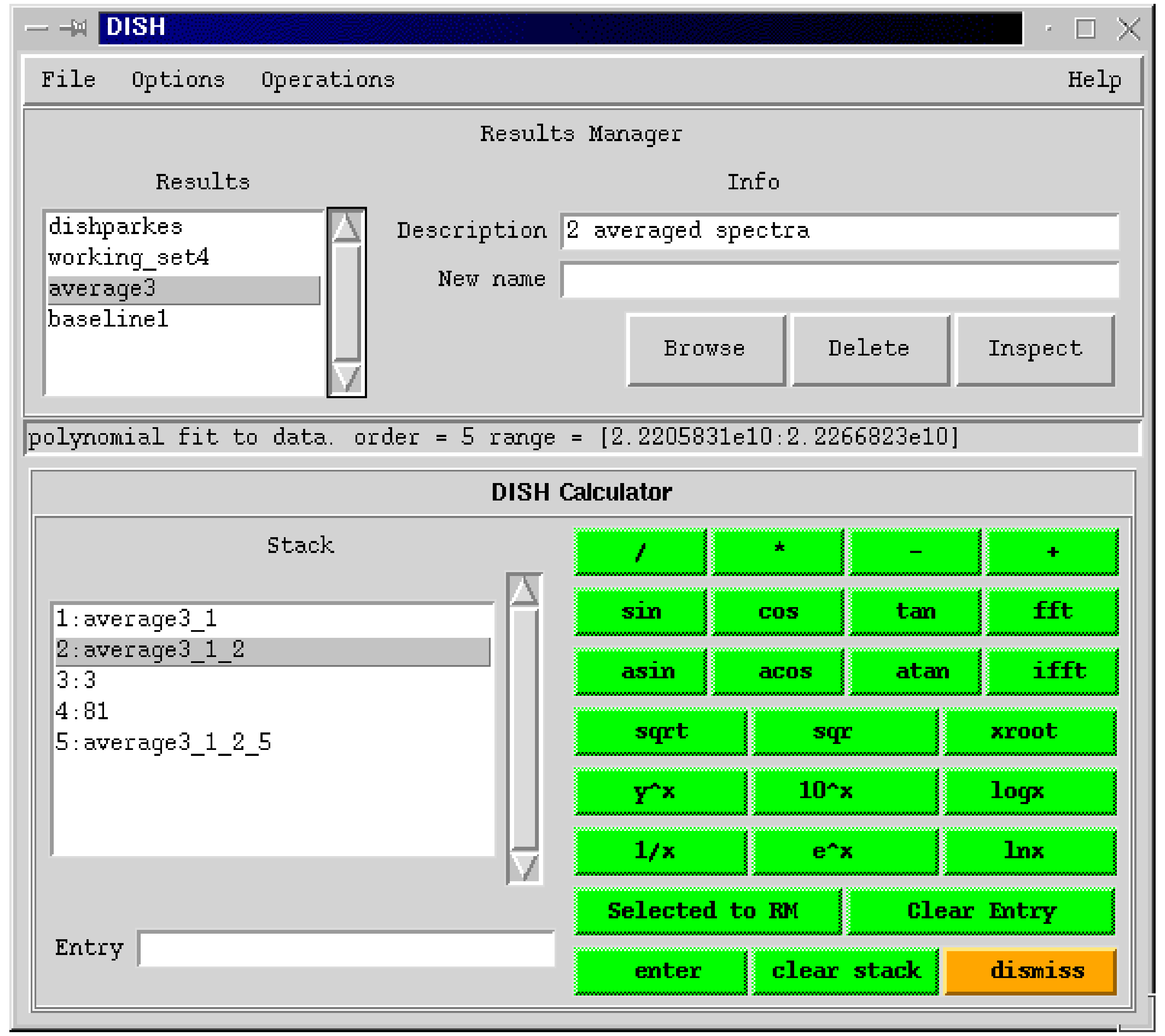

Figure 3: Example window showing basic Dish frame with the Calculator Operation frame open

3.5.4 Function on Data

The Function on Data operation is another means of manipulating the displayed data. The string typed in the entry box of this operation is evaluated by glish after any of the standard macros are replaced by their equivalent glish. The result of this evaluation is then reassigned to the data array of the last viewed SDRECORD. The standard macros and their meanings are:

DATA sdrecname.data, the data record of sdrecname.

DESC sdrecname.data.desc, the data description record of sdrecname.

ARR sdrecname.data.arr, the data array.

HEADER sdrecname.header, the header record.

NS_HEADER sdrecname.ns_header, the ns_header record.

In the above definitions, sdrecname is whatever the name of the last viewed data set. For these macro substitutions to work, they must be given as upper case (to distinguish them from their lower case equivalents, in which no substitution would occur).

Since whatever this text evaluates to after macro substituion is reassigned to the data array, it must have the exact same shape as the data array initial had.

Basic scaling or manipulations can be obtained by typing expressions that operate on ARR. For example, a simple scaling can be obtained by typing the function "ARR*1.5", and then pressing "Apply". Any function available from within *Glish* or known at the command line (e.g. a function defined by the user during the run of Dish), can be accessed and used in these expressions.

3.5.5 Multi-Operation

The Multi-Operation Frame allows the user to group sequences of commands that will be used repeatedly within a working set on spectra into a single operation. In Figure 4, an Average, Baseline and Smoothing operation are grouped under the cache name of 'mylist'. Hitting the Apply button will execute the sequence of commands in their given order. The combination of the Add and Delete buttons allows easy manipulation of the command list caches.

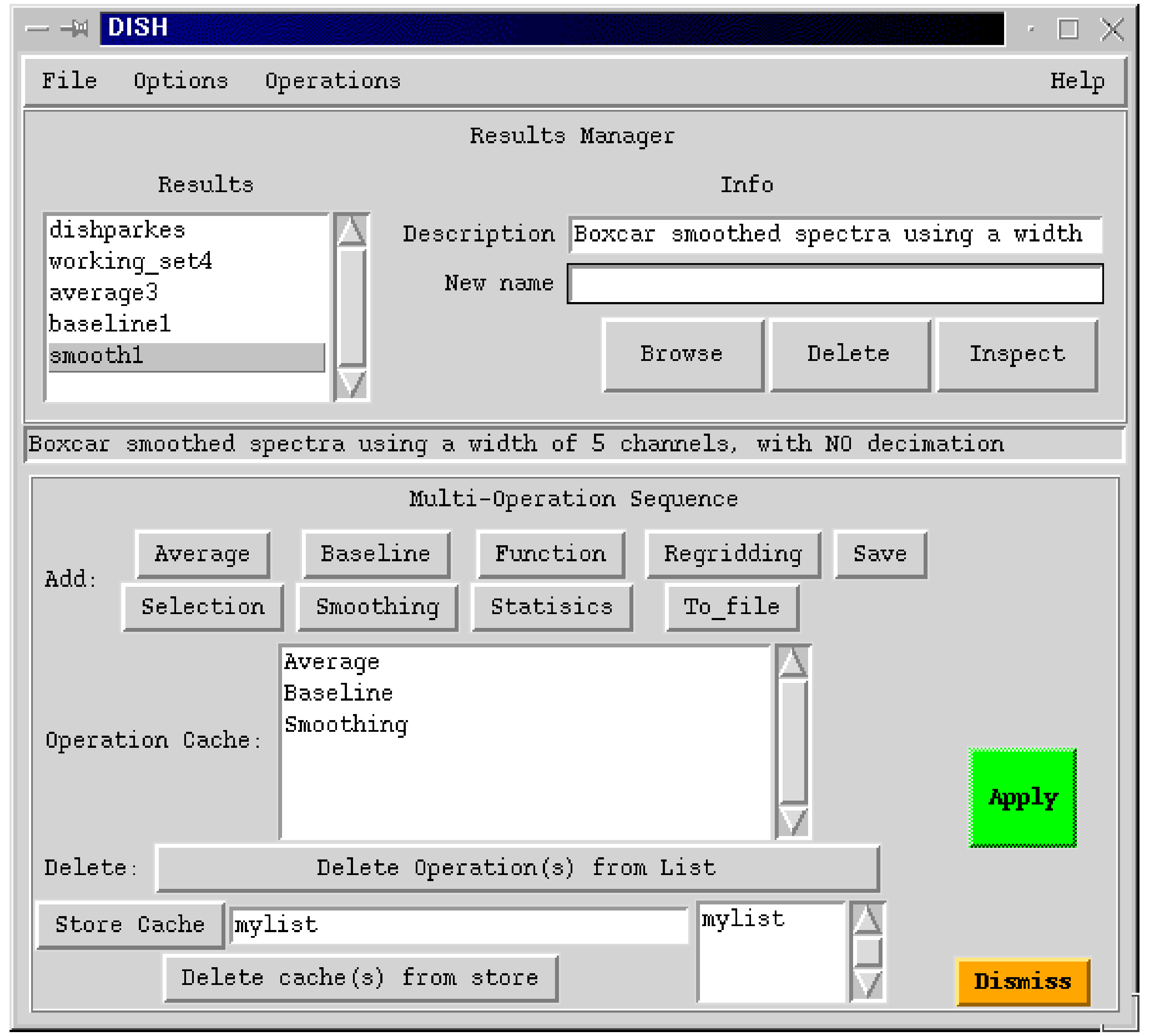

Figure 4: Example window showing basic Dish frame with the Multi-Operation Operation frame open

3.5.6 Re-gridding

The Re-gridding operation is a more generalized version of the Smoothing Operation as it allows both finer and coarser adjustments of the grids. Currently, it allows Hanning, Boxcar and Gaussian smoothing along with Spline and Fourier Transform Interpolations.

3.5.7 Saving

The Saving operation allows the user to store and organize spectra into named working sets. The working set must first be defined through the File Menu option "Open" (selecting on Read and Write). Once the working set is defined, any displayed spectrum may be written to the working set on disk.

3.5.8 Selection

The Selection operation allows a number of selection criteria to be chosen to create new working sets which are subsets of existing working sets. The recognized terms are: Object Name, Record, Scan, Date, LST, and Rest Frequency. Typing into any of the entry fields is allowed but the selection will only be used if the check box on the right is marked. A history of selected ranges or names will be preserved. For numerical selections, ranges may be made as follows:

```
Record,Scan Entry:
8,9,10 # Select scans 8, 9 and 10
8,[10:13],17 # Select scans 8, 10, 11, 12, 13, and 17
```

The 'All' button will select all relevant values for the given selection criteria.

3.5.9 Smoothing

The Smoothing operation is a more specific tool than the Re-gridding operation. It offers easy Hanning, Boxcar and Gaussian smoothing.

3.5.10 Statistics

The Statistics operation calculates basic statistics on an interval of the spectrum. The interval must be defined by the cursor (activate the Plot Range and Cursor Active buttons) and then left click on the left and right regions

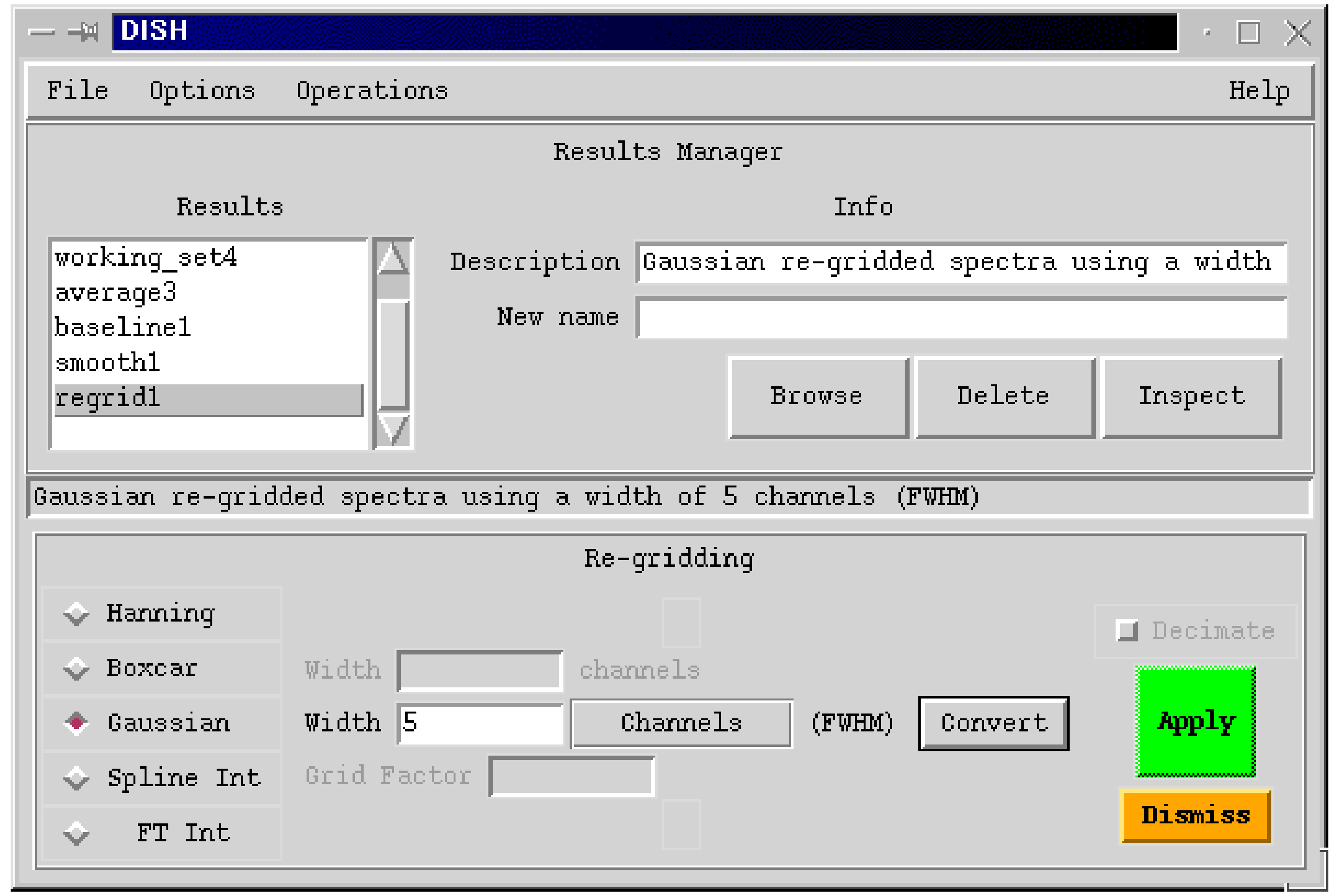

Figure 5: Example window showing basic Dish frame with the Regrid Operation frame open

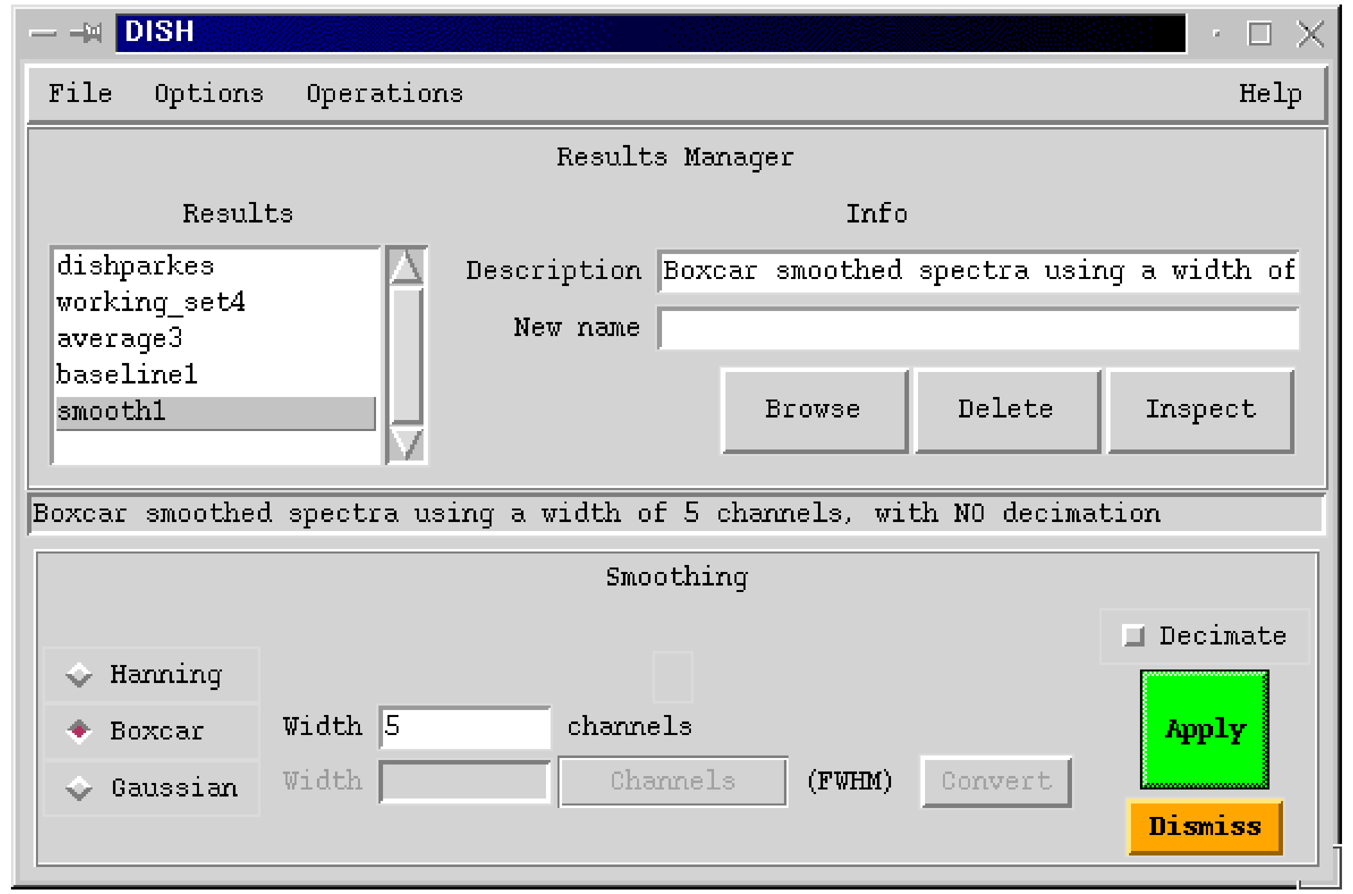

Figure 6: Example window showing basic Dish frame with the Smoothing Operation frame open

of the spectrum which are of interest. The region will appear in the Range entry field. The units may be converted between X-Axis units and channels through the convert button (similar to the Baseline operation capability). The Apply button will calculate the Peak value, the location of the peak, the Area over the region, the location of the centroid (half area), the rms, and the minimum. These values may be written as a record to the clipboard through the "Copy to CB" button. The record looks as follows:

```
- statrec:=dcb.paste()
- statrec
[peak=2.09596825, area=29425582.1, min=-0.26479581, rms=0.580018938, scan=20, ce
ntroid=2.22369707e+10, vpeak=2.22353317e+10]
-
```

In addition, the data may be written to a file through the "Print to File" button. This pulls up a frame which prompts the user for a filename. If the file exists, it will append the values to the end of the file. The file looks as follows:

```
  scan  start          stop           atPeak         Centroid
    20  2.2225160e+10  2.2245390e+10  2.2235332e+10  2.2236971e+10
        Peak           Area           Mininum        rms
        2.0959682e+00  2.9425582e+07 -2.6479581e-01  5.8001894e-01
```

3.5.11 Write to File

This operation allows convenient output to a disk file in the current directory. It automatically defaults to the file name of "scan_number.spc"; for example if you Write to File while looking at scan 20, the file will be "20.spc". The file looks as follows:

```
     FREQ              other axis
       Hz              other unit
22203023360.000000    -3.907563
22203086216.617188    -4.986684
22203149073.234375    -4.020397
22203211929.851562    -4.558375
22203274786.468750    -4.556173
22203337643.085938    -3.856671
22203400499.703125    -4.001695
```

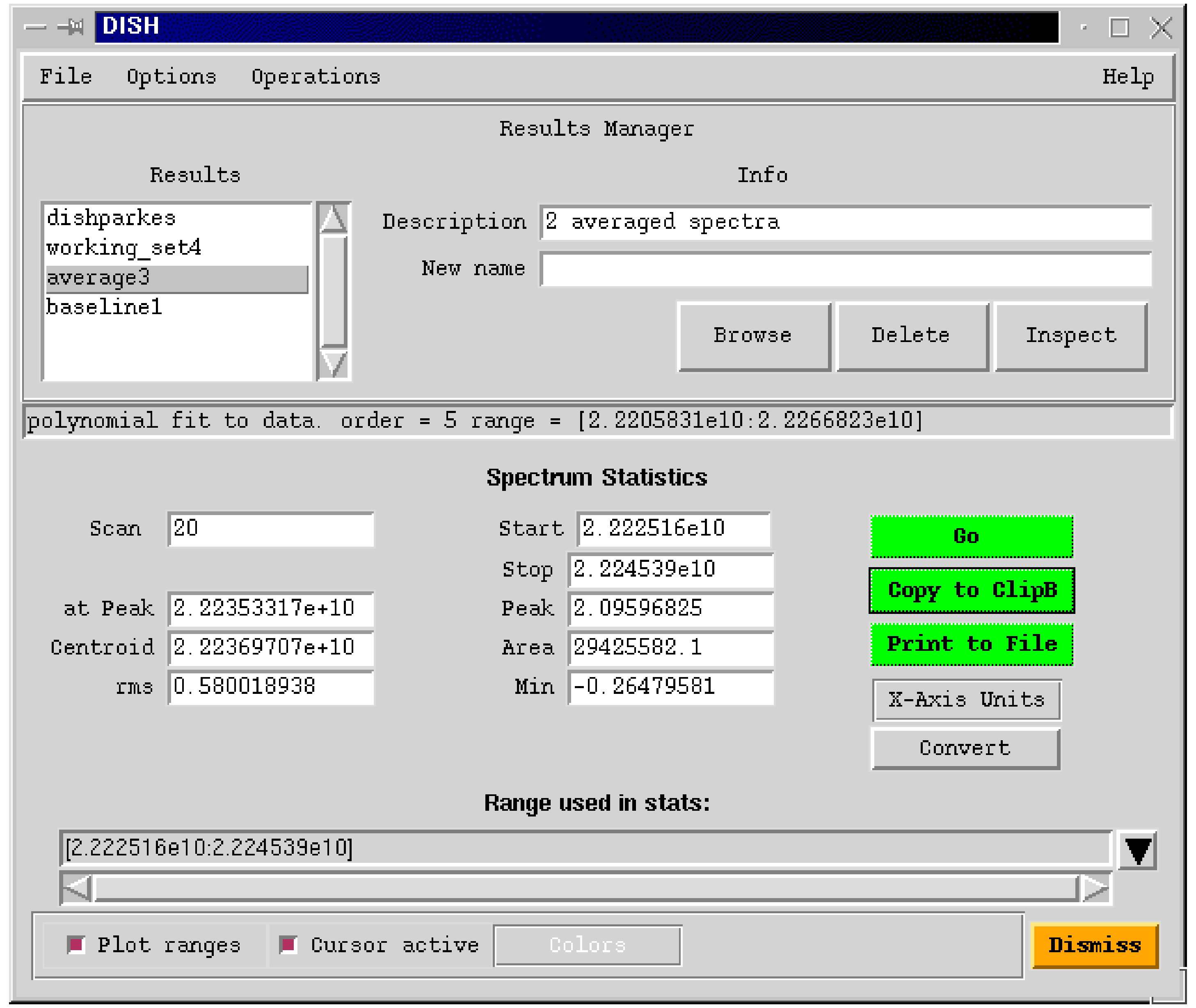

Figure 7: Example window showing basic Dish frame with the interval Statistics Operation frame open

```
22203463356.320312    -3.464213
22203526212.937500    -4.438780
22203589069.554688    -3.296077
...etc.
```

4 Saving/Restoring State

If you do nothing, a file called *default* will be created within the `$HOME/aips++/dishstate` directory upon exiting from a DISH session. The file is created automatically and dish will be restored to the state saved in that file automatically upon startup. The dish state file contains all of the values known to the results manager as well as the various settings for each GUI element and operation. So, by default a record is kept of the state of DISH after each use of DISH and DISH is restored to this state the next time DISH is used.

There are several ways in which you can alter this default behavior.

- The state can be saved at any time by selecting the *File/Save state* menu item.

- A state can be restored to a previously restored state at any time by selecting the *File/Restore state* menu item.

- The state can be saved to a state file other than the default file at any time by selecting the *File/Save state as ...* menu item. This makes the AIPS++ catalog tool visible and allows you type in or select a file name to hold the state information. Once you have made your selection and the state saved to that file, that file becomes the default state file until changed with *Save state as ...* or *Restore state from ...*.

- The state can be restored from a state file other than the default file at any time by selecting the *File/Restore state from ...* menu item. This makes the AIPS++ catalog tool visible and allows you to select a file name to retrieve the state information from. Once you have made your selection and the state retrieved from that file, that file becomes the default state file until changed with *Save state as ...* or *Restore state from ...*.

- Restore the state to the default initial DISH state by selecting the *Reset to default state* menu item.

- The default behavior of saving the state when `dish.done()` is invoked (which happens which you exit glish) can be turned off by toggling that option in the *Options* menu.
- The default behavior of restoring from the default state file when DISH is first started can be turned off by setting the aipsrc variable "dish.restorestate" to F (for False). To turn it back on, set this variable to (T) or remove it from your `.aipsrc` file.
- The default state file can be set before DISH starts by setting the aisprc variable dish.statefile to whatever file you wish to designate as the default state file. This is always the state file used when DISH starts up. Its default value is `$HOME/aips++/dishstate/default`

5 Dish at the *Glish* Command Line

All of the values in the DISH results manager can be manipulated at the *Glish* command line. These results can then be returned to the DISH results manager. Please see Recipe 1 for an example.

6 The Dish Plotter

The Dish Plotter is the familiar AIPS++ pgplotter tool, modified for additional utilities and capabilities. Any activity or task within Dish which produces a plottable result, will automatically have the result displayed in the Dish Plotter; for example, selecting entries within the Results Manger, the SDRecord Browser, or Calculator Operation stack will plot the selection chosen. The plotter is arrayed as shown in Figure 8.

A: File Menu This is the standard pgplotter File menu. Open...: This operation will read in an plot file on disk written by the Save command. Save: This operation will write a vector plot file of the displayed image to disk. Save will write to the file specified in feature L (defaults to dish.plot). Preview...: This operation brings up a ghostview session with the currently displayed plot shown. This feature allows you to see how such a plot will look printed out on a page. Print: This operation will print the current spectrum. Save...: This operation will also write a vector plot file of the displayed image but allows you to specify the file name unlike the 'Save' feature and the quicksave button (feature L). Exit: Exits from the Dish Plotter; closes the frame.

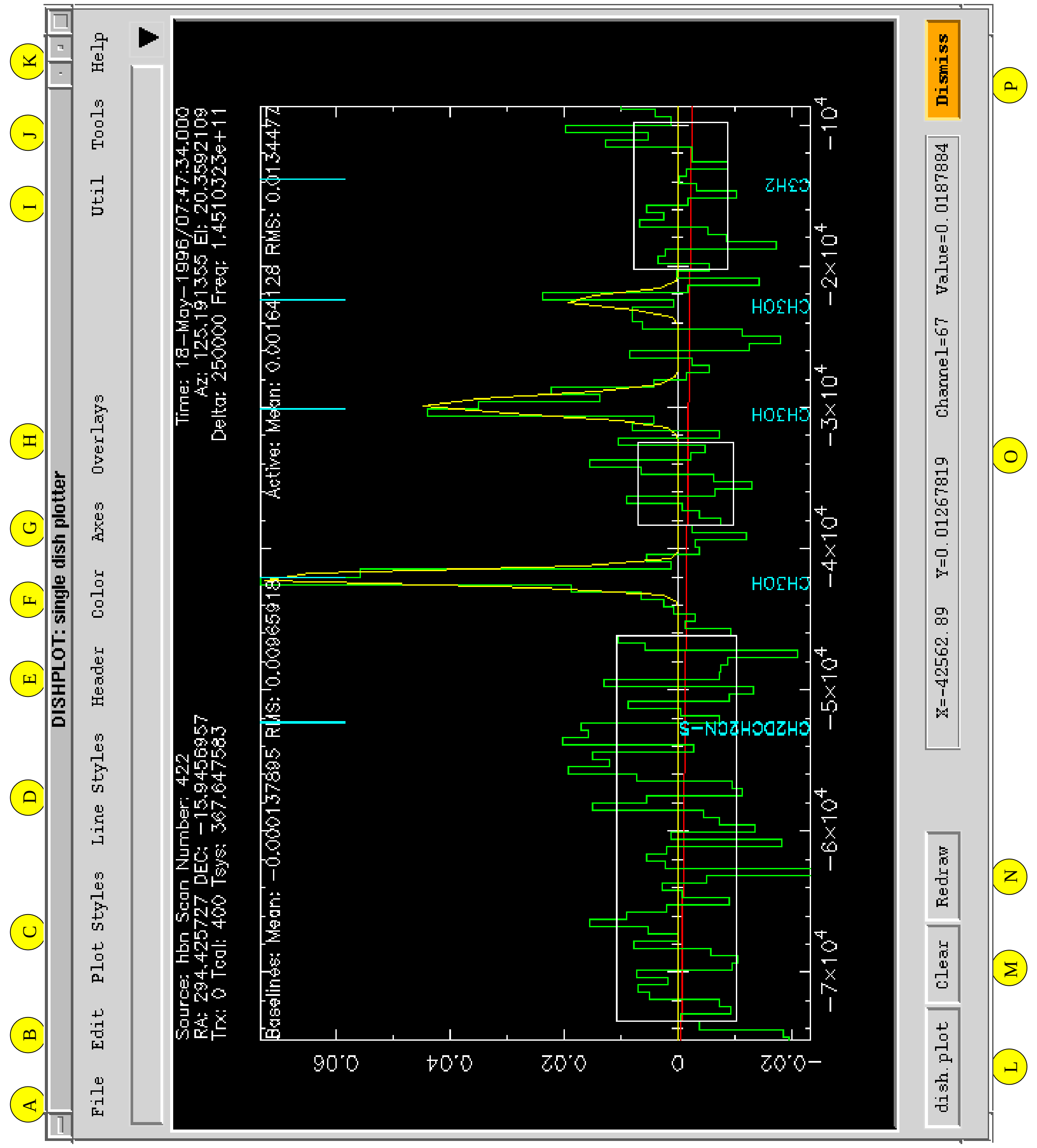

Figure 8: Dish Plotter Appearance. The lettered circles designate various plotter features. Displayed is a baselined spectrum (red) with the baseline fit to the original spectrum shown in green. The baseline regions are marked by the white boxes whose height indicates the rms in that region. The blue lines and labels on the top and bottom, respectively, indicate molecular lines from the Poynter and Pickett catalog within the frequency band of the plotted spectrum.

B: Edit Menu This is the standard pgplotter Edit menu. Add commands...: This allows additional PGPLOT commands to be added to the existing plot. For example, additional labeling or fiducial lines may be easily added. The Add commands brings up a frame with a list of all of the available PGPLOT commands. Selecting one brings up a brief explanation along with the various arguments required. An example frame is shown in Figure 9.

Change commands...: This allows already executed commands to be altered. A frame is brought up with the existing list of PGPLOT commands which produced the displayed frame. Again selecting a command brings up a small explanation along with the arguments used in its execution. Cut, Copy, and Paste features are available to duplicate and add additional commands. An example session is shown in Figure 10.

C: Plot Styles Menu This menu allows selection of four different plot styles: 1) Line (default), 2) Histogram, 3) Points and 4) Connected Points. This feature operates on the active (most recently displayed) plot in the frame.

D: Line Styles Menu This menu allows selection of five different line styles. This feature is only effective if the Line Plot Style is selected. The five line styles are: 1) Solid, 2) Dashed, 3) Dotted, 4) Dot-Dash, and 5) Dash-dot-dot-dot.

E: Header Menu The header menu offers four levels of header information on each plot: 1) None (the default), 2) Stats which will give the rms and mean of the displayed plot and, if available, the rms and mean of selected baseline regions on the plot, 3) Brief which gives only information on the Source name, scan number, date and time, and 4) Full, which renders all of the above information plus values for the position, altitude, azimuth, frequency, channel resolution. MORE

F: Color Menu The color menu currently offers only two options. Normal which is the default black background and Reverse, which is a white background.

G: Axes Menu Not yet implemented.

Four options for labeling the X-axis of a plot: 1) X-axis units: this will default to displaying whatever the default units of the plot are. In the case of the figure this is frequency, 2) Channel units: this will

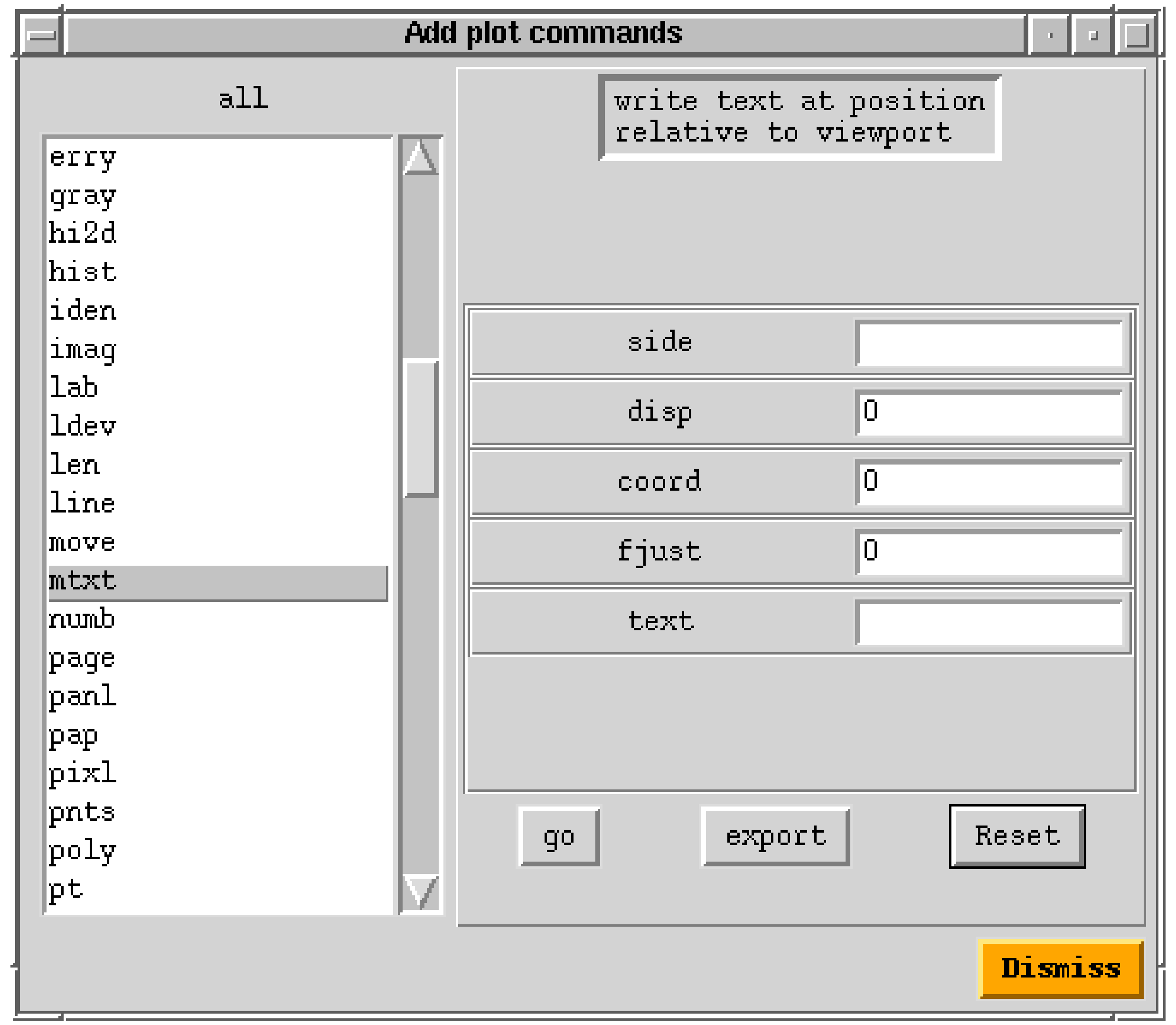

Figure 9: Add commands frame for Dish Plotter.

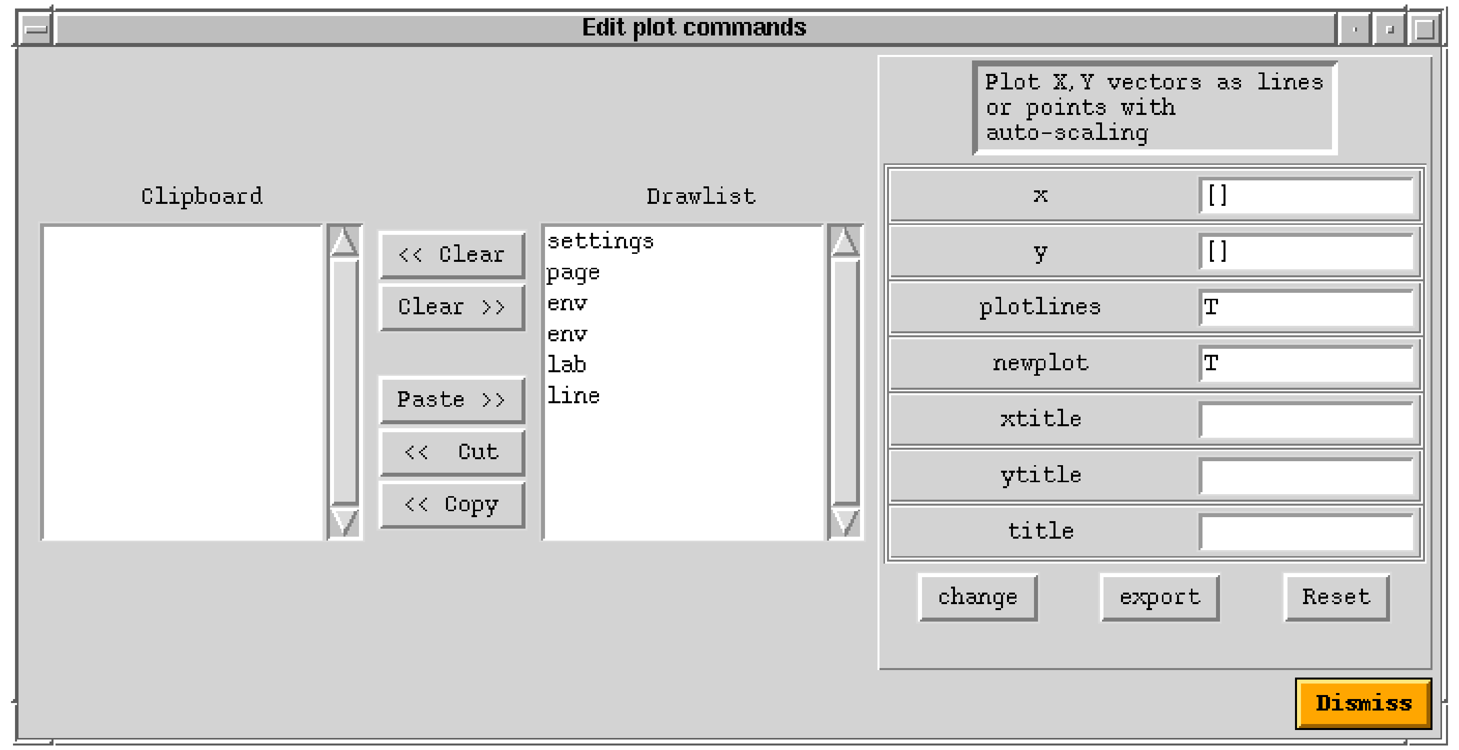

Figure 10: Change commands frame for Dish Plotter.

label the X-axis by channel number, 3) X-axis-Channel: this will label the bottom X-axis with the default units of the spectrum while the top will indicate the channel number, 4) Channel-X-axis: this is the reverse of three.

H: Overlays Menu The overlays menu is a simple toggle switch between On and Off. In the Off mode, all new plot results or selections will clear the screen before plotting the new result. The only exceptions to this are the Baseline operation in Show mode, which will overplot the calculated baseline over the spectrum, and multi-channel data which will plot as many channels as found on the same screen. In the On mode, all new selections and results will be overplotted; the axes will be scaled, if necessary, to accomodate all overlayed plots.

I: Util Menu Currently this menu offers only an unzoom feature. The unzoom handles the Dish Plotter zooming capability (the middle mouse button zoom), not the standard pgplotter zoom capability found in the Tools Menu. The middle mouse button zoom is required to appropriately handle overlayed plots; the pgplotter zoom will only zoom the active plot which is the last one plotted. This isn't generally a problem unless the axes for overlayed plots differ.

J: Tools Menu There are two tools offered. Zoom which is the standard pgplotter zoom capability. You may select either a boxed region or an X-axis or Y-axis zoom. Another button restores the plot to the full range.

The second tool is called LineID. This tool browses an aips++ table version of the Poynter and Pickett molecular line catalog, and plots any lines found that fall within the frequency range of the displayed plot. Line positions are plotted with short tick marks along the top of the plot while the corresponding molecular name tag is plotted vertically along the bottom.

K: Help Menu Standard pgplotter help menu. The options are: 1) PGPlotter: this will drive your netscape browser to the help section on the PGPlotter. 2) Reference Manual: this will drive the browser to the Reference Manual main page. 3) About Aips++...: This provides basic information about the AIPS++ version being used.

L: Plotfile name This is the filename that the Save option in the File menu will save to by default. The default plotfile name is dish.plot.

M: Clear Plotter Clears all currently displayed plots. Resets most plot parameters to their defaults. The color of displayed data resets to red, but line styles and plot styles previously selected will be retained.

N: Redraw Plotter The Redraw essentially re-executes all of the cached plot commands again. This becomes important if the plot has deviated from the original through, for example, command line additions.

O: Cursor Position Whenever the cursor is within the Dish Plotter frame, the cursor positions will be continuously read back. The coordinates shown are the X and Y position as defined by the X and Y axes' units. The channel number and Y-value of the cursor position are also displayed.

P: Dismiss Plotter Dismisses or closes the Dish Plotter frame. The frame will be automatically re-opened if an action occurs which produces a plottable result, i.e., if a spectrum is selected, a new Dish Plotter frame will be initialized to display the selection.

7 SDImager

The sdimager tool is an added utility to the DISH analysis environment which can perform simple imaging tasks using the DISH Plotter; it is designed to use SD working sets so it's use is exclusive to DISH.

sdimager may be summoned through either (1) the tool manager or from (2) the command line. From the tool manager, select "Packages, Modules, Tools, and Functions" bar, selecting the DISH package and the sdimager tool. This brings up a constructor gui which lets you name your version of the sdimager (the default is mysdimager). Hitting "Create", makes the tool and brings up a GUI showing the available functions. Selecting on a function brings up the entry fields with their default values (See Figure ref-fig:looktool). There is also an sdimager tool created upon entry into DISH, called dishsdi, which can be used immediately from the command line. For example:

```
- field_names(dishsdi)
*agent* type done lookmap spectramap contourmap otfmap clumpfind
- dishsdi.lookmap('w49n_bl1',-180,180,-180,180,0,0,410,F,'Test','Test2','Test3')
T
-
```

This creates the same output as seen in Figure reffig:looktool.

In addition, this object can be used to either create the sdimager constructor GUI or to go directly to one of the function GUIs. For example:

```
- tm.show('dishsdi') # use the toolmanager to call the constructor
T # GUI for the sdimager
- tm.show('dishsdi.lookmap')    # use the toolmanager to call up the GUI for
F # the lookmap function
-
```

Currently, there are three working functions within sdimager with plans for two more (an otfmap utility and a clumpfind utility for spectral line data cubes are both planned for late summer).

7.1 lookmap

The lookmap function plots either crosses or scan numbers at the locations where they were obtained relative to a selected reference scan. The inputs for the function are:

- working_set: This is the working set or sditerator from the DISH results manager that is selected for display.
- xmin,xmax,ymin,ymax: These are the spatial boundaries of the map in units of arcseconds offset from the reference scan position.
- lowscan,highscan: These set the scan numbers to be used within a chosen working set. If they are set equal, then the whole working set is used.
- refscan: This is the reference position used (center of the map) for measuring the offsets to other scans.
- plot_scan_number: This determines whether crosses are plotted at the scan positions (False) or scan numbers (True).
- xlabel,ylabel,mainlabel: The label for the x and y axes and the Title.

7.2 spectramap

The spectramap function plots spectra at the locations where they were obtained relative to a selected reference scan. The inputs for the function are:

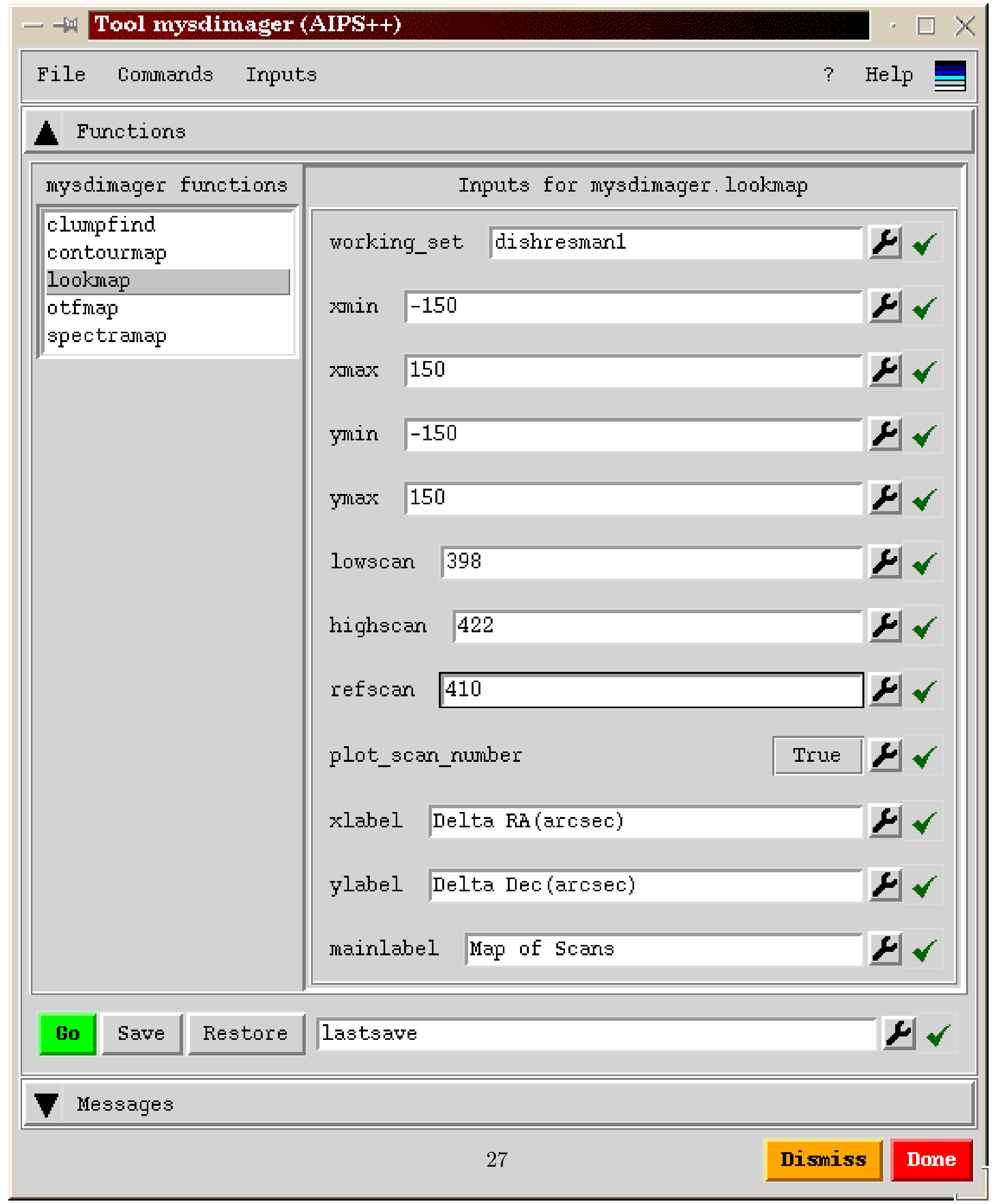

Figure 11: sdimager tool with associated functions and showing the input display for lookmap

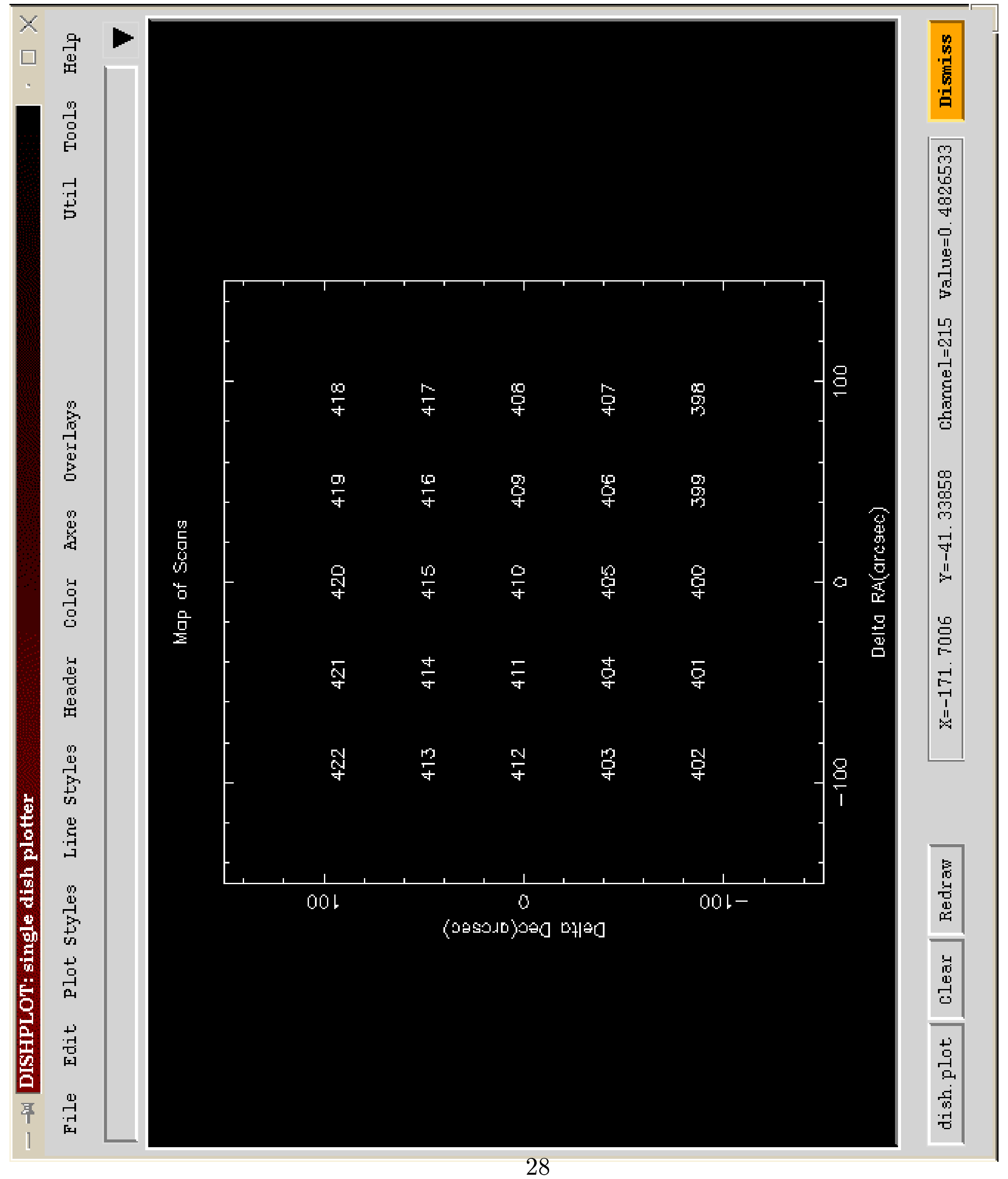

Figure 12: Example display of the lookmap function. It is displaying the relative spatial locations of different scans with respect to the reference scan, 410.

- working_set: This is the working set or sditerator from the DISH results manager that is selected for display.
- xmin,xmax,ymin,ymax: These are the spatial boundaries of the map in units of arcseconds offset from the reference scan position.
- lowscan,highscan: These set the scan numbers to be used within a chosen working set. If they are set equal, then the whole working set is used.
- refscan: This is the reference position used (center of the map) for measuring the offsets to other scans.
- xlabel,ylabel,mainlabel: The label for the x and y axes and the Title.

7.3 contourmap

The contourmap function plots contours of integrated intensity over a region specified by the map boundaries relative to a reference scan. *Currently this function expects regularly gridded maps and performs no interpolation. This should be remedied soon but it serves as a quick look facility.* In addition, contour map arrays may be written out as image files to be displayed in the more sophisticated AIPS++ viewer (not yet implemented). The inputs for the function are:

- working_set: This is the working set or sditerator from the DISH results manager that is selected for display.
- xmin,xmax,ymin,ymax: These are the spatial boundaries of the map in units of arcseconds offset from the reference scan position.
- lowscan,highscan: These set the scan numbers to be used within a chosen working set. If they are set equal, then the whole working set is used.
- refscan: This is the reference position used (center of the map) for measuring the offsets to other scans.
- plot_scan_number: This determines whether crosses are plotted at the scan positions (False) or scan numbers (True).
- xlabel,ylabel,mainlabel: The label for the x and y axes and the Title.

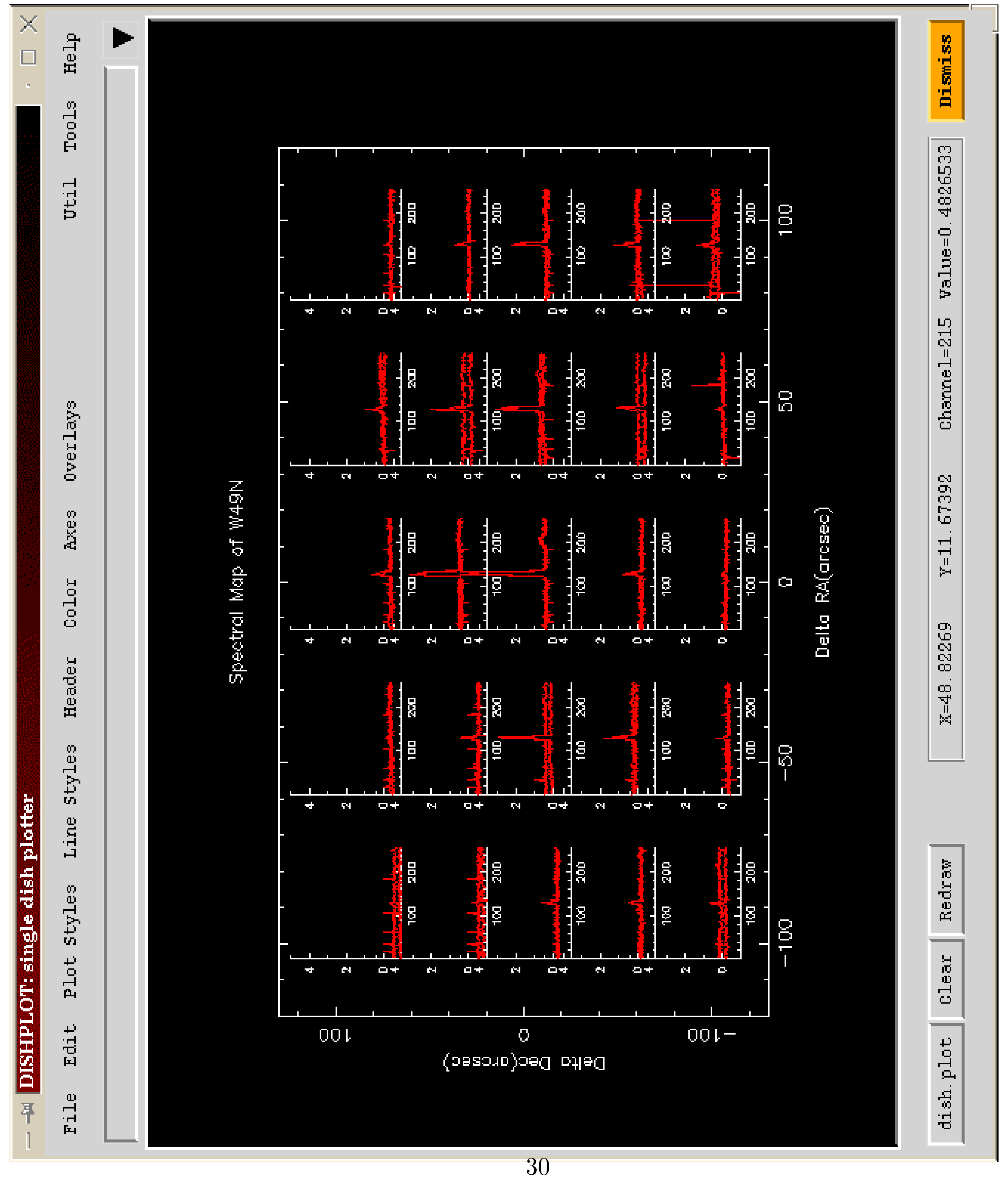

Figure 13: Example display of the spectramap function. It is displaying the spectra corresponding to the scans seen in Figure reffig:lookmap.

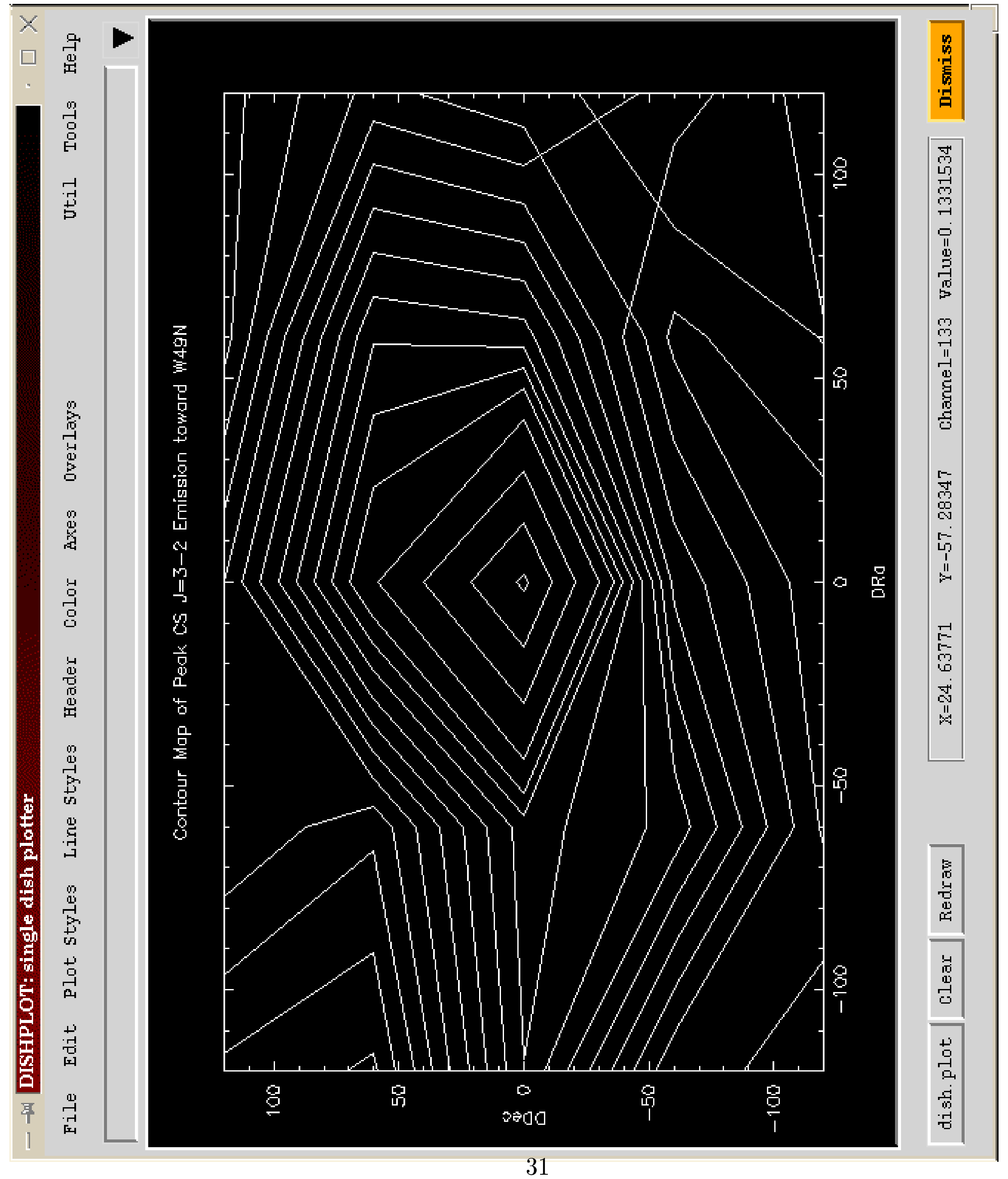

Figure 14: Example display of the contourmap function. It is displaying the peak temperature (center channel) in the spectra seen in Figure ref-fig:spectramap.

8 Recipes

8.1 Recipe 1: Reduce an ON/OFF Total Power scan

Goal: To reduce an on/off total power scan pair by extracting the 'on' and the 'off' source scans from an opened data set, constructing a difference scan from them, and inserting the result into the DISH results manager.

Assume: You have a data set named rawdata opened and available in the dish results manager. An 'on' scan is located at the first record in rawdata and an 'off' scan is located at the third record in rawdata.

```
AIPS++/Glish commands and results   Purpose and Background

rawdata.setlocation(1)              Move the rawdata pointer so that it
                                    points at the first record, where the
                                    'on' scan is located.

on:=rawdata.get()                   Get that scan and assign it to a variable
                                    named on. on is a glish record having a
                                    known structure. For example, the data and
                                    its description (axis type, value, etc.) is
                                    in a subrecord, data, and a subfield of
                                    that, arr, contains the data array.

rawdata.setlocation(3)              Move the pointer to point at the 'off'
                                    scan location.

off:=rawdata.get()                  Get it and assign it to 'off'.

result:=off;     Set result initially to 'off' so that it
    is a complete SDRECORD. Now adjust the
    the data array...

result.data.arr:=(on.data.arr -     Subtract the 'on' data array from the 'off'
       off.data.arr)/off.data.arr   data array and divide the result by the
                                    'off' data array. Additional operations to
                                    appropriately scale the data and adjust
                                    relevant header words would be done here.

dish.rm().add('result','Difference  Add this result to the DISH results
```

```
of rows 1 and 3',result,     manager. The final argument tells the
'SDRECORD')                  results manager that this is an SDRECORD
                             something the results manager knows how
                             to display and interact with.
```

8.2 Recipe 2: Perform a gaussian fit on a spectral line.

Currently, the gaussian fit operation is available only as a command line option. For this example, we use data obtained at the NRAO-12 meter telescope of methanol emission from Comet Hale-Bopp (the resultant figure of this operation may be seen in Figure 8.

Goal: To apply multiple gaussian fits to observed lines.

```
- field_names(dish)
done dismiss gui rm gaufit ops addop normalcursor busycursor savestate restorestate debug open showscript message plotter
- dish.gaufit()
How many gaussians to fit?
>> 3
please enter:
 height, width and center (separated by commas), for gaussian  1
>> 0.08,1100.,-42048.
please enter:
 height, width and center (separated by commas), for gaussian  2
>> 0.045,1500.,-29794.
please enter:
 height, width and center (separated by commas), for gaussian  3
>> 0.02,1100.,-22484.
[0.08 0.045 0.02]  [1100 1500 1100]  [-42048 -29794 -22484]
lengths of others are  3 3 3
Warning: fit has not converged # Note the lines in this case appear
# to be non-gaussian, hence the fit
# doesn't converge with our initial
# estimates. Errors in the fit are
# presented below.

fit result:
-----------
[converged=F, currentIteration=0, maxIter=15, criteria=0.00100000005, height=[0.08 0.045 0.02] , center=[-42048 -29794 -22484] , width=[1100 1500 1100] , height_error=[0.91675759 0.780412991 0.912533633] , center_error=[40.9339062 0.216167586 274.088285] , width_error=[-5922.02412 -10144.8242 -23016.1549] , chisq=0.0108240801]
```

```
mean of residuals:  -9.10550447e-05
rms of residuals:   0.00919536679
F
```

8.3 Recipe 3: Add a function to DISH (or fun with extensibility)

DISH is intrinsically enabled for extensibility. Currently, any files of the type dishops_xxxxx.gp (where xxxxx can be any string, e.g. dishops_cli.gp is used for the gaufit operation to indicate it is a command-line-interface operation), within the working directory will be automatically loaded. Functions within these files will be added to the those naturally available within dish. A simple template example is the following:

```
# dishops_template.gp -- template file for adding command line operations
#                        to dish
# Copyright (C) 1999,2000
# Associated Universities, Inc. Washington DC, USA.
#
# This library is free software; you can redistribute it and/or modify it
# under the terms of the GNU Library General Public License as published by
# the Free Software Foundation; either version 2 of the License, or (at your
# option) any later version.
#
# This library is distributed in the hope that it will be useful, but WITHOUT
# ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
# FITNESS FOR A PARTICULAR PURPOSE.  See the GNU Library General Public
# License for more details.
#
# You should have received a copy of the GNU Library General Public License
# along with this library; if not, write to the Free Software Foundation,
# Inc., 675 Massachusetts Ave, Cambridge, MA 02139, USA.
#
# Correspondence concerning AIPS++ should be addressed as follows:
#        Internet email: aips2-request@nrao.edu.
#        Postal address: AIPS++ Project Office
#                        National Radio Astronomy Observatory
#                        520 Edgemont Road
#                        Charlottesville, VA 22903-2475 USA
#
```

```
pragma include once;

dishops_template:=[=];

dishops_template.attach := function(ref public) {

        # now add whatever command line operation needed
        # for a specific example look at dishops_cli.gp which includes
        # the gaufit operation
# Add your functions here vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
        public.myfunction := function(input1='yes',input2=3.1415926) {
                print 'This is my function and it will do whatever I want';
                print 'My arguments are: ',input1,input2;
                print 'If these are yes and 3.1415925 then I used the defaults';
                print 'I can have as many arguments as needed.';
                return;
        }
# End of your function ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
# you can add as many functions as you desire

        return T; # to indicate the attachment went successfully

}
```

Running this operation within dish looks like the following:

```
- field_names(dish)
done dismiss gui rm ops addop normalcursor busycursor savestate restorestate debug open showscript message plotter gaufit myfunction
- dish.myfunction()
This is my function and it will do whatever I want
My arguments are:  yes 3.1415926
If these are yes and 3.1415926 then I used the defaults
I can have as many arguments as needed.
F
```

9 Development Plan

- First public release of DISH in AIPS++ October 1999. Respond to user feedback.
- Complete generalized command line interface.
- Read data directly from an AIPS++ MeasurementSet as well as from an AIPS++ Image Cube (which is itself a collection of spectra or SDRECORDs).
- Generalize data axis especially to accomodate non-linear x-axis.
- Long range calibration plan: single dish calibration should follow the same model as used for calibrating synthesis data within AIPS++This calibration will be accessible from within the DISH environment.
- Imaging: single dish imaging will use the same model as that used for synthesis imaging.

A The sdrecord

The SDRecord is the fundamental data atom of dish.

The current sdrecord structure was arrived at early in the design process. The next version of the sdrecord will more closely resemble individual rows of the `MeasurementSet`.

Unless otherwise stated, the units of all values in an SDRecord are SI units. Angles are expressed in degrees. This structure was arrived at before Measures and coordinates were fully supported in *Glish*. As a result, temporary fields in an SDRecord are used to hold sufficient information to indicate the coordinate system and reference frame where appropriate. All of the fields described below are required unless stated otherwise.

The value of a field is undefined in the following circumstances:

- The *Glish* function **is_nan(x)** returns **T** for a floating point value.
- The value is < 0 for an integer field.
- The string is empty.

For boolean fields, all values are considered valid. In addition, all values in the data array should be considered valid. The data array flag field should be used to indicate invalid values in the data array.

An SDRecord is a *Glish* record having the following structure:

data A *Glish* record which contains the data array and a description of the data array (axis type, values, increments, and units).

arr The data array. The dimensionality of the data array is `(nchan, nstokes)` where `nchan` is the number of channels and `nstokes` is the number of different stokes types found in this data. `nstokes` is either 1, 2 or 4.

desc A *Glish* record which describes the data array.

ctype, crval, crpix, and cdelt Follow standard FITS usage. They describe the first axis of the data array. An SDRecord can therefore only contain data where the first axis is linear in some coordinate system. This limitation will be removed in the next version of this structure.

units A string describing the units of the data array.

stokes A vector of strings describing the Stokes type of each element of the second axis of the data array. The possible Stokes types are:

I, Q, U, V, RR, RL, LR, LL, XX, XY, YX, YY, RX, RY, LX, LY, XR, XL YR, YL, PP, PQ, QP, QQ

For a typical single dish telescope, raw data will have `stokes = ("XX", "YY") or ("RR", "LL")`.

flag A boolean matrix with the shape as **arr**. When true, the corresponding value in **arr** has been flagged as bad and should not be used. When false, the corresponding value in **arr** is good.

weight A floating point matrix having the same shape as **arr**.

header A *Glish* record with the following fixed structure. The bulleted items are mearly separators to make this list easier to read, they are not members of **header**. The type of each field is indicated at the end of each description. Vector fields are indicated by giving the length of the vector in parenthesis after the field type.

- Identity

time The mid-point of the observation (UT). Double.

scan_number The scan ID number. Typically this is an identification number given to a chunk of data when the data is taken. Not all telescopes provide a scan ID number. Integer.

object The name of the thing being observed. String.

direction Where the telescope was pointed. Double(2).

direction_rate Any slew rate during the observation. Double(2).

- Times

ut_date The UT date corresponding to time. String.

ut The UT on ut_date corresponding to time. Double.

lst The LST corresponding to time. Double.

exposure The total amount of time actually spent collecting photons (e.g excludes blanking time, etc).

duration The elapsed (clock) time. Float.

- Observer

observer String

project String

- Backend

rest_frequency Double.

resolution The frequency resolution. This need not be the same as the channel spacing.

bandwidth The total bandwidth. Double.

obs_frequency The sky frequency at the reference channel. Double.

tcal The cal temperature. Float(nstokes).

trx The receiver temperature. Float(nstokes).

tsys The system temperature. Float(nstokes).

sigma The theoretical noise per channel. Float(nstokes).

- References

reference_position A reference position on the sky for position switched data. Double(2).

reference_frequency The frequency reference for frequency switched data. Double.

- Telescope

telescope A string identifying the telescope. String.

telescope_position The (x, y, z) position of the telescope in the ITRF (VLBI) coordinate system. Double(3).

telescope_diameter The diameter of the dish. More detailed telescope models will no doubt be useful but the parameters to describe these models are beyond the scope of the header record. If standard components could be identified, they would eventually be placed in the header. Until then, additional parameters to describe the shape of the telescope surface should be placed in ns_header. Float.

telescope_mount The mount type. Chosen from the following values: "alt-az", "equatorial", "X-Y", "orbiting", "bizarre".

azel The azimuth and elevation at time. Double(2).

vcorr The velocity of the telescope with respect to the velocity reference frame at time using the given velocity definition.

- Weather

pressure The atmopheric pressure. Float.

dewpoint The dew point. Float.

tambient The ambient temperature. Float.

wind_dir The wind direction. Float.

wind_speed The wind speed. Float.

- Coordinate system kludges

direction_coord A *Glish* Record describing the coordinate system for the **direction** field. It has the following structure.

name The coordinate name. String.

equinox The equinox for the coordinate. Double.

reference_coord A *Glish* Record describing the coordinate system for the **reference_position** field. It has the following structure.

name The coordinate name. String.

equinox The equinox for the coordinate. Double.

is_offset True if this is an offset position, else this is an absolute position. If this is an offset position then it is offset from the **reference_position**. Bool.

veldef The velocity definition (e.g. "radio", "optical"). String.

velref The velocity reference frame (e.g. "LSR", "HEL", ,etc.). String.

hist This vector of strings contains a history of the operations which resulted in this SDRecord. Each element is a valid *Glish* command. Taken as a whole, the hist field should be sufficient to regenerate the SDRecord from the original data source.

In addition to the fixed structure described above. An SDRecord may contain an optional `ns_header` field. This is a *Glish* record which can contain any additional information not found in the fixed portion of an SDRecord. This field will typically be used for telescope-dependent information.

The x-axis can be easily constructed from the `data` field by the following simple *Glish* function:

```
make_xaxis := function(ref data)
{
   x := data.crval;
   x +:= ([1:data::Shape[1]] - data.desc.crpix)*data.desc.cdelt;
   return x;
}
```

To summarize, in pseudo-*Glish* an SDRecord looks like this:

```
sdrecord := [data=[=], header=[=], hist=[=], ns_header=[=]];
sdrecord.data := [arr=[...], flag=[...], desc=[=], weight=[...]];
sdrecord.data.desc := [units=,stokes=,ctype=,crpix=,crval=,cdelt=];
        sdrecord.header := [time=,scan_number=,object=,etc.];
sdrecord.hist := ["# useful history","# as a vector","# etc. etc."];
        sdrecord.ns_header := [non-standard fields]

ns_header is optional
```

Additional information can be found in the Reference Manual section on the **get** function of the Sditerator `tool`. See also the **is_sdrecord** function.

B sditerator

See the Reference Manual for information on the Sditerator `tool` and the `is_sditerator` function.

C SDFITS

SDFITS is a convention for the use FITS binary tables to store single dish data. This appendix will eventually be turned into a full fledged note. At the present time, it is simply a summary of the various rules regarding the interpretation of keywords and columns which, taken together, constute the Single Dish FITS convention.

A FITS binary table is also a SDFITS binary table if it follows these rules.

C.1 EXTNAME keyword

```
EXTNAME = 'SINGLE DISH'
```

The EXTNAME keyword must have a value of "SINGLE DISH". A table having this value for EXTNAME is assumed to be fully compliant with this convention.

C.2 Virtual columns

Most keywords should be thought of as a column of the same name which has a constant value for all rows in the table.

There are a few exceptions to this rule. FITS keywords which are required to describe the structure of the table are not virtual columns. These include NAXIS, NAXIS1, NAXIS2, BITPIX, XTENSION, EXTNAME, EXTVER, EXTLEVEL, PCOUNT, GOUNT, TFIELDS, TFORM, TTYPE, TUNIT, TNULL, and TDISP. The TDIMnnn keyword is the only table description keyword which may be a column as described elsewhere in this appendix.

DATAMAX and DATAMIN are special keywords. When used as keywords it is most likely that the writer of that table meant them to reflect the range of all of the data found in that table and not to be used on a row-by-row sense. If the writer feels a need to supply row-by-row values for these two keywords, they should explicitly make them true columns.

When creating a table, a column may be made virtual if it's value can be represented as a single keyword (if it is constant over all rows) and if that constant value is NOT an IEEE special value (it has a standard ASCII representation) and the name of the column is 8 character or less.

If a column exists which has the same name as a keyword, the column values should be assumed to take precedence (writers should attempt to ensure that this confusing situation does not happen).

C.3 The DATA column and the DATA axes

The DATA column containsthe primary data array. It is required. Each row of this column contains an n-dimensional array of data. The axis descriptions are handled in a similar way to FITS images.

The TDIMnnn convention This is how the shape of the DATA array should be specified.

```
TDIMnnn = '(n1,n2,n3,n4...)'
```

- The product of all of the n1, n2, n3, etc., must be $\leq$ the width of the column as given by the TFORMnnn keyword.
- TDIMnnn may either be a keyword (constant shaped columns) or it may itself be a column (variable shaped columns).
- The data may be stored directly - unused elements in a particular row should contain the appropriate value to indicate an undefined value as described in the Binary Table Extension paper (i.e. NaN or use TNULL), Or the DATA column may use the P type of storage (storage on "the heap").

This convention may be used with other columns which do NOT use the same axis description (these will be site specific columns and readers are free to ignore these columns).

Axis description This description applies to any column which has `TMATXnnn = T`. It is assumed that `TMATXnnn = T` for the DATA column even when not present. The use of `TMATXnnn = F` for the DATA column is an error.

Axes descriptor keywords/columns include (not all need be present for each axis):

CTYPEn The type of physical coordinate on axis n.

CRVALn The value of the physical coordinate on axis n at the reference pixel.

CRPIXn The array location of the reference pixel along axis n. CRPIX may be a fractional pixel and/or be outside of the limits of the array. This descriptor is optional for degenerate axes.

CDELTn The increment in physical coordinates along axis n. This descriptor is optional for degenerate axes.

CROTAn The axis rotation

CUNITn The unit of coordinate values along this axis.

Note that these are the original FITS axes description keywords. It is anticipated that when an agreement is reached on the WCS convention that the appropriate keywords specified in that convention will be used to describe the axes here (e.g. CROTA is deprecated in favor of a more general mechanism).

Due to the virtual column convention, the following conflict may arrise:

```
TTYPE10 = 'CDELT4'
TFORM1  = '1E'
TUNIT10 = 'radian'
CUNIT4  = 'degree'
```

Writers should avoid this conflict. Readers encountering such a conflict should report this as an error in the FITS table. However, readers may also choose to proceed by arbitrarily resolving the conflict so that some attempt may be made within the anaysis package to deal with this error. It is recommended that the CUNITxxx specification take precedence over any TUNITxxx specification. In any case, this conflict and the readers resolution of the conflict (if at all) should be reported to the user.

The other standard FITS Image axis keywords may be used here. This will include all of the WCS convention keywords when they become accepted as part of the FITS standard.

Defined axis types (CTYPEnnn values) for this convention (others are allowed but their values may be ignored by readers) are:

frequency-like An 8 character string consisting of the 4 character axis type plus 4 characters describing the reference frame. An axis having this type is required.

axis types
- 'FREQ' - frequency (Hz)
- 'VELO' - velocity (m/s) (radio convention, unless overridden by use of the VELDEF SHARED keyword)
- 'FELO' - regularly gridded in frequency but expressed as velocity in the optical convention (m/s)

reference frames
- '-LSR' : Local Standard of Rest
- '-HEL' : heliocentric (barycentric)
- '-OBS' : the frame of rest of the observer/telescope (topocentric)
- 'LSRK' : LSR as a kinematical definition
- '-GEO' : Geocentric
- 'REST' : rest frequency
- '-GAL' : Galactocentric

Longitude-like 'RA','GLON','ELON' plus an optional WCS projection code (degrees). This axis is required.

Latitude-like 'DEC','GLAT,'ELAT' plus an optional WCS projection code (degrees). This axis is required.

'TIME' The time since DATE-OBS (seconds) If the TIMESYS keyword is present, that keyword defines the time system for this table, including this column, otherwise UTC is assumed. This axis is optional. If not present, a value of 0.0 should be assumed. This axis will often be absent if DATE-OBS contains a time as well as a date.

'STOKES' The Stokes parameter of the data.

- $1, 2, 3, 4 \Rightarrow$ I,Q,U,V
- $-1, -2, -3, -4 \Rightarrow$ RR, LL, RL, LR
- $-5, -6, -7, -8 \Rightarrow$ XX, YY, XY, YX

This axis is optional. If not present, a value of 1 (Stokes I) should be assumed.

'BEAM' Beam ID. This axis is optional. If not present, a value of 1 should be assumed.

'RECEIVER' receiver ID. This axis is optional. If not present, a value of 1 should be assumed.

C.4 CORE keywords and columns

These must be provided in all SDFITS tables. They are essential and common to all observations and telescopes. All single dish FITS readers and writers must acknowledge (write and properly interpret) all CORE keywords.

OBJECT A string value giving an object name.

TELESCOP A string value giving the telescope name.

BANDWID The total bandwidth of the backend in units of Hertz.

DATE-OBS A string giving the observation date and optionally the time at the start using the new FITS y2k convention. The TIMESYS keyword may be used to indicate the time system. UTC is assumed if TIMESYS is absent.

EXPOSURE The effective integration time in seconds.

TSYS The system temperature in Kelvin.

C.5 SHARED keywords and columns

These have agreeded definitions and interpretions however their presense is obtional. These are largely common to all observations and telescopes but not essential. These may be ignored by a single dish FITS reader.

OBSERVER A string giving the observer's name.

OBSID A string describing the observation.

PROJID A string describing the project.

SCAN A scan ID number. Typically this is an identification number given to a chunk of data when the data is taken. Not all telescopes provide a scan ID number.

OBSMODE The type of data and observing mode (8 characters total). The type (LINE, CONT, PULS, etc) + the mode (PSSW, FQSW, BMSQ, PLSQ, LDSW, TLPW, etc). These rules do NOT define these observing modes. Writers are strongly encouraged to use the FITS comments to document these modes.

MOLECULE A string used as a line identifier (with TRANSITI).

TRANSITI A string used as a line identifier (with MOLECULE.)

TEMPSCAL A string describing the scaling applied to reach the output intensity scale ("TB","TA","TA*","TR","TR*").

FRONTEND A string giving the name of the front end device.

BACKEND A string giving the name of the back end device.

TCAL The calibration temp (K).

THOT The hot load temp (K).

TCOLD The cold load temp (K).

TRX The receiver temp (K).

FREQRES The frequency resolution in Hz. This may differ from the channel spacing.

TIMESYS The time system which applies to all time columns and keywords (see the y2k FITS DATE agreement).

VELDEF The velocity definition and frame (8 characters). The first 4 characters describe the velicity definition. Possible definitions include:

RADI radio

OPTI optical

RELA relativistic

The second 4 characters describe the reference frame (e.g. "-LSR", "-HEL", "-OBS"). If the frequency-like axis gives a frame, then the frame in VELDEF only applies to any velocities given as columns or keywords (virtual columns).

VFRAME The radial velocity of the reference frame wrt the observer. $V_frame - V_telescope$.

RVSYS The radial velocity, $V_source - V_telescope$.

OBSFREQ The observed frequency (Hz) at the reference pixel of the frequency-like axis.

IMAGFREQ The image sideband freq (Hz) corresp. to OBSFREQ.

LST The LST (seconds) at the start of scan.

AZIMUTH The azimuth at TIME (deg) (if the TIME axis is non-degenerate, then this is the azimuth at the TIME of the first pixel on the TIME axis.

ELEVATIO The elevation at TIME (deg) (same caveat as for AZIMUTH)

TAU The opacity at OBSFREQ.

TAUIMAGE The opacity at IMAGFREQ.

TAUZENIT The opacity per unit air mass.

HUMIDITY The relative humidity (fraction, 0..1).

TAMBIENT The ambient temp (K).

PRESSURE The atmospheric pressure (mm Hg).

DEWPOINT The dew point (K).

WINDSPEE The wind speed (m/s).

WINDDIRE The wind direction (deg. west of north).

BEAMEFF The main-beam efficiency.

APEREFF The antenna aperture efficiency.

ETAL The rear spillover and scattering efficiency.

ETAFSS The forward spillover and scattering efficiency.

ANTGAIN K per Jy.

BMAJ The major main-beam FWHM (deg).

BMIN The minor main-beam FWHM (deg).

BPA The beam position angle (degrees east of north).

SITELONG The site longitude (deg).

SITELAT The site latitude (deg).

SITEELEV The site elevation (m).

RESTFREQ The rest frequency (Hz).

C.6 Other columns

Any additional columns not explicitly mentioned here can be added to an SDFITS table, although, obviously, most readers will not be able to properly interpret those columns.

C.7 Multiple SDFITS tables in a single file

Any number of FITS binary tables can be attached to the same FITS table. This convention doesn't have anything to say about how the tables in such a file might be related. This strategy, attaching multiple SDFITS binary tables to a single FITS file, is one possible strategy for storing variable length DATA. DATA with similar sizes could be stored in separate tables, minimizing the amount of padding required in any one table without resorting to using the table heap convention to store variable length arrays. SDFITS readers should be capable of appending the contents of several SDFITS tables to a single result.