GILDAS
Grenoble Image and Line Data Analysis System
Users Manual and Task Reference

a GILDAS working group software

March 17, 1999

Version 2.1

The GILDAS working group is a collaborative project of the Observatoire de Grenoble (1,3) and IRAM (2), and comprises: G. Buisson[1], L. Desbats[1], G. Duvert[1], T. Forveille[1], R. Gras[3], S. Guilloteau[1,2], R. Lucas[1,2], and P. Valiron[1].

(1) Laboratoire d'Astrophysique
Observatoire de Grenoble
BP 53 X
414 Rue de la Piscine
F-38402 Saint Martin d'Hères CEDEX

(2) Institut de Radio Astronomie Millimétrique
300 Rue de la Piscine
F-38406 Saint Martin d'Hères

(3) CEPHAG
Observatoire de Grenoble
F-38402 Saint Martin d'Hères CEDEX

Contributions from and invaluable discussions with J. Cernicharo, P. Begou[3], S. Delahaye[3], A. Dutrey[1,2], C. Kahane[1], P. Monger, J.L. Monin[1], R. Neri[2] and all **GILDAS** users are gratefully acknowledged.

Contents

1 Introduction

The present user manual assumes the user is already familiar with the **SIC** command monitor, and also with the **GreG** graphic program. If not, read the documentation entitled "An Introduction to **GILDAS**" to get started.

1.1 The Data Format

GILDAS has two slightly different data types : Images and Tables. Images are data sets of up to 4 dimensions, with a header specifying the World coordinate system, the type of projection used, spectroscopic information, etc... Tables are essentially like 2-D images (a 2-D image can be treated as a Table in fact), but the only relevant information in this case is the number of lines and columns in the table. The header is described in more details in the programmer's guide.

GILDAS was originally based on the mapping memory concept, in which an image, resident on the disk, is considered as part of the virtual memory space of the user. Memory mapping has the great advantage of separating the Algorithms from the Input/Output system. The portable Unix version, though no longer using memory mapping, still preserves a complete separation between I/O statements and Algorithms. Algorithms are just standard subroutines operating on arrays storing the images.

1.2 Drawbacks

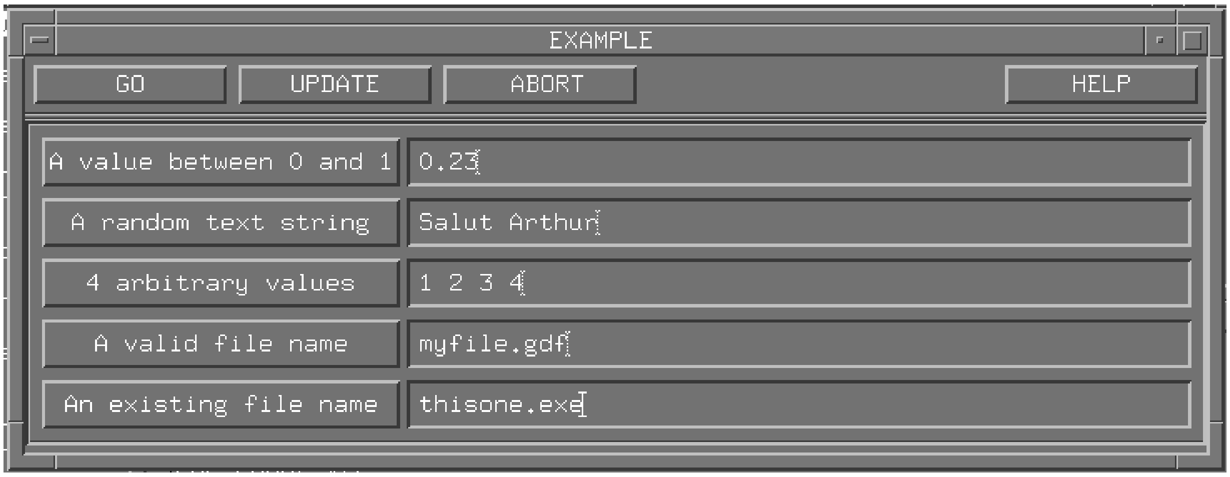
"Nobody is perfect"
(attributed to the nameless God)

GILDAS was designed to be fast and use little disk space. The consequence is that it is not as comprehensive as larger packages. In particular, it does not handle any "history" like AIPS does. You should then take care of what a particular image is really. All vital information is in principle correctly transmitted : axis types, coordinate conversion formula, projection information. Extrema may need to be recomputed, and the "Spectroscopy" section is still experimental.

Other drawbacks comes from the virtual memory concept itself. The maximum virtual memory a process can have is often limited by the operating system configuration. With modern computers, a reasonable limit of say typically 32 Mbytes corresponds to data cubes of 128 by 128 by 128 (as you typically access several data cubes at the same time...). Few instruments are capable of producing larger data sets however, and large computers would be required to handle these data in any case. These limitations can be partly removed by proper programming of the applications (i.e. reading subsets whenever possible).

GILDAS does not clean up things as for example AIPS does when you exit. Command files for the tasks are in principle deleted after task completion. But it is up to the user to delete scratch files and log files which are no longer needed. However **GILDAS** never creates data or work files without having requested a file name, so you know which files have been created.

Finally, **GILDAS** contains an increasing number of algorithms. It is always assumed that the user understands the basics of image processing when using these algorithms. They may not apply to your special cases. Although they are usually tested, some of them may still be in the development phase. A message in the `HELP` or in the parameter file will signal these programs.



2 Running Tasks: the VECTOR\ Language

This section contains the minimum information required to use the **GILDAS** image processing tasks.

The usual way to run the tasks is to activate the VECTOR or GRAPHIC programs and use commands `RUN` and `SUBMIT`. Both commands are very similar. The `RUN` command will execute the task as a detached process, and the `SUBMIT` command in a batch queue named `GILDAS_BATCH`. The maximum number of detached processes a single user can have in **GILDAS** is at most four (this number may in addition be limited by your system manager), and a single task cannot be active in two different detached processes of the same user.

The `VECTOR\` language contains the following commands :

```
EXPLAIN [Task]  : Types help about GILDAS tasks
RUN Program     : Activates a GILDAS task in a detached process
SUBMIT Program  : Submit a GILDAS task to GILDAS_BATCH queue
SPY [Task]      : Look at the status of one or all GILDAS tasks.
WAIT [Task]     : Wait until completion of one or all GILDAS tasks.
```

2.1 Window Mode

The window mode is the default mode on X-Window systems with Motif interface. Let us assume in the following example we want to execute a task named `EXAMPLE`. To activate `EXAMPLE`, the user will type

```
    VECTOR> RUN EXAMPLE
or
    VECTOR> SUBMIT EXAMPLE
```

A separate input window is created: The user can then modify any of the parameters by clicking in the dialog areas. Help can be obtained by clicking on the HELP button, or on any parameter description.

Since **SIC** is used, parameter values can be variables or arithmetic expressions (e.g. `2*PI+EXP(X[3])` is a perfectly valid value for a real, provided the array `X[n]` with `n>3` has been previously defined).

Once all parameters are defined, the task can be launched by clicking the OK button, or aborted using the ABORT button. Parameter values are checked, and if all parameters are valid, the task is executed (or submitted). If one parameter is invalid, the `RUN` or `SUBMIT` command sends back a message :
`E-RUN, Missing GO command`
and returns an error.

2.2 Query Mode

When no window-mode is available, the user is prompted for the parameters. In this example, the dialog will be

```
A value between 0 and 1
REAL        A$           0.1 <CR>
Any character string
CHARACTER   CHAIN$       ABCD <CR>
4 Real values
REAL        ARRAY$[4]    1 2 3 4 <CR>
Output file name
FILE        FILE$        TESTFILE.DAT <CR>
```

The prompting method is always the same: an explanatory first line indicating the meaning of the parameter, and a second line in the following format:

```
TYPE NAME[Dimensions]
```

where

- `TYPE` indicates the type of parameter (`CHARACTER, FILE, INTEGER, LOGICAL, REAL`). A parameter of type `FILE` is a character string containing a valid file name. Because of the use of memory mapping, access to files on a remote DECnet node is forbidden.
- `NAME` is the parameter name
- `[Dimensions]` are the parameter dimensions, in case it is an array. Only `REAL` and `INTEGER` parameters may be arrays.

Query mode is also used for missing parameters in Window-mode.

2.3 EDIT Mode

Commands `RUN` and `SUBMIT` execute two **SIC** command procedures, the *Initialization File* `Task.INIT`, which defines all parameters needed for `EXAMPLE`, and the *Checker File* `Task.CHECK`, which verifies that all parameters are valid. In the example above, the `EXAMPLE.INIT` file is

```
TASK\REAL "A value between 0 and 1" A$
TASK\CHARACTER "Any character string" CHAIN$
TASK\REAL "4 Real values" ARRAY$[4]
TASK\FILE "Output file name" FILE$
TASK\GO
```

This is a standard procedure containing commands of a **SIC** language named TASK\. Commands from this language are used to define the parameters required by the task, and cannot be called interactively. The command syntax is always the same :

```
TASK\Command "Prompt String" Parameter$[Dimensions] [Value [...]]
```

where

- `Command` indicates the type of parameter (`CHARACTER, FILE, INTEGER, LOGICAL, REAL`). A parameter of type `FILE` is a character string containing a valid file name. Because of the use of memory mapping, access to files on a remote DECnet node is usually impossible.
- `"Prompt String"` is a character string used as a prompt to ask for the parameter value(s)
- `Parameter$` is the parameter name
- `[Dimensions]` are the parameter dimensions, in case it is an array. Only `REAL` and `INTEGER` parameters may be arrays.
- `Value(s)` are the parameter values, an array requiring as many values as array elements.

Once all parameters have been assigned values, RUN and SUBMIT commands execute the GILDAS_RUN:EXAMPLE.CHECK file, writing the current parameter values in an auxiliary file which will be used by the task EXAMPLE. If a parameter is incorrect, an error is returned, and the task EXAMPLE not executed.

Instead of supplying the parameters in a query mode, the user can use a text editor to edit the .INIT file using command

```
VECTOR> RUN EXAMPLE/EDIT
```

or

```
VECTOR> SUBMIT EXAMPLE/EDIT
```

The parameter values can then be typed after the parameter names in the EXAMPLE.INIT file, using **SIC** continuation marks ("-" as the last character of a line) if needed for long command lines. EXAMPLE.INIT will be executed after exiting the editor. If a parameter value is missing, the user will nevertheless be prompted for it after exiting the editor.

The text editor called is user defined by the command SIC\SIC EDITOR or the logical name GAG_EDIT.

2.4 Specifying the .INIT File

By default, in Query mode RUN and SUBMIT commands use the .INIT file located in GILDAS_RUN:. In EDIT mode, the .INIT file located in the current default directory is used if it exists. To override this default behaviour, you can specify any .INIT file as the second argument to commands RUN and SUBMIT.

2.5 Errors and Aborting

If an error occurs in the .INIT or .CHECK procedure, the erroneous command will be returned to the user, and the procedure execution is interrupted by a pause. You can then correct the error, execute the command, and type C or CONTINUE to resume the procedure execution. Or you can type QUIT (as in any **SIC** procedure indeed) to abort the execution, until the RUN or SUBMIT command returns an error.

You may also want to abort a RUN or SUBMIT command while you are in the editor: typing QUIT instead of EXIT to end the editing will do it.

2.6 Log Files

A log file is created by the RUN command in your GAG_LOG: directory with the task name as file name and the extension .GILDAS; this log is printed by the SUBMIT command. If the user is still running the main program (VECTOR or GRAPHIC, etc...) when a task completes, he (or she) is warned of the completion with the return status. Log files are not purged automatically, so that you should take care of that. They are intended essentially as a debugging aid if something goes wrong, but you can print them as archive of your data processing.

A command file is created in your GAG_LOG: directory to run or submit the programs. It is in principle deleted at task completion.

2.7 Synchronizing Jobs

The batch queue GILDAS_BATCH should have a job limit of 1, so that all tasks submitted by command SUBMIT execute in sequence. There may even be intervening jobs from other users.

This is not the case for tasks activated by command RUN, which starts execution immediately. Command WAIT can be used to place the activating program in "hibernation" until a specified task (by default the last one) has completed.

Command SPY can also be used to monitor the execution of tasks activated by command RUN.

2.8 Obtaining Explanations: EXPLAIN Command

There are three ways to obtain help about **GILDAS** tasks :

- Using command EXPLAIN: EXPLAIN Atask gives general explanations about the **GILDAS** task Atask, EXPLAIN Atask Apar gives more details about the parameter Apar of the task Atask.
- Using command HELP. HELP works as EXPLAIN, except when there is ambiguity between a command name and a task name. In such a case, to obtain the help for the task name, use HELP GILDAS_RUN:Atask.
- In EDIT mode of commands RUN and SUBMIT, using key GOLD ? allows to obtain help on the current task.
- In Query mode, answering ? to a prompt returns help on the current parameter.

3 VECTOR Language Internal Help

3.1 EXPLAIN

```
    [VECTOR\]EXPLAIN [Task [Parameter]]

Gives explanation about  a  GILDAS  task.  If  Parameter  is  specified,
EXPLAIN will give help about the parameter. Parameter = * can be used to
list help about all parameters of  the  specified  task.  Note  that  if
Parameter is absent, EXPLAIN accesses the topic "Summary".

The GILDAS tasks can only be activated from the VECTOR\  language,  with
commands  RUN  or  SUBMIT.  The VECTOR\ language is included in programs
VECTOR, GRAPHIC and other reduction programs such as MAPPING.

From within RUN and SUBMIT commands, you can get help upon  the  current
Task  by  typing  GOLD ? in the editor, or by answering ? to a prompt in
non editing mode.
```

3.2 RUN

```
    [VECTOR\]RUN   Task_Name   [Parameter_File]   [/EDIT]    [/NOWINDOW]
[/WINDOW]

Execute a GILDAS  task  as  a  detached  process.  If  no  directory  is
specified  in  the  task  name,  the  Task  is  assumed  to  be  in  the
GILDAS_LOCAL: or GILDAS_RUN: area. The input parameters  are  read  from
the file Parameter_File, which is a SIC procedure with commands from the
TASK\ language. A * can be used instead to specify a parameter  file  of
name Task_name.INIT in the current directory.

The parameter file can be prepared in "Window-mode",  or  using  a  text
editor.
        In Window-mode, activated implicitely in the RUN_WINDOW variable
is  YES  (default  on  X-Window  systems), or explicitely if the /WINDOW
option is specified, a panel appears to enter all  parameters.  Help  if
available by clicking on the prompt string for each parameter, or on the
HELP button. Once all parameters have  been  adequately  specified,  the
task can be activated by clicking OK, or aborted by clicking ABORT.
```

The text-editor mode is activated using the /EDIT option. The parameter file of default name Task_name.INIT is edited before submission, taking a template in GILDAS_RUN: area if no version of this file already exist. If this template file does not exists, it may be that the Task you want to run doesnot exists either, or is not yet debugged at all.

Once the parameter file has been prepared in Edit-mode or Window-mode, the RUN command will prompt you for all missing parameters in the answering ? to the prompt.

The RUN command checks that the Task exists, and only GILDAS Tasks can be submitted in this way. A second SIC procedure is executed before task submission to check the validity of input parameters. If any parameter is invalid, an error is returned and the Task not submitted.

The input file which is created by the RUN command is located in the GAG_SCRATCH: directory and may be deleted after Task execution. The output of the Task is in the file GAG_LOG:Task_Name.GILDAS, which may be listed or printed later on. Task execution may be synchronous (the main program waiting for task completion) or asynchronous (control returns to the main program immediately). If the Task terminates before you exit from the activating program, a termination message will be typed on the terminal, giving the termination status.

Use the EXPLAIN command for help on available Tasks.

Task may execute on a remote node rather than on the local machine. The node name is controlled by logical name GILDAS_NODE. If GILDAS_NODE = LOCAL, local execution is performed. If not, GILDAS_NODE must be the node name of the computer on which execution will be performed. No synchronisation is offered for remote execution.

3.3 SPY

[VECTOR\]SPY [Task_Name]

Displays the status of all active GILDAS tasks, or list the last output from the specified task.

3.4 SUBMIT

[VECTOR\]SUBMIT Task_Name [Parameter_File] [/EDIT] [/NOWINDOW] [/WINDOW]

The SUBMIT command is similar to the RUN command, except that the Task is submitted to a batch queue (named GILDAS_BATCH) instead of being executed as a detached process. See RUN command for details.

Use the EXPLAIN command for a documentation on available Tasks.

3.5 TRANSPOSE

```
[VECTOR\]TRANSPOSE Input Output Order
```

```
This command takes an input 3-D data cube to produce an output
transposed 3-D cube according to the transposition order specified by
Order (312, 213, etc...). For example:
    VECTOR\TRANSPOSE TEST.VLM TEST.LMV 231
```

3.6 WAIT

```
[VECTOR\]WAIT [Task_name]
```

```
Place the current program in a wait state until the specified Task
(started by command RUN) terminates. By default, the last started Task
is used. WAIT * waits until all tasks complete. Waiting can be
interrupted by pressing ^C.

It may be used to synchronize complex procedures where the result of
some task is used as input by others. For batch jobs sent through
command SUBMIT, this synchronization is normally ensured by the job
limit of the GILDAS_BATCH batch queue.
```

4 Displaying Images: the GRAPHIC\ Language

All Tasks are (small) non interactive programs which only require initial setup (parameters). When interactive processing is required (visualization, pixel value interrogation, etc...), one uses the GRAPHIC program.

GRAPHIC includes the VECTOR program, and hence are able to run or submit **GILDAS** tasks.

4.1 The GRAPHIC Program

GRAPHIC is essentially VECTOR and **GreG**combined in a single program. In addition, it includes access to a "current image" and the following commands to handle this image and catalogs:

```
ASTROMETRIC     : Plots astrometric star positions (for finding charts)
HEADER          : Give the Header of the current Image.
IMAGE [Filename]: Read the gildas image Filename.
IRAS Band       : Select and plot IRAS sources of a given band
KILL            : Kills pixels in the current image
PSC_IRAS        : Get characteristics of IRAS point source
REGRESSION [Val]: Computes regression lines
SPECTRUM        : Extract a spectrum from the current image.
```

The GRAPHIC program uses **GreG** to display images, either in bitmaps or contour levels, with full user control over the image boundaries, user coordinate limits, etc...

Three specific commands deal with **GILDAS** images:

- `IMAGE`
  Read an image or a subset of an image in the "Regular Grid" area of **GreG**˙This command allows to have access to individual planes of a 3-D or 4-D data cubes for contouring, perspective plotting, pixel value interrogation by using the **GreG** `GREG2\` language. All relevant information is transmitted to **GreG**: user coordinate values, coordinate system, projection used if any, blanking values.

- `HEADER`
  Displays the header of the currently mapped image, or a specified image.
- `SPECTRUM`
  Extracts a spectrum from the currently mapped image by putting the values of the specified column in **GreG** buffer Y and the corresponding coordinates, computed from the header information, in **GreG** buffer X.
  (i.e. `X(i) = (i-Xref)*Xinc + Xval for i=1 to NX`)
- `KILL` delete bad pixels, provided the image has been opened with Read-Write access.

By default, GRAPHIC opens the input image in `ReadOnly`, so that it will never alter them *unless specifically requested.* To do so image should be accessed with ReadWrite access (option `/WRITE` of command `IMAGE`). Even in this case, commands in the `GREG2\` language never modify the image mapped by GRAPHIC; only commands from the language `GRAPHIC\` can do so.

4.2 Image Header Editing

4.2.1 Header Format

The image header may be displayed using command `HEADER`. This command produces a screen with the following aspect (except for the numbers in the last two columns which are added here for reference purpose):

```
File :         /users/me/work.lmv                                              1
Size        Reference Pixel          Value                 Increment
    51 -0.5000000000000000      -21.00000000000000       1.000000000000000     2
    53 -0.5000000000000000      -25.00000000000000       1.000000000000000     2
   113   57.50000000000000      -15.60490417480469       1.300384521484375     2
     1  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00 2
Blanking value and tolerance      -1000.0000      0.10000000                   3

Source name         VHYA                                                       4
Map unit            K                                                          5
Axis type           RA           DEC          VELOCITY                         6
Coordinate system   EQUATORIAL                                                 7
Right Ascension     10:49:11.300        Declination        -20:59:05.00        8
Lii       -91.03457538867391            Bii       33.60127769133289            9
Epoch              1950.0000                                                  10
Projection type     RADIO               Angle    0.0000000000000000E+00       11
Axis 1     A0       10:49:11.300        Axis 2     D0      -20:59:05.00       12
Minimum                                                                       13
Maximum
Axis 3 Line Name    12CO(2-1)           Rest Frequency   230537.9900000000    14
Resolution in Velocity    1.3004038     in Frequency    -1.000000000000000    15
Offset in Velocity       -15.600000     Image Frequency  0.0                  16
```

The lines contain:

1. Image filename
2. Number of pixels, reference pixel, value, increment for each axis
3. Blanking value and tolerance
4. Source name for bookkeeping
5. Units of map

6. Type of each axis
7. Type of coordinate system
8. Source coordinates in Equatorial coordinate system
9. Source coordinates in Galactic coordinate system
10. Epoch of equatorial system
11. Type of projection, and angle of the projection axis with respect to north
12. Projected axis numbers, and coordinates of projection center
13. Value and pixel coordinates for the minima and maxima of the image.
14. Frequency axis number, Line name (for bookkeeping) and frequency of the reference pixel.
15. Velocity and frequency resolution
16. Velocity and image frequency of the reference pixel.

Each item can be modified, except the numbers of pixels of course.

4.2.2 Variables

The user can modify directly the header variables whose names are given below. The file is modified when typing the command `HEADER /UPDATE`.

```
G_VELOFF        REAL               Velocity at reference pixel
G_VELRES        REAL               Velocity increment
G_RESTFRE       DOUBLE             Rest frequency at reference pixel
G_FREQOFF       DOUBLE             Image frequency at reference pixel
G_FREQRES       DOUBLE             Frequency increment
G_LINE          CHARACTER *12      Line name
G_BII           DOUBLE             Galactic latitude
G_LII           DOUBLE             Galactic longitude
G_DEC           DOUBLE             Declination
G_RA            DOUBLE             Right ascension
G_EPOCH         REAL               Epoch of equatorial coordinates
G_SOURCE        CHARACTER *12      Source name
G_COORD         CHARACTER *12      Coordinate system
G_UNIT          CHARACTER *12      Map units
G_UNIT4         CHARACTER *12      4th axis type
G_UNIT3         CHARACTER *12      3rd axis type
G_UNIT2         CHARACTER *12      2nd axis type
G_UNIT1         CHARACTER *12      1st axis type
G_EXTREMA       INTEGER   [ 2, 4]  Position of extremax (min, max)
G_MIN           REAL               Value of minimum
G_MAX           REAL               Value of maximum
G_BLANK         REAL      [ 2]     Blanking and tolerance values
G_CONVERT       DOUBLE    [ 3, 4]  Pixel to user coordinate conversion
G_DIM           INTEGER   [ 4]     Dimensions
G_NDIM          INTEGER            Number of dimensions
```

The pixel to user coordinate conversion for a given `Axis` is

```
Val(Pixel) = (Pixel-G_CONVERT[1,Axis])*G_CONVERT[3,Axis]+G_CONVERT[2,Axis]
```

and the user coordinate to pixel

```
Pixel(Val) = NINT(Val-G_CONVERT[2,Axis])/G_CONVERT[3,Axis]+G_CONVERT[1,Axis])
```

4.3 Astronomical Processing

Finally, a few additional commands are used for dedicated processing :

- IRAS plots on an astronomical map the IRAS point sources.
- PSC_IRAS locates and lists characteristic of IRAS point sources
- ASTROMETRIC finds and plots on an astronomical map stars from an astrometric star catalog.

IRAS, PSC_IRAS and ASTROMETRIC work only if if the corresponding catalogs are kept on line. The IRAS and PSC_IRAS commands require one big table (14 Kblocks) to be present somewhere on the disk under the logical name PSC_COMPACT, and possibly a big direct access file (40 Kblocks) under the logical name PSC_IRAS_EQU.

In addition, a separate task working either on disk file or on tape can be used, the task PSC_IRAS.

5 GRAPHIC Language Internal Help

5.1 ASTROMETRIC

```
    ASTROMETRIC [RAmin RAmax DECmin DECmax]  [/MAGNITUDE  Mlim]  [/EPOCH
date] [/OUTPUT File] [/EQUINOX date]

Searches and plots all astrometric stars of  magnitude  less  than  Mlim
(brighter than Mlim) in the (RA,DEC) box specified by RAmin RAmax DECmin
DECmax (if present; otherwise, in the current Greg box provided that the
coordinate  system  is  equatorial). The current marker type is used and
the size depends on the current marker size and on the star magnitude.

Proper motions are included to represent the sky at the  date  specified
in the /EPOCH option if present, at the current day otherwise.

The stars are precessed to the equinox specified by option  /EQUINOX  if
present, and otherwise to the equinox defined in variable G_EPOCH.

If option /OUTPUT is present, star positions, magnitudes, spectral types
and  names  are  written  in  the specified file (which defaults to user
terminal).

NOTES:
  - Palomar/SRC Survey Schmidt Plates (POSS#I) are squares of size 35.56
    cm (14 inches) with a scale of 67.14 Arcsec. per mm.
  - ESO Survey Plates are squares of size 30 cm with  a  scale  of  67.6
    Arcsec. per mm.
If you want to make overlays, do not forget the /EXACT option in command
HARDCOPY/PLOT ...
```

5.2 GSC

```
    GSC [RAmin RAmax DECmin  DECmax]  [/MAGNITUDE  Mlim]  [/EPOCH  date]
[/OUTPUT File Ident] [/MERGE]

Search and plots all stars in the Guide Star Catalog with magnitude less
than  Mlim  (brighter  than Mlim) in the (RA,DEC) box specified by RAmin
RAmax DECmin DECmax (if present, and in the current Greg box  otherwise,
provided that the coordinate system is equatorial). Please note that the
```

very brightest stars (V<6) have only been added at version 1.1 and will be missing if you are using GSC version 1.0. Most of them can be obtained by using the ASTROM command. The current marker type is used and the size depends on the current marker size and on the star magnitude.

No proper motion is applied as this information is not available in versions 1.x of the GSC catalog. The /EPOCH option is therefore not used.

If option /EQUINOX is present, the stars are precessed to the equinox it specifies. They are otherwise precessed to the equinox of the map, as defined in variable G_EPOCH.

If option /OUTPUT is present, star positions and magnitudes with their error estimates, are written in the specified file (which defaults to user terminal), together with some ancillary Guide Star Catalog information (original plate, multiplicity flag,...). The presence of the second argument, Ident, will generate an identification number for each star in the output file, and plot this number near the star's position on the graphic display. Useful for identification purposes. The list of information stored in the file is located under subtopic Format

If option /MERGE is present, multiple detections of objects on overlapping plates are merged (by simple averaging). Individual entries are otherwise plotted and listed.

NOTES:
- Palomar/SRC Survey Schmidt Plates (POSS#I) are squares of size 35.56 cm (14 inches) with a scale of 67.14 Arcsec. per mm.
- ESO Survey Plates are squares of size 30 cm with a scale of 67.6 Arcsec. per mm.

If you want to make overlays, do not forget the /EXACT option in command HARDCOPY/PLOT ...

5.2.1 GSC FORMAT

Format of table lines is ([i4],I5,F9.5,F9.5,F5.1,F5.2,F4.2,I2,I1,A4,A1) for
* [optional star Ident number]
* ID within region
* RA (degrees)
* DEC (degrees)
* Position error (arc seconds)
* Magnitude
* Magnitude error
* Magnitude band
* Classification
* Plate ID
* Multiplicity flag

5.3 HEADER

```
HEADER [Image_Name] /EXTREMA
```

```
Compute the extrema of the current or specified image,  and  update  the
header accordingly.

    HEADER /UPDATE

Update the header of the current image, for example after  modifications
to the G_* header variables have been made. The header variables are
G_NDIM         Number of dimensions
G_DIM[4]       Image dimensions
G_CONVERT[3,4] Conversion formula (Ref, Val, Inc for each axis)
G_BLANK[2]     Blanking and tolerance
G_MAX          Maximum value
G_MIN          Minimum value
G_EXTREMA      Zero (0) if no extrema defined
G_WHERE[2,4]   Pixel of maximum and minimum
G_UNITi        Units of axis i
G_UNIT         Units of map
G_COORD        System coordinates
G_SOURCE       Source name
G_EPOCH        Epoch of coordinates
G_RA           Right Ascension (of Object)
G_DEC          Declination (of Object)
G_LII          Galactic longitude (of Object)
G_BII          Galactic latitude (of Object)
G_PTYP          Projection Type (0:none,1:Gnomonic,2:Orthographic,
                3:Azimuthal,4:Stereographic,5:Lambert,6:Aitoff,7:Radio).
                Warning: these codes are subject to change without notice.
G_XAXI         The first axis of projection (1,2,3 or 4)
G_YAXI         The second axis of projection (1,2,3 or 4)
G_A0           Position of projection center for axis G_XAXI (R.A. or LII)
G_D0           Position of projection center for axis G_YAXI (DEC. or BII)
G_ANGLE        Angle of projection (East of North)
G_LINE         Line name
G_FREQRES      Frequency resolution
G_RESTFRE      Signal rest frequency
G_FREQOFF      Image rest frequency
G_VELRES       Velocity resolution
G_VELOFF       Velocity of reference channel
G_PA           Position angle of beam
G_MINOR        Minor axis size of beam
G_MAJOR        Major axis size of beam
G_BEAM         Size of RESOLUTION section (0 or 3)

    HEADER [Image_Name] [/OUTPUT FIle_Name]

Give the Header of the current or specified Image, on the screen  or  in
the  specified output file.  Provided you have write access to the image
and nobody else is already using it, the header may be  editted  if  the
terminal  is  a  VT100 compatible. The keys of the VT100 keypad have the
same meaning as in EDT:
           +--------+--------+--------+--------+
           |  enter |        |        |  Del L |
           |  edit  |        |        |        |
```

```
            +--------+--------+--------+--------+
            |  print |        |        |  Del W |
            |  header|        |        |        |
            +--------+--------+--------+--------+
            | Advance| Backup |        |  Del C |
            |        |        |        |        |
            +--------+--------+--------+--------+
            |  Word  |   Eol  |  Char  |        |
            |        |        |        |  Enter |
            +--------+--------+--------+        |
            |       Line      |        |(Update)|
            |                 |        |        |
            +-----------------+--------+--------+
 To edit a field, press the GOLD (PF1) key.
 To compute MAXIMA and MINIMA of current image, strike ENTER key.
 Strike <^Z> to exit.
 The header will be automatically updated if any modification  has  been
made.
```

5.4 IMAGE

IMAGE [Filename] [/PLANE I1 I2] [/SUBSET Imin Imax Jmin Jmax] [/WRITE]

Read the GDF map Filename. The default image extension is .GDF This command defines the image found in given file as the new current image. Projection, system and extremas are also defined if needed. Unless options are present, the first plane of the image becomes the new Regular Grid array in GreG.

The /PLANE I1 I2 option allows to define part of a more than 2-Dimensional image as the regular grid array. A file name should not be specified if you want to select a new plane in the current image.

The /SUBSET Imin Imax Jmin Jmax option allows to load only a subset of the input image or image plane. Again, a file name should be specified only to change the input image.

The /WRITE option allows you to map the image writeable (instead of Readonly by default), a prerequisite to use some functions (e.g. KILL which kills pixels). The mapped image is then non-shareable by other users...

You can combine /PLANE and /SUBSET options.

5.5 IRAS

IRAS Band [Ramin Ramax Decmin Decmax] [/FLUX Threshold] [/SIZE Flux_value]

Plots a marker at the position of all IRAS point sources detected at band Band (specified by its wavelength in microns 12, 25, 60 or 100) in the area defined by Ramin, Ramax (in hours), Decmin, Decmax (in degrees). If the option /FLUX is given, only sources with a flux greater

```
than Threshold will be select. If the option /SIZE is given, the  marker
size  will  be proportional to the logarithm of the flux (sources with a
flux equal to Flux_value having the current marker size as specified  by
SET MARKER command). The current marker type is used.

Command IRAS completely ignores the current GreG box to select the  IRAS
point sources, but the markers will be clipped in this box when plotted.
Hence, it is up to the user to ensure that the system is EQUATORIAL  and
that  the  projection  area (specified by command LIMITS and PROJECTION)
reasonably matches the selection area specified in command IRAS.
```

5.6 KILL

```
    KILL

KILL calls the interactive cursor. Recognised keys are:
  - V to give the value of the pixel
  - K to give the current blanking value to the pixel
  - I to interpolate the value from the neighbour pixels.
  - E to exit from interactive cursor mode
Any other key is ignored.
```

5.7 PSC_IRAS

```
    [ALL\]PSC_IRAS [Ra. Dec.] [/OUTPUT Filename] [/LOOK Around]

Find  IRAS  characteristics  of  point  sources  around  the  specified
position.  If no position is given, it calls the interactive cursor, and
uses the returned coordinates to look for all IRAS sources  around  this
point.  The  value  Around (argument of option /LOOK) is the size of the
search area in arc minutes (default 1). The result depends  whether  the
full compact catalog is on line or not.
  - If not, the fluxes of the sources found in the search box are  typed
    on the screen.
  - If it is on line, full information about the sources is either typed
    on  the  screen,  or  written  on the file specified by the argument
    Filename of option /OUTPUT.
In addition,  a  binary  output  file  (whose  name  is  PSCSUB.TAB)  is
produced.   This binary file is suited for later processing by the Point
Source Catalog software. The same files (formatted and binary) are  used
for  multiple  use  of  this command (unless the Filename is explicitely
changed).
```

5.7.1 PSC_IRAS FORMAT

```
The listing produced by PSC_IRAS or the IRAS command has  the  following
format :

Right Ascension         Sexagesimal notation in Hours
Declination             Sexagesimal notation in Degrees
Lii                     (Degrees)
Bii                     (Degrees)
```

```
Semi-Major axis of position uncertainty ellipsis  (  Arc Seconds
Semi-Minor axis of position uncertainty ellipsis   95 confidence)
Position Angle of the uncertainty ellipsis. (Degrees from North)
Number of Hour confirmations
Number of LRS spectra
Type of LRS spectra
Variability flag
Discrepant flux flag
Confusion flux flag
Number of nearby hour confirmed point sources
Number of nearby week confirmed point sources
High source density bin flag
CIRR1   Cirrus flag 1
CIRR2   Cirrus flag 2
CIRR3   100 micron sky brightness (MJy/Sr)
Number of identifications
Type of identifications

For each flux (12, 25, 60 100 microns)
        Flux    (Jy)
        Flux quality (1 upper limit, 2 poor, 3 good)
        Calibration uncertainty (%)
        Signal to noise ratio multiplied by 10
        Point source correlation coefficient
        Number of hour confirmed small extended sources
        Number of week confirmed small extended sources

Identifications (one line per identification)
        Name
        Catalog dependant informations
        Catalog name
```

5.8 REGRESSION

```
    REGRESSION Nmin [/RANGE [X Xmin Xmax] [Y Ymin Ymax]]

Computes linear regressions in a image which is the correlation  of  two
coincident images. The regression lines
       Y = A * X + B   and    X = A'* Y + B'
are computed assuming the weight of any (X,Y) is the value of the  pixel
at  X,Y. Nmin is the minimum pixel value considered as significant.
```

5.9 SPECTRUM

```
    SPECTRUM I [J [K]]

Extract a spectrum from an image according to the following numbering :
    Spectrum (l) = Image (l,i,j,k)
for a 4-dimensional image. The spectrum is loaded into the  X,Y  buffers
of  GreG.  It  can  be  processed later by the standard commands LIMITS,
CONNECT, CURVE and so on.
```

6 Communication with the Outer World

6.1 From CLASS

Some **CLASS** commands directly produce **GILDAS** images. `ANALYSE\CUBE` produces a spectral line data cube, `ANALYSE\STRIP` an image (either position-velocity, or continuum map). Command `ANALYSE\GRID` produces a pseudo-table which can further be processed to produced a spectral line data cube.

For spectra, tables can be produced by commands `ANALYSE\GREG`, `ANALYSE\PRINT`.

6.2 From AIPS

There was a dedicated AIPS task (named `GILDA`) to produce **GILDAS** images from AIPS maps. Coordinate systems are entirely preserved, including projection information. If this task is not available in your AIPS installation, ask your local **GILDAS** and AIPS managers to do it.

6.3 From a Program

A complete and efficient use of the **GILDAS** format from a program is beyond the scope of this chapter : refer to the programmer's guide for that. There is however a simple and easy-to-use subroutine that accepts a Fortran array and writes a **GILDAS** image. The price for simplicity is that only minimal header information is written: many useful stuff like World coordinates, Blanking value, and so on, will be missing. With this restriction, you can simply write a Fortran array in **GILDAS** format using the routine `GDF_IMAGE`.

```
     SUBROUTINE GDF_IMAGE(NAME,NX,NY,NZ,NT,Z,ERROR)

Where the arguments are
     NAME   C*(*)   the GILDAS file name
     NX     I       the number of pixels in the first dimension
     NY     I       the number of pixels in the second dimension
     NZ     I       the number of pixels in the third dimension
     NT     I       the number of pixels in the fourth dimension
     Z      R*4     the array of dimensions NZ,NY,NZ,NT
     ERROR  L       a logical error flag (Output)
```

If you need more complete information, refer to the programmers guide. You may also later modify the file header using command `HEADER` in GRAPHIC, or through the **SIC** variables defined by commands `DEFINE HEADER` or `DEFINE IMAGE`.

6.4 From other Packages: the ACCEPT command

The `ACCEPT` command in **SIC** often allows to directly read files from other packages provided one knows the file structure. Here is an example of how to read a bitmap from a PostScript file.

```
SIC\DEFINE INTEGER A[314,590] J
SIC\ACCEPT A/ARRAY IC348.TXT/FORMAT "(8(36Z2.2,/),26Z2.2)"
SIC\DEFINE IMAGE B TEST.GDF REAL/LIKE A
FOR I 1 TO B%DIM[2]
LET J B%DIM[2]+1-I
LET B[J] A[I]
NEXT
```

We assume you have extracted (using a standard text editor) the bitmap in a intermediate file (here IC348.TXT), and taken the sizes (here 314 by 590) from the PostScript description. The format specifier is generic, except for the exact padding required to read one bitmap row at a time.

6.5 From and To other Packages: FITS

For astrnomical applications, conversion from other data formats can be done through FITS format. The FITS image can reside on tape, or on disk.

Tape FITS conversion can be done only through the GFITS program. GFITS is a FITS to (from) **GILDAS** data format translator.

Disk FITS conversion is best done using the `FITS_GILDAS` (FITS to **GILDAS**) and `GILDAS_FITS` (**GILDAS** to FITS) Tasks, which process one file at a time. The GFITS program can be used also.

```
DISMOUNT        : Logically dismount the tape
INIT            : Initialize the tape
HEADER [Name]   : List a FITS header
IMAGE Name      : Define the GILDAS image name for the Read or Write
LIST [Keyword]  : List headers of all files from current position
MOUNT           : Switch to Tape FITS, and logically mount the tape
READ [Name]     : Read the a FITS file to create a GILDAS image
REWIND          : Rewind the tape
SKIP N          : Skip N files on tape
STYLE Arg       : Define the FITS "Style"
SET Arg         : Set some parameters
WRITE [Name]    : Writes the current GILDAS image in FITS format.
```

Since FITS keywords are not fully normalized, although a standard subset is recommended, the GFITS program supports keyword redefinition. If you receive a tape with source name coded as `SOURCE` (instead of `OBJECT`), all you need to do is to define a **SIC** symbol named `SOURCE` with translation `OBJECT`. This is done simply by typing the command

```
SIC\SYMBOL SOURCE OBJECT
```

This trick does not work with the `FITS_GILDAS` task.

7 GFITS Language Internal Help

7.1 DISMOUNT

```
DISMOUNT
```

```
Logically dismount the tape. The tape is not unloaded by this command.
```

7.2 HEADER

```
HEADER [Filename]
```

```
Without argument, list the next header on tape. The tape is left at  the
end  of the file, ready to read the next header, or may be it is left at
the beginning of the current file, ready to process the same file header
again, I can't remember.

With an argument, list the header for the specified FITS file.
```

7.3 IMAGE

```
[FITS\]IMAGE Name
```

```
Define the GILDAS file to be used for the next READ or WRITE commands.
```

7.4 INIT

```
INIT
```

```
Initialize the tape without any label (standard FITS tape).
```

7.5 LIST

```
LIST [Keyword]
```

```
List headers of all files  from  current  position.   If  a  Keyword  is
specified, only this keyword will be listed.  The list can be aborted by
<^C>, but I am not quite sure about what the tape position will be after
such an horrible action...
```

7.6 MOUNT

```
MOUNT [Device:]
```

```
Logically mount the tape on the specified tape drive, or  by default the
one  indicated  by the logical name TAPE_DEVICE.  This must be done once
for each tape before any other command (included INIT).
```

7.7 READ

```
READ [Filename] [/BLC Bx [By [Bz[ Bt]]]] [/TRC Tx [Ty [Tz [Tt]]]]
```

```
Without argument, read from tape the current FITS file to create  a  new
GDF  image.  Options /BRC and /TRC can be used to select a subset of the
input FITS image only.

With argument, read from the specified FITS file.
```

7.7.1 READ /BLC

```
Define the bottom left corner of the input FITS file to  be  considered.
Bx  By Bz Bt indicates the pixel coordinates of this bottom left corner.
Trailing arguments (if the image has less  than  4  dimensions)  can  be
omitted, and 0 means 1.
```

7.7.2 READ /TRC

```
Define the top right corner of the input FITS file to be considered.  Tx
Ty  Tz  Tt  indicates  the  pixel  coordinates of this top right corner.
Trailing arguments (if the image has less  than  4  dimensions)  can  be
omitted, and 0 means the actual dimension.
```

7.8 REWIND

```
REWIND
```

```
Rewind the tape
```

7.9 SET

```
SET Argument Value
```

```
Define some FITS program parameters.
```

7.9.1 SET BLANKING

```
SET BLANKING Breal
```

```
Indicates the blanking value to be used.
```

7.9.2 SET BLOCKING

```
SET BLOCKING Factor
```

```
Indicates the desired tape blocking factor  (for  GILDAS  to  FITS  tape
only). Factor is an integer number from 1 to 10.
```

7.9.3 SET NBITS

```
SET NBITS Value
```

```
Indicates the desired number of bits. Valid values are 16, 32,  and  -32
for IEEE real numbers.
```

7.9.4 SET NOPROMPT

```
SET PROMPT or SET NOPROMPT
```

```
To indicate wether non standard FITS keywords  require  user  input  for
interpretation or not.
```

7.9.5 SET PROMPT

```
SET PROMPT or SET NOPROMPT
```

```
To indicate wether non standard FITS keywords  require  user  input  for
interpretation or not.
```

7.9.6 SET STYLE

```
STYLE Arg
```

```
Define the FITS "Style"  of  the  tape.   FITS  has  unfortunately  many
"styles"  because  the  keywords  are not fully normalized. This program
recognizes automatically some keywords but for some  cases  the  "style"
must be specified either to obtain a proper GILDAS image header from the
FITS tape, or to write the appropriate FITS format (e.g. UVFITS).

  - STANDARD  Normal FITS tape
  - CPC       Chopped  Photometric  Channel  from  the  IRAS  satellite.
    Images are 3-D with 1 plane for each band, and the projection system
    is almost a RADIO one (but not quite exactly). For  FITS  to  GILDAS
    only.
  - SPLINE    Spline maps from the IRAS satellite. Projection system  is
    GNOMONIC  (but  not  with  standard  keywords),  and in addition the
    correct calibration is applied according to  the  note  by  Francois
    Boulanger. For FITS to GILDAS only.
  - UVFITS    UV FITS tape.
```

7.10 SKIP

```
SKIP N
```

```
Skip N files on tape.  N can be positive or negative. N = * can be  used
to go at the logical end of tape.
```

7.11 WRITE

```
    [FITS\]WRITE [Filename]

Without argument, write the current image (as defined by command  IMAGE)
to the output tape. The tape is moved to end of tape before writing.

With argument, create a disk FITS file with the specified name.
```

8 Programing Manual

This section has been moved to the **GILDAS** programming guide.

9 The Current Tasks

This section contains a thematic summary in which some algorithms may appear more than once, followed by a description of each task. Since the list of available programs is growing steadily, for up to date information, use the `EXPLAIN SUMMARY` command in programs VECTOR, GRAPHIC.

9.1 BACKGROUND

```
    BACKGROUND

Computes the background image of a given map.  The basic idea is to make
a  crude  mesh  model of the background by finding the most likely value
of the original map (within some intensity range) for each cell  of  the
crude   mesh.   The most likely value is found by making an histogram of
the intensity distribution in the  specified  range   for   all   points
lying  in  a circle around the cell center.  The circle may (and should)
be greater than the cell size.  If  the  number  of  pixels  within  the
intensity range is too small, no value is attributed to the intermediate
mesh.

The intermediate model is then resampled to the original  map   using  a
general    triangulation    technique.    This    procedure  has  several
advantages over a simpler method which would compute a "smoothed"  image
as  the background, because it is not biased by any emission outside the
selected range, and it is able to interpolate  over  large  non  sampled
area of the intermediate mesh.  Other tasks with similar names use other
interpolation algorithms.  They will hopefully be merged into  a  single
task one (with a switch for the interpolation algorithm).

CAUTION:  In the present version, blanking in the  input  image  is  not
correctly  handled.  This can usually be worked around by specifiying an
apropriate range.
```

9.2 BLANKING

```
    BLANKING

Modifies the blanking value of an image.  This operation cannot  be done
through  the  HEADER  program, which only modifies the blanking value of
the header...
```

9.3 CIRCLE

```
    CIRCLE

This program computes annular averages on an input cube to   produce  an
output  map.   Each line of the output map is the radial profile for one
particular plane of the input cube.  The task can be used to  derive the
radial  profiles,  for  example  in circumstellar envelopes.  The radial
pixel separation is always the X and  Y  increment.   If  the  X  and  Y
increments  are  different,  the  averages  are  therefore  computed  on
elliptical rings, despite the name of the task.
```

9.4 CLEAN

```
    CLEAN

This  task    contains    implementations    of    several    CLEAN-like
deconvolution  algorithm.  The algorithm is selected through the METHOD$
keyword, which can be :
 SIMPLE    :  The straightforward direct method.
 CLARK     :  The Barry Clark major-minor cycle method.
 MRC       :  a Multi-Resolution Clean.
 THRESHOLD :  Components are not restricted to be point sources.

The input map and beam should have the  same  sizes,  except   for   the
SIMPLE  method where different beam sizes are allowed, and must have the
same grid spacing.  Sources are searched only in the  inner  quarter  of
the  map,  but  their  sidelobes are removed in the full map.  A smaller
search box can be specified if desired.

The program can process all planes of a data cube at the same time,  but
a   separate  list  of  component  will be produced for each plane.  The
cleaning cannot be restarted.
```

9.5 COMBINE

```
    COMBINE

This task makes "combinations" of two  input   images   to   produce   a
third  one.   The  two input images may have the same dimensions, or the
first one (Z one) may have less dimensions than the second (Y) one.   In
the latter case, combinations will occur for all the extra planes of the
Y image.  For example you can divide all planes of an input (Y) 3-D cube
by  a  2-D (Z) image, provided each plane of the cube matches the single
image...

Operations are
        ADD             X = Ay*Y + Az*Z + C
        MULTIPLY        X = AY*Y * AZ*Z + C
        DIVIDE          X = AY*Y / AZ*Z + C
        OPTICAL_DEPTH   X = - LOG (AY*Y / AZ*Z + C)
provided Y > TY and Z > TZ , TY and TZ being thersholds for the Y and  Z
images.

Image combinations may  also  be   done   using   the   SIC   arithmetic
capabilities,  but  COMBINE  offers  the advantage of handling correctly
blanking information.
```

9.6 CONVERT

```
 CONVERT

 This task produces files that contain two planes: the source in the
 first, the weighting function in the second. The MAP procedure plots the
 weights if the variable PLOTWEIGHT is true (Default value of PLOTWEIGHT
 is NO).  Depending on variable SCAN_LIST two different processes are
```

```
executed:

    - SCAN_LIST is left open. The task CONVERTs the single-beam maps of
      all unflagged channels which were obtained with RESTORE or with the
      SHIFT_AND_ADD technique to a single equatorial map. The horizontal
      maps are weighted with their one sigma noise level (CHAN_RMS) squa-
      red, shifted to a common reference position, transformed to the
      equatorial frame of reference and, finally, combined to a single
      map. By default, CONVERT uses the same gridding function as MAKPLAN.

    - SCAN_LIST contains a list of scans: the task creates a mosaic from
      the equatorial maps associated with a list of scans. Each map is
      weighted according to its weighting function, shifted to the common
      center of reference, and added.  Modify the center of reference and
      the size of the mosaic by setting the appropriate input variables.
      Type INPUT CONVERT to verify the content of the variables.
      As CONVERT produces mosaics with the source name as filename, use
      command MAP Source_Name to plot the results.
```

9.7 CORRELATE

```
    CORRELATE

Computes correlation of two images or data  cubes.   The  result  is  an
image (data cube) containing

      Out(i,j) = < In1(k-i,l-j)*In2(k,l) >         averaged over k,l
      For mode correlation (MODE$ = YES)
or
      Out(i,j) = < In1(k-i,l-j)**2 + In2(k,l)**2
                   - 2 * In1(k-i,l-j)*In2(k,l) >   averaged over k,l
      For mode square (MODE$ = NO)

Actually, linear conversion formulas are used to  keep  the  correlation
image meaningful in user coordinates.  The input images must match.

When used for example to  recenter   images,   the   position   of   the
maximum  of the correlation image (or equivalently of the minimum of the
sum of squares  image)  yields  the  required  recentering.   MODE$  YES
(Correlation)  is  to  be  used when the input distribution has a finite
extent, while MODE$ NO (Square) can be used in any case, but is somewhat
slower of course.
```

9.8 CV_SMOOTH

```
    CV_SMOOTH

This task smoothes an image  using  a  cross-validation  algorithm.  The
idea   behind   the  algorithm  is  to use all data points except one to
estimate the value of this one point.  The difference between  estimated
and measured values is used to determine the noise level in the map, and
therefore the appropriate amount of smoothing.   The  smoothing  is  non
uniform,  and  depends  on  the  signal to noise:  low level emission is
```

smoothed more than strong peaks.

The present algorithm has a number of restrictions which we hope to suppress in future versions:
- It flatly refuses to do anything if the noise is correlated between adjacent points, but it takes quite a long time to find this... Of course this always occurs for interferometric maps. Hence, it is possible to add a small amount of extra noise to the image to by-pass this stupid restriction.
- It is a basically 1D algorithm. The algorithm smoothes along axis 1 then 2 first, and in a second pass along 2 then 1. It takes the average of the two results to produce the smoothed image, and keeps the difference which should be representative of the errors. A 2D generalisation of the algorithm exists (Girard D., 1987, Rapport de Recherche RR 669-M, TIM3. Universite de Grenoble), and we hope to implement it in the near future.

9.9 DG_SMOOTH

```
                    DG_SMOOTH
    Smoothes an image using the Conjugate Gradient Algorithm.
                        (Author: Didier GIRARD, Groupe d'Astrophysique)
```

The smoothed image is the equilibrium state of a thin flexible plate constrained to pass near each height datum by a spring attached between it and the plate. The smoothing is controled by the "Smoothing Parameter" P :

P = (plate stiffness) / (springs stiffness)

A low value for P (0.001) means high fidelity to the original data and should be used for high signal to noise ratios. If the data are noisy, use higher values of P (0.1 to 10). The algorithm is iterative and needs a work space equal to 3 input maps. It usually converges in 10 iterations, and can be restarted if you save the work files. Timing is of the order of 2 seconds of microVAX II CPU time for a 128 by 128 map. It takes into account blanked pixels properly.

9.10 EXAMPLE

This is a sample program doing nothing, but used to test GILDAS...

9.11 EXTRACT

EXTRACT

Extracts a subset of an input image. The output image can be larger than the subset; in this case the additional pixels are blanked if the output image is being created, unmodified if it already exists. It works on images of any dimensions, and any subset of the input image can be placed anywhere in the output image. This routine can also be used to build a N+P dimensional image from a set of N dimensional ones, by initializing the output image once with its full dimensions and then placing the (subset of the) input images at the appropriate place in the output image.

9.12 EXTREMA

```
    EXTREMA
```

A simple minded routine to compute and store the extrema of a GDF-like file into its header. It is functionnally equivalent to the command "HEADER File/EXTREMA" in the GRAPHIC program.

9.13 FIELD_FIND

```
    FIELD_FIND
```

Makes the field labelling of an image, i.e. identifies connex areas with image values higher than a given threshold and attribute them a number. The result is an image (of same size as the input one), in which the value of a pixel is the number of the field to which it belongs. A zero value is attributed to pixels under the threshold.

Note that blanked pixels may adversely affect the field labelling...

9.14 FIELD_LIST

```
    FIELD_LIST
```

This task computes statistics on fields of an image. It accepts as input the image itself and a label image describing the fields. The label image is an image (of same size as the input image), in which the value of a pixel is the number of the field to which it belongs. A zero value is attributed to pixels which do not belong to any field. The label image is usually obtained by running task FIELD_FIND.

```
     The output is a Table file containing :
Col 1 Number of pixels
Col 2 Integrated flux
Col 3 X Abscissa of centroid ( i.e., Weighted by pixels value)
Col 4 Y Ordinate of centroid ( idem)
Col 5 Major axis of fitted ellipsis       (should be correct now)
Col 6 Minor axis of fitted ellipsis       (idem)
Col 7 Position angle of fitted ellipsis   (idem)
```

9.15 FIELD_STAT

```
    FIELD_STAT
```

This task computes the integrated spectrum of a number of fields in a data cube. It requests as input a 3-D image, and a label image (usually produced by task FIELD_FIND). The output is a table where every even column is the average spectrum for one field. Odd columns contain the number of data points in the corresponding field (not an optimal storage indeed!).

9.16 FILL_CUBE

```
    FILL_CUBE

This program resamples an input data cube on  a  finer  grid   for   the
first  two  dimensions (hence the data cube is treated as an ensemble of
images).   The  output  grid  may  be  explicitly  determined  from  the
conversion formulae of the two first axis, or implicitly from the number
of pixels of the two first axis and the input cube conversion  formulae.
Two different methods are available for the resampling :

The SLOW method, which is general and takes properly  into  account  the
input blanking value.  This method is based on the same algorithm as the
RANDOM command in  GreG.   It   triangulates   the   input   non-blanked
values  and  uses Lagrange polynomials to interpolate on the finer grid.
For optimisation purposes,  the  same  triangulation  is  used  for  all
planes,  which  assumes  that the same pixels are blanked in all planes.
If this is not the case, the individual planes should be extracted,  and
processed separately.

The FAST method, which ignores   the   input   blanking   value.    This
method  is  based on the same algorithm as the RESAMPLE command in GreG.
It is faster, but should be used only if no input pixel is blanked.

If the data is undersampled, this task is  the  recommended  second step
in  the  data  analysis  of  a  cube produced by command ANALYSE\CUBE in
CLASS, immediately after  the  TRANSPOSE  program.  Task  MAKE_CUBE  is
preferable for oversampled data.
```

9.17 FITS_GILDAS

```
    FITS_GILDAS

Converts a disk FITS image in a GILDAS image.  A disk FITS image  is  an
2880   bytes   per   block   file  conforming  to  FITS standard for the
information written in  the  blocks.   Such  disk  FITS  images  can  be
produced  by  other  packages (e.g.  AIPS).  This task is intended for a
single image.  For many  images,  or  tape  FITS,  use  the  interactive
program GFITS.

Images are assumed to have at most  4  axis.  FITS_GILDAS  also  handles
UVFIT data.

Conversion for FITS to GILDAS data format is not necessary: "on-the-fly"
conversion  of  FITS files into GILDAS images is automatically attempted
by all GILDAS programs and tasks. This conversion  creates  a  temporary
GILDAS  image  (of  extension  .fits.gdf) which is normally deleted upon
completion. However, this "on-the-fly" conversion will fail if the  FITS
keywords are highly non-standard.
```

9.18 FLOW

```
    FLOW
```

This is a dedicated routine to produce bipolar outflow maps, with optimum signal to noise ratio and little bias. It takes as input a cube (N by NX by NY) with the velocity along the first axis, and produces a pseudo-cube cube (3 by NX by NY) containing the maps of the line width and of the red and blue lobes integrated intensities.

For each spectrum, the algorithm determines the peak channel, then determines on each side of this channel the velocity at a given threshold, given as ratio to peak value. From this velocity, it integrates out to a second threshold. The area found is the contribution to the blue (or red) lobe of the flow.

9.19 FOURIER

FOURIER

This task computes the (complex) Fourier transform of a (real) 2D image. It does not request the number of pixels to be powers of two, but the algorithm is of course much faster if they are.

9.20 GAUSS_1D

GAUSS_1D

This task makes a multi-component non-linear gaussian fit into columns of a table.

It fits a function Y = f(X) where f is the sum of up to five gaussian, and X and Y data are taken in specified columns of the input table. The method is identical to that used by the GAUSS command in CLASS (simplex followed by conjugate gradient). The format for input of initial parameters is also similar. All messages are sent to a specified output file. The fitted profiles can be kept in separate columns of the input table.

9.21 GAUSS_2D

GAUSS_2D

This task fits a single 2 dimensional elliptical gaussian to an image. The fitted gaussian is kept on an output image. All coordinates are USER coordinates, not pixels. Each parameter is followed by a code to indicate whether it is fixed (code 1) or variable (code 0).

If the input image is actually a cube, a gaussian is fitted to each plane.

9.22 GAUSS_SMOOTH

GAUSS_SMOOTH

This task performs a 2-D smoothing by an elliptical gaussian. The algorithm works in the Fourier plane and the first 2 dimensions (in pixels) should therefore be powers of two (but need not be equal). It is much faster than the equivalent sky plane algorithm, but you should beware of aliasing effects when the size of the convolving gaussian becomes a sizeable fraction of the map (say 30%).

9.23 GILDAS_FITS

```
GILDAS_FITS
```

Converts a GILDAS image in a disk FITS image. A disk FITS image is an 2880 bytes per block file conforming to FITS standard for the information written in the blocks. Such disk FITS images can be produced by other packages (e.g. AIPS). This task is intended for a single image. For many images, or tape FITS, use the interactive program GFITS.

9.24 GRID_CUBE

```
GRID_CUBE
```

This task makes a cube from a table containing data regularly sampled on a 2-d grid, but possibly uncompletely sampled. Several gridding modes are possible.

In mode GRID, since the input data is assumed to be regularly sampled, this gridding task does not use any convolution or interpolation, but just fills a grid with values.

In mode NATURAL, a convolution kernel is used to interpolate the irregularly sampled data using the weight of each individual point.

In mode UNIFORM, the same convolution process is used but the individual points are assumed to have equal weight.

The grid step is specified by the user, but the map size can be determined by the task. The input table can be created by command GRID in CLASS, or using task TABLE from a formatted file.

9.25 GRID_EXTEND

```
GRID_EXTEND
```

This task makes a cube from a table containing irregularly sampled data. The data is interpolated/extrapolated using a convolution taking into account the beam size of the observing telescope.

In mode NATURAL, a convolution kernel is used to interpolate the irregularly sampled data using the weight of each individual point.

In mode UNIFORM, the same convolution process is used but the individual points are assumed to have equal weight.

The pixel size is specified by the user, but the map size is determined

```
by  the task.  The input  table can be created by command GRID in CLASS,
or using task TABLE from a formatted file.
```

9.26 GRID_SG

```
    GRID_SG

Optimum image reconstruction from irregularly sampled data  with  finite
angular resolution.

This task makes a cube from  a  table   containing  irregularly  sampled
data.   The data is interpolated/extrapolated using a convolution taking
into account the beam size of the  observing  telescope.  The  smoothing
effect  of  the  convolution  can be optionally removed by this task, by
Fourier filtering. Fourier filtering  is  also  used  to  suppress  high
spatial frequency noise. The resulting image is thus optimal in terms of
signal to noise and angular resolution.

In mode NATURAL,  a  convolution  kernel  is  used  to  interpolate  the
irregularly sampled data using the weight of each individual point.

In mode UNIFORM, the same convolution process is used but the individual
points are assumed to have equal weight.

The pixel size is specified by the user, but the map size is  determined
by  the task.  The input  table can be created by command GRID in CLASS,
or using task TABLE from a formatted file.
```

9.27 HEADER

```
    HEADER

A (not so) simple minded routine to list edit and modify the  header  of
a  GDF-like file (Image only, Table edition is usually meaningless).  It
is functionnally equivalent to command "HEADER File"  in   the   GRAPHIC
program.



On some systems, editing of header parameters is possible  if  you  have
write access to the image.

  -  To modify a header parameter, position the   cursor   on   it   and
    press   key Gold (PF1).  You can then enter the new value.  When you
    are happy with the modified value, press Enter.
  -  To compute the extrema of the image, press key Enter.
  -  To exit from the edit mode, type <^Z>.
```

9.28 HISTO_CLOUD

```
    HISTO_CLOUD
```

```
Makes the cross histogram of two input  images.   The  result  is  a
table of two columns:
  - First column :  value of pixel (I,J) of first image
  - Second column :  value of pixel (I,J) of second image

Using the GreG command POINT on this  output  table  will  produce  a
cloud  plot  of the correlation between the two images.  This program is
reasonably well adapted for small images, but you should use HISTO_CROSS
for large images.
```

9.29 HISTO_CROSS

```
    HISTO_CROSS

Computes the cross histogram of two input  images.  The  result  is  an
image  of  dimensions  (Number of slots for first image, Number of slots
for second image) which  represents  the  density  distribution  of  the
correlation.  The value at a given (I,J) is thus the number of pixels in
the input images that have the value corresponding  to  slot  I  in  the
first  image  and  to slot J in the second one.  The output image can be
used as input to task REGRESSION to evaluate some statistical parameters
of  the  correlation.  See  also  HISTO_CLOUD  for a slightly different
information.
```

9.30 HISTO_DOUBLE

```
    HISTO_DOUBLE

Computes the histogram of an image (Y_image)  as   a   function   of   a
second  one  (Z_image).  This  gives  an  information  similar  to, but
slightly different from  the  cross  histogram  (cf  HISTO_CROSS).  The
program  computes  histogram  of  the mean value and standard deviation.
The result is a table of 4 Columns, in which
  1) the first column contains the mean value of the  Y_image   (average
    of all pixels in the Y_image for which the Z_image value is included
    in the corresponding slot)
  2) the second the standard deviation (of the Y_image values).
  3) the third the number of pixels of the Y_image   used   to   compute
    the mean and deviation.
  4) the fourth column contains the histogram abscissa   (actual   value
    of the input Z_image)
```

9.31 HISTO_SIMPLE

```
    HISTO_SIMPLE

Computes the histogram of an image (or of a table).   A  subset  of  the
image   may   be   specified.    The  result is a table with 2 columns :
Column 1 contains the middle value of the  interval,  Column 2  contains
the number of input image pixels in the interval.
```

9.32 HISTO_TABLE

```
    HISTO_TABLE
```

Computes the cross histogram of two columns of a table. The result is an a 2-D image where the value of pixel (I,J) is the number of points in the input table for which the first column value correspond to bin I, and the second column value to bin J.

9.33 IMAGE

```
IMAGE
```

This task transforms an RGDATA file to the standard GILDAS format. This can also be done within the main GRAPHIC or OVERLAY programs, using command RGDATA and then WRITE IMAGE.

9.34 INTERPOLATE

```
INTERPOLATE
```

This program resamples an input data cube ALONG ITS FIRST AXIS. The resampled output cube may have higher or lower resolution than the original one, but extrapolation is strictly forbidden. The program does not handle blanking values.

9.35 LIST

```
LIST
```

This program lists in free format part of an input table.

9.36 MAKE_BACK

```
MAKE_BACK
```

Computes the background image of an input image using a thresholding and a smoothing algorithm based on the conjugate gradient method. The background image is computed as follows

- First the image is thresholded between THRESHOLD$[1] and THRESHOLD$[2]. All pixels outside this range are blanked.
- Then the image is smoothed using the conjugate gradient method, with NITER$ iterations and a smoothing parameter P$. Blanked pixels are thus interpolated in the process.

The output image is the background image. P$ should be large enough to yield some smoothing (1 to 100, try...), and NITER$ also to assure convergence especially in "blanked" areas (NITER$ = 20 is typical).

9.37 MAKE_CUBE

```
MAKE_CUBE
```

This is an image construction task which is able to produce a filled image from one containing many blanked pixels. The reconstructed filled image is not constrained to fit exactly the observed data points. On the opposite, the construction is made by the analogy to a flexible plate attached to fixed points by springs: the plate is the analogous of the surface represented by the image, and the fixed points are the analogous of the observed data points. By adjusting the parameter P

P = (plate stiffness) / (springs stiffness)

it is possible to control the fidelity to the original data and the amount of smoothing involved in the image reconstruction. Low values of P mean high fidelity to observed data, and negligible amount of smoothing.

The original grid is first expanded by a factor EXPANSION$, new pixels being attributed the blanking value. Then, the minimization proceeds iteratively to adjust the final image, until convergence is reached. Initially blanked pixels are ignored in the convergence criterium.

The algorithm works on cubes, processing each plane independently. It can be used as an alternative to FILL_CUBE for oversampled images, but works also for undersampled data.

9.38 MAKE_MOSAIC

This task combines individual dirty images and beams corresponding to various pointing centers into a "mosaic" which is suited for further deconvolution using the MAPPING program.

Mosaics are ensemble of overlapping fields, with typically half power primary beam overlap, and similar (but not identical) UV coverages for each field. Individual images for every field should be created with a common phase tracking center. The UV_SHIFT, MAP_RA and MAP_DEC variables in task UV_MAP must be used for this. The image names must derive from a generic name as follows (e.g. for a generic name MAP)

MAP-1.LMV, MAP-2.LMV, ... MAP-N.LMV

where N is the number of fields.

For example

```
LET MAPSIZE 256 512
LET MAPCELL 0.5
LET UVSHIFT YES
LET MAPRA     20:39:10
LET MAPDEC    68:01:20
!
LET WCOL 1
FOR I 1 TO 10
LET NAME CONT-'I'
GO UVMAP
NEXT
```

MAKE_MOSAIC produces a combined image (MAP.LMV), a data cube of primary beams (MAP.LOBE), a data cube of dirty beams (MAP.BEAM), and a weight image MAP.WEIGHT, which are required for cleaning.

Two parameters are very important in the mosaic creation: the primary beam width (FWHM), and the truncation threshold of the primary beam. It

```
is recommended to use a relatively high value for this threshold, 0.2 to
0.4, to avoid contaminating adjacent field in case of pointing errors.
```

9.39 MAKPLAN

```
MAKPLAN

This task creates double-beam maps of ON-THE-FLY scans for RESTORE.
It uses the position variable SCAN_COORD to resample the data variable
SIGNAL on an azimuth-elevation grid G. The data is gridded line by line
(subscan by subscan), apodized in azimuth to the frequency cutoff of the
telescope using the envelope of a sinc function tapered over a gliding
window F, weighted according to the inverse density factor W required to
obtain uniform data sampling and, finally, written to a file with the
extension .mdb. The size of the gliding window is chosen to be 5x the
beamwidth at the effective observing frequency. The weighting factor W
is introduced to control the form of the beam, and its value is related
to the area density of the data at any point j in the new grid:

                      --
                 Wj = \    sin (uj)/(uj).Tj
                      /
                      -- F

             uj = A .pi.D/lambda.(j-SCAN_COORD)

where T is an optional apodization function and A is the ratio of
the halfpower width of the gridding function W to the beamwidth of
the telescope. T can either be truncated-gaussian (by default),
parabolic, linear or equal to unity (no taper). So far, the i-th
point of a specific line in the double-beam map G can be written to:

                      --
            Gi = 1/Wi.\ SIGNAL.sin (uj)/(uj).Tj
                      /
                      -- F

Apodization in elevation is done according to the setting of the
logical SMOOTH_EL. The grid cells are by default quadratic and
their size is set either to the sampling interval in azimuth or in
elevation, whichever is larger. However, different sizes in both
directions can be set using variable MAP_CELL. Flagged channels
are not reduced.
```

9.40 MAP_AVER

```
   Average several planes of a 3-D image to produce a 2-D image.   This
task  can be used for example on LMV data cubes to get average intensity
images. To convert to integrated  intensities,  you  must  multiply  the
resulting image by NC$[2]-NC$[1]+1.
```

9.41 MAP_COMPRESS

Compress the first two dimensions of an image or data cube by powers of 2 using Fourier transform.

9.42 MAP_EXPAND

Expand the first two dimensions of an image or data cube by powers of 2 using Fourier transform.

9.43 MAP_INTER

This program resamples an input data cube ALONG ITS THIRD AXIS. The resampled output cube may have higher or lower resolution than the original one, but extrapolation is strictly forbidden. The program does not handle blanking values.

9.44 MASK

```
MASK
```

This task masks either the inside or the outside of a polygon in all planes of a data cube. It is similar to the MASK command in GreG, GRAPHIC or OVERLAY, but it does affect the output image and not an internal copy of it. The polygon must have been created previously using the WRITE POLYGON command in GreG, for example.

9.45 MERGE

```
MERGE
```

This program merges two input tables WITH THE SAME NUMBER OF LINES.

9.46 MINIMIZE

```
MINIMIZE
```

This task finds the best linear combination of two input arrays, in the form

```
        X(i,j) = A*Y(i,j) + B
```

The results are A and B of course, written in MINIMIZE.GILDAS.

9.47 MOMENTS

```
    MOMENTS

This task extracts the first 3 moments of an image, e.g. for a LMV cube,
the  integrated intensity, mean velocity and line width. A threshold for
detection can be specified. Velocity smoothing before detection  can  be
applied.
```

9.48 MOSAIC

```
    MOSAIC

Combines individual fields of a mosaic in a single image  for  cleaning.
This task makes assumptions about file extensions for the images and the
dirty beams, and computes the primary beams.

Noting x the direct product, and * a convolution, the combined image is
    Sum (Bj x Ij)
where Bj is the primary beam of field j, and Ij its dirty  image.   This
task also compute a weight image, W = 1/Sum(Bj^2).

If the dirty beams Dj were Diracs, the sky distribution would just be
    S = W . B
because Ii = Di * (Bi x S)
```

9.49 MOSAIC_CLEAN

```
    MOSAIC_CLEAN

This  task    contains    implementations    of    several    CLEAN-like
deconvolution   algorithm  working on MOSAICs. The algorithm is selected
through the METHOD$ keyword, which can be :
 SIMPLE    :  The straightforward direct method.
 CLARK     :  The Barry Clark major-minor cycle method.

The input map and beam should have the  same  sizes,  except   for   the
SIMPLE  method where different beam sizes are allowed, and must have the
same grid spacing.  Sources are searched only in the  inner  quarter  of
the  map,  but  their  sidelobes are removed in the full map.  A smaller
search box can be specified if desired.

The program can process all planes of a data cube at the same time,  but
a   separate  list  of  component  will be produced for each plane.  The
cleaning cannot be restarted.

This task uses several input maps:
    .BEAM     A combination of the synthesized beams for all fields
              in order (nx,ny,fields)
    .PRIMARY  A combination of the primary beams for all fields,
              in order (fields,nx,ny)
    .LMV      The dirty mosaic, in order (nx,ny,nv)
```

```
all produced by task MOSAIC from the individual fields.

It produces a number of output maps
    .LMV-MOS  A cleaned, primary-beam corrected, image
    .LMV-RES  A residual image, without primary beam correction
    .CCT      A clean component table
```

9.50 MVIEW

```
    MVIEW

is an X-Window viewer for GILDAS data cubes. MVIEW is  oriented  towards
fast  interactive  display of 3-D data cubes, and has capabilities which
allow it to prepare parameter files for some GILDAS task.

* To increase the speed of  interactive  cursor,  desactivate  the  zoom
window, by clicking the right button in it and select zoom off.

* In NORMAL mode, press button1 to visualize X,  Y,  Z  coordinates  and
spectrum    along    Z.    A    Button1Press    immediately    followed  by  a
Button1Release in the integrated mean window  memories  a  summit  of  a
polygon  in  a  queue; same action with button2 removes the last polygon
summit in the queue; click button3 to close the  polygon  and  visualize
the integrated spectrum inside the polygon on the right side.

* In SLICE mode, press button1 and drag the mouse to define the slice.

* In 3D AREA mode press button2 and drag the mouse to define a reference
subset  in  the  mosaic; when the cursor falls into an other subset, the
box is propagates itself  from  current  to  reference  subset  or  from
reference  to current subset; release button2 and press anywhere else to
select another reference subset; Press button1 (resp  button3)  anywhere
to  define  first  (resp last) subset. In the spectrum window, the first
and last subset are given by the position of  the  big  tics  under  the
integrated  spectrum;  drag  the  tics  with  button1  to  compute  the
integrated mean between the tics.
```

9.51 MX

```
    MX : Make and Clean maps from UV data

This task combines UV_MAP  and  CLEAN  in  a  single  step.   The  CLEAN
algorithm  is  similar  to  the  CLARK  method,  but  with  major cycles
operating directly on the ungridded UV table rather than  in  the  image
plane.  Accordingly, there is aliasing only of the residuals, not of the
clean components.

MX can process all planes of a data  cube  at  the  same  time,  but   a
separate  list  of  component  will  be  produced  for  each plane.  The
cleaning cannot be restarted.

The mapping process is identical to UV_MAP.  It makes a map from UV data
```

by gridding the UV data using a convolving function, and then Fast Fourier Transforming the individual channels. However, MX always produces a single beam for all channels, thus neglecting frequency change between channels.

MX allows to shift the map center and rotate the image, by shifting the phase tracking center and rotating the UV coordinates of the input UV table.

MX produces several output maps, of name derived from MAP_NAME$, and a residual output UV table.

9.52 NOISE_SMOOTH

NOISE_SMOOTH

Smoothes an input image using the noise cheating enhancement method. This method works only for strictly positive images. Values for adjacent pixels are summed until a given total is reached. Then, the total is divided by the number of pixels added and the result is used for the output pixel value. This smoothing is very non linear (in particular no smoothing occur on pixels stronger than the smoothing threshold). A parameter allows to restrict the averaging to nearby pixels only : in this case the output image is not necessarily strictly positive.

9.53 ONOFF

ONOFF

This task produces tables of counts or fluxes, whichever is available, from ON-OFF scans. Select whether you require sky noise removal or not. The following is carried out:

- compute the mean signal <Sj> and its variance for all subscan j.
- remove a baseline from the raw signal by least-square fitting to a straight line a weighted sequence of means. Each mean value is the mean value of subsequent ON and OFF phases.
- on sky noise removal it removes the high frequency noise in excess. In this case the task computes the weighted means of the ON and of the OFF phases for the whole scan.
  - it then removes from a channel k the signal difference between the baseline corrected signal and, depending on subscan, either the mean ON or OFF signal averaged over all other channels except the central one.
- it then removes the excess high frequency noise from all the channels and computes the mean noise corrected signal <Sj> and its standard deviation sj.
- depending on the COUNTS_PER_JY conversion factor it compute either counts or fluxes and the errors W by averaging all weighted differences between the ON and OFF phases.
- finally it converts all the beam offsets from the horizontal to the

```
    equatorial frame of reference for the epoch corresponding to the
    source coordinates.
  - the results are written to files with the extension .onf

Avoid repetitive noise removal. If the fluctuations in two adjacent
channels i and j are correlated, we expect to find on average the
variance in both channels increased by the covariance between the two
channels. Once the correlated term is removed, we expect the fluctua-
tions to have faded away leaving uncorrelated noise with similar va-
riance in channel i and j. Suppose the noise removal was very effec-
tive and no scrap of correlated noise is left, then every further
attempt to reduce the fluctuations with ONOFF will fail. Indeed, as the
fluctuations will be independent further removal will tend to increase
on average the variance of the j-th channel by a fraction M, where M
is the number of channels from which the task derives the correlated
noise contribution.

Be aware that the data can be perturbed by low frequency noise that
can create virtual positive or negative sources. Moreover, the low
skynoise variation can hardly be estimated (in general, not at all)
and, so far, is not taken into account in the standard deviation of
the results.
```

9.54 PLANE

PLANE

This is a simple task that subtracts a plane from an image. The plane is determined by least square fitting through all non-blanked pixels of the original map, weighted by a weight image. The latter can be used to account for a non uniform signal to noise ratio, but is more frequently used to ignore some areas (typically the sources) in the fit.

It supersedes an older obsolete version (still available as PLANE_OLD) which fitted a plane through 3 points of the input image.

9.55 PLANE_OLD

PLANE_OLD

This is a (very) simple task that subtracts a plane from an image. The plane is determined by three non-colinear non-blanked pixels of the original map. The plane image is also computed. This task is usually a first step before more elaborate background algorithms (such as BACKGROUND or MAKE_BACK) are applied.

9.56 PRIMARY

PRIMARY

Apply primary beam correction to an interferometer map.

9.57 PSC

PSC (Point Source Catalog)

This is a general utility to look through the compacted IRAS Point Source Catalog. This program is able to select sources according to various criteria and to create binary tables and/or formatted files containing the information about the sources. Even on tape, the program is fast to select sources (less than 8 minutes in any small positional box, and often less). The time needed to list the information may be longer if you select many sources..

It requests as input a compacted file containing the full catalog information and a table containing pointers to this compacted file, as well as positional and flux information. The compacted catalog can be either on magnetic tape (GAG242 at the Groupe d'Astrophysique in Grenoble), or on disk (with logical name PSC_COMPACT). The input table must be on disk. It can be either the default table IRAS_PSC_EQU (full catalog) or any previous output of the program (to work on a reduced source list).

The task produces as output a formatted file and/or an output table. A description of the formatted output can be obtained in GRAPHIC or OVERLAY by HELP PSC_IRAS FORMAT. Reference to the IRAS explanatory supplement will probably be needed except for IRAS wizards. The output table has the same content as the input one (but of course only for selected sources) and can therefore be used as input to another run of the program: as an example, this will allow to select first on the 12/25 color and then on the 60/100 color. Since this table is a standard GreG table, it can also be used in GreG's COLUMN command to produce plots. Column 1 is Right Ascension (in radians), column 2 is Declination (also in radians). Columns 3 to 6 contain the 4 fluxes (12 to 100), with upper limits coded as negative. Column 7 is an internal pointer. This can be used for example, to plot the distribution on the sky of a given class of sources. After some simple processing (in SIC), color/color plots can easily be produced.

9.58 REGRESSION

REGRESSION

This task computes the best least square linear regression between two images. It uses as input a cross histogram of the two images which must have been created previously by program HISTO_CROSS.

9.59 REPROJECT

REPROJECT

This task resamples an input image to a different projection and coordinate system. Two interpolation methods are available. The SLOW one takes properly the blanking value into account, the FAST one ignores it. Although it is usually much faster than the SLOW one, it may be slower in a few cases (large output map and small input map with change of coordinate system essentially). The task works on data cubes,

```
processing them plane by plane.

CAUTION:  This task uses an  interpolation  method,  hence  the   output
image increment should be SMALLER than the input image increment.

See also the GRID_PROJECT task if you start from CLASS data.
```

9.60 RESTORE

```
RESTORE

The most frequently-used method for making single-beam maps from dual-
beam data is implemented in RESTORE. It uses the Emerson-Klein-Haslam
algorithm, a two-step process, in which the two-dimensional dual-beam
map of the source is treated as a set of separate one-dimensional
strips, so-called subscans.  The following is executed:

   - the dual-beam maps obtained via MAKPLAN are convolved with a
     function which filters out spatial frequencies of wavelength
     equal to multiples of the double-beam spacing, and
   - the double-beam maps are convolved using a discrete up-and-down
     comb-like restoring function to deconvolve the maps of the
     dual-beam response due to the wobbling.

The first step is executed if the relative gain of the ON to OFF beams
is either equal to plus or to minus one (UNBAL), the second one is exe-
cuted if RESTORE_FILTER is true.  Finally, if variable RESTORE_MASK is
true, RESTORE restricts both steps to a region within a polygonal sup-
port defined via the SUPPORT command. Flagged channels are not reduced.
```

! Task RGBTOPS !

9.61 RGBTOPS

```
RGBTOPS

This task takes as input three pixel (8-bits bytes)  maps,  created  for
example  with  the ALL\SAVE command in OVERLAY or the GFITS\READ command
in GFITS (provided the FITS is on  8  bits),  and  merges  them  into  a
PostScript  TrueColor  file  for further printing. This is a dirty patch
since TrueColor maps should be supported in OVERLAY in a  more  friendly
manner.  The program asks for the names of the RED, GREEN and BLUE maps,
in this order, that will  be  merged  in  a  "truecolor"  image  on  the
PostScript printer.
```

9.62 SHIFT

```
SHIFT

Shift an image by Fourier transform,  phase shifting and back  transform
to image plane. NOT DEBUGGED ?
```

9.63 SHIFT_AND_ADD

SHIFT_AND_ADD

This task is recommended when there is a clear-cut separation of the ON and OFF images. This is the case of many filamentary structures, edge-on galaxies, and all object for which wobbling in-the-object can a priori be avoided. Contrary to RESTORE the single-beam maps are computed directly from variable SIGNAL (the calibrated temporal sequence of dual-beam data) and NOT from the dual-beam maps obtained via MAKPLAN. The position variable SCAN_COORD is used to resample the data variable SIGNAL on an azimuth-elevation grid G. The signal is displaced by half the effective ON to OFF beam separation BEAMSEP once towards positive azimuths and once to negative azimuths, is gridded line by line (subscan by subscan), apodized in azimuth to the frequency cutoff of the telescope using the envelope of a sinc function tapered over a gliding window F and weighted according to the inverse density factor W required to obtain uniform data sampling. The size of the gliding window is chosen to be 5x the beam-width at the effective observing frequency. The weighting factor W is introduced to control the form of the beam, and its value is related to the area density of the data at any point j in the new grid:

```
                 --
            Wj = \    sin (uj)/(uj).Tj
                 /
                 -- F

        uj = 2.A.pi.D/lambda.(j-SCAN_COORD)
```

where T is an optional apodization function and A is the ratio of the halfpower width of the gridding function W to the beamwidth of the telescope. T can either be truncated-gaussian (by default), parabolic, linear or equal to unity (no taper). So far, the i-th point of a specific line in the single-beam map G can be written to:

```
             --                            --
  Gi = 1/Wi.(\ SIGNAL.sin (uj)/(uj).Tj + \ SIGNAL.sin (uk)/(uk).Tk)
             /                            /
             -- F                         -- F'
```

However, depending on variable SAA_MASK two different processes are executed:

- SAA_MASK is set to NO: only data inside a window whose limits are defined by half the beam separation from either side of a line of position angle PA and whose azimuthal offset from the tracking center of the telescope is given in OFFSET, is retained.
- SAA_MASK is set to YES: only data inside a polygonal support defined via SUPPORT is retained. Note, however, that the boundaries of the support region are equally clipped in each line (subscan) to the effective separation of the ON and OFF beams.

Finally, the retained data is written to a file with the extension .saa.

Apodization in elevation is done according to the setting of the logical SMOOTH_EL. The grid cells are by default quadratic and their size is set either to the sampling interval in azimuth or in elevation, whichever is larger. However, different sizes in both directions can be set using variable MAP_CELL. As for RESTORE compute UNBAL from the mean sign inverted ratio of the ON to OFF beam amplitudes. Flagged channels are not reduced.

9.64 SLICE

SLICE

This task extracts a slice from a 3-D Ra-Dec-Velocity cube. The output is a 2-Dimensional Position-Velocity plot.

9.65 SMOOTH

SMOOTH

Smooths an input map using various methods, all based upon a convolving kernel of 5 points large only. This method is crude, but very fast, and supports blanking in principle... Data cubes are processed plane by plane.

1. BOX A simple 5 by 5 boxcar smoothing. This is a very strong smoothing...
2. GAUSS A gaussian smoothing. The gaussian is sampled on a 5 by 5 grid so it must not be too broad. 3 pixels seem a maximum. Use task GAUSS_SMOOTH if you want to smooth with a broader gaussian, but beware it does not handle properly blanked pixels.
3. HANNING Smoothing by a 5 by 5 pyramid, with weights 3 2 1.
4. USER User defined smoothing coefficients on a 5 by 5 grid, assuming biaxial symmetry. Hence 6 coefficients only are required, S00 S10 S11 S20 S21 S22. The corresponding smoothing kernel is :

   ```
   S22 S21 S20 S21 S22
   S21 S11 S10 S11 S21
   S20 S10 S00 S10 S20
   S21 S11 S10 S11 S21
   S22 S21 S20 S21 S22
   ```

   Such a generalised "smoothing" enables some image enhancement based on gradients (e.g. with weights 4 -1 0 0 0 0). The "smoothed" image is normalised by the sum of absolute values of the coefficients.

9.66 SORT

SORT

This program sorts all columns of a given table by ascending order of one of its columns. The same table is used for both input and output. Only single precision real tables are accepted by the present version.

9.67 SPECTRUM

SPECTRUM

This task extracts a spectrum from a 3-D Ra-Dec-Velocity cube. The output is a table with 2 columns. Column 1 contains the velocities, Column 2 the spectrum values.

9.68 SPECTRUM_SUM

SPECTRUM_SUM

This task computes a integrated spectrum from a 3-D Ra-Dec-Velocity cube, over an area specified by an arbitrary polygon. The output is a table with 2 columns. Column 1 contains the velocities, Column 2 the spectrum values.

9.69 SUM

SUM

This program is used to sum many images, handling a weight image in order to produce the average image later on.

```
         It computes     X(i,j) = X(i,j) + F * Z(i,j)
         and             Y(i,j) = Y(i,j) + F
```

for all non blanked pixels of the input image Z. When called repetitively with the same input/output images X and Y, X contains the weighted sum of all Z images, and Y the weight of each pixels. A parameter is provided to initialise the process.

The last step to obtain the average image is to divide X by Y using program COMBINE with option DIVIDE. You can also use SIC command LET to do so, but SUM and COMBINE will handle properly blanking values.

This is a quick reference about all programs available under the GILDAS package. It may not be up to date, and not all listed programs are fully debugged.

For convenience, this is a thematic summary in which some algorithms may appear more than one time.

* GFITS *	* GRAPHIC *	* MAPPING *	* VECTOR *
BACKGROUND	BLANKING	CIRCLE	CLEAN-C
CLEAN	COMBINE	COMPRESS	CONVERT
CORRELATE	CV_SMOOTH	DG_SMOOTH	EXAMPLE
EXAMPLE1	EXPAND	EXTRACT	FIELD_FIND
FIELD_LIST	FIELD_STAT	FILL_CUBE	FITS_GILDAS
FLOW	FOURIER	GAUSS_1D	GAUSS_2D
GAUSS_SMOOTH	GILDAS_FITS	GRID_CUBE	GRID_EXTEND
GRID_PROJECT	GRID_SG	HEADER	HISTO_CLOUD

HISTO_CROSS	HISTO_DOUBLE	HISTO_SIMPLE	HISTO_TABLE
IMAGE	INTERPOLATE	LIST	MAKE_BACK
MAKE_CUBE	MAKE_MOSAIC	MAKPLAN	MANDEL
MANDEL_REAL	MAP_AVER	MAP_COMPRESS	MAP_EXPAND
MAP_INTER	MASK	MERGE	MINIMIZE
MIX-C	MOSAIC	MOSAIC_CLEAN	MX
NOISE_SMOOTH	ONOFF	OVERLAY	PLANE
PLANE_OLD	PRIMARY	PSC	REGRESSION
REGUL	REPROJECT	RESTORE	RGBTOPS
SELF_CAL	SHIFT	SMOOTH	SORT
SUM	SWAP	TABLE	TRANSPOSE
TRUE_COLOR	UV_ASCAL	UV_AVERAGE	UV_CENTER
UV_CIRCLE	UV_CLIP	UV_COMPRESS	UV_CRASH
UV_DFT	UV_EXTRACT	UV_FIT-OLD	UV_FIT
UV_FITC	UV_FLAG	UV_LIST	UV_MAP
UV_MERGE	UV_MODEL	UV_NOISE	UV_SELFCAL
UV_SHIFT	UV_SHORT	UV_SINGLE	UV_SUBTRACT
UV_ZERO	UVMAP	UVSORT	

To access the following subtopic, type (under UNIX) "EXPLAIN SUMMARY subtopic"

9.70 AVAILABLE_TASKS

BACKGROUND	Compute background images
BLANKING	Change the blanking value
CIRCLE	Make the circular average of an image
CLEAN-C	?
CLEAN	Cleaning (of data cubes).
COMBINE	Combine two input images in many ways
COMPRESS	?
CONVERT	NIC task.
CORRELATE	Compute correlation image of two images.
CV_SMOOTH	Cross validation smoothing
DG_SMOOTH	Conjugate Gradient smoothing
EXAMPLE	A simple example for input parameters
EXPAND	?
EXTRACT	Extract a subset of an image to fill a subset of an image
FIELD_FIND	Find the fields in an image
FIELD_LIST	Find parameters of fields in an image
FILL_CUBE	Resample a cube by Random or Regular interpolation
FITS_GILDAS	Single Image disk-FITS to GILDAS translator
FLOW	A bipolar flow hunter !...
FOURIER	Compute complex fourier transform of an image
GAUSS_1D	Multi component one-dimension gaussian fitting
GAUSS_2D	Single component two-dimension gaussian fitting
GAUSS_SMOOTH	Smooth by a gaussian in the fourier plane
GILDAS_FITS	Single image GILDAS to disk-FITS translator
GRID_CUBE	A simple gridding task to make an image from a table.
GRID_EXTEND	A gridding task to make an image from a table.
GRID_PROJECT	A task to reproject using convolution.
GRID_SG	A gridding task to make an image from a table.
HEADER	List and modify an image header
HISTO_CLOUD	Cross histogram of two input images (Table output)

HISTO_CROSS	Cross histogram of two input images (Image output)
HISTO_DOUBLE	Histogram of an image as a function of another
HISTO_SIMPLE	Histogram of an image
HISTO_TABLE	Cross histogram of column tables
IMAGE	Make an image from an RGDATA-like file
INTERPOLATE	Resample and smooth an image along one direction
LIST	List a table
MAKE_BACK	Compute a background image
MAKE_CUBE	Image reconstruction from uncomplete sample
MAKE_MOSAIC	Build a mosaic for deconvolution by MAPPING
MAKPLAN	NIC task
MANDEL	Mandelbrot ensemble (integer version)
MANDEL_REAL	Mandelbrot ensemble (real version)
MAP_AVER	Average plans of a 3-D data cube.
MAP_COMPRESS	Compress images (by Fourier truncation)
MAP_EXPAND	Expand images (by Fourier expansion)
MAP_INTER	Resample data cubes along 3rd axis.
MASK	Mask part of cube
MERGE	Merging of Tables
MINIMIZE	Find best linear correlation between two images
MIX-C	?
MOSAIC	?
MOSAIC_CLEAN	?
MX	UV plane-based CLEAN technique
NOISE_SMOOTH	Noise smoothing enhancement
ONOFF	NIC Task
OVERLAY	?
PLANE	Compute the plane going through three points
PLANE_OLD	Older version of PLANE
PRIMARY	Correct for primary beam attenuation.
PSC	To deal with the IRAS Point Source Catalog
REGRESSION	Compute the linear regression from a cross histogram
REGUL	A regularisation deconvolution method
REPROJECT	Reproject an image in a different system
RESTORE	NIC Task
RGBTOPS	Convert RGB images to PostScript
SELF_CAL	Self-Calibrate (in some way...) a UV data set
SHIFT	?
SMOOTH	General pattern smoothing algorithm
SORT	Sort a table (real version)
SUM	Add a large number of images
SWAP	Rotate or make a mirrored image of an input image
TABLE	Make a Table from a formatted file
TRANSPOSE	Transpose images
TRUE_COLOR	To produce true color images of (bipolar) flows.
UV_ASCAL	Self-calibrate (in some way...) a UV data set
UV_AVERAGE	Average several channels of a UV table
UV_CENTER	?
UV_CIRCLE	?
UV_CLIP	Clip UV data
UV_COMPRESS	Compress UV data by spectral channel averaging
UV_CRASH	N.A.
UV_DFT	Compute an image using Direct Fourier Transform (no FFT)
UV_EXTRACT	Extract some planes from UV data

```
UV_FIT        Fit simple functions into UV data
UV_FITC       Fit centered functions into UV data
UV_FLAG       Flag UV data
UV_LIST       List UV data
UV_MAP        Makes map from UV data using gridding + FFT
UV_MERGE      Merge UV data sets
UV_MODEL      Computes UV data from input model.
UV_NOISE      ?
UV_SELFCAL    Self-calibrate (in some way...) a UV data set
UV_SHIFT      Shift phase center of UV data
UV_SHORT      Compute short spacing from single-dish (obsolete)
UV_SINGLE     Compute short spacing from single-dish (recommended version)
UV_ZERO       Enter zero spacing
UVSORT        Precess and sort raw UV tables
* VECTOR *    To activate GILDAS tasks.
```

9.70.1 ASTRONOMICAL_PROCESSING

```
PSC           To deal with the IRAS Point Source Catalog
REPROJECT     Reproject an image in a different system
FLOW          A bipolar flow line-wing finding program.
TRUE_COLOR    Produces true color images of bipolar flows.
```

9.70.2 CORRELATION_ANALYSIS

```
CORRELATE     Computes correlation image of two images.
HISTO_CLOUD   Cross histogram of two input images (Table output)
HISTO_CROSS   Cross histogram of two input images (Image output)
HISTO_DOUBLE  Histogram of an image as a function of another
HISTO_SIMPLE  Histogram of an image
HISTO_TABLE   Cross histogram of column tables
MINIMIZE      Find best linear correlation between two images
REGRESSION    Compute the linear regression from a cross histogram
```

9.70.3 DISPLAY

```
* GRAPHIC *   GRAPHIC program. Uses GreG for contouring.
* MAPPING *   Deconvolution program. Includes GRAPHIC.
Both programs include VECTOR, and thus can RUN GILDAS tasks.
```

9.70.4 IMAGE_ANALYSIS

```
    BACKGROUND    Compute background images
CIRCLE        Make the circular average of an image
COMBINE       Combine two input images in many ways
FIELD_FIND    Find the fields in an image
FIELD_LIST    Find parameters of fields in an image
PLANE         Compute the plane going through three points
```

9.70.5 IMAGE_CONSTRUCTION

```
BLANKING      Change the blanking value
CLEAN         Cleaning (of data cubes).
UV_DFT        Slow Fourier Transform from UV to map (cubes)
EXTRACT       Extract a subset of an image to fill a subset of an image
FILL_CUBE     Resample a cube by Random or Regular interpolation
FITS_GILDAS   Single image disk-FITS to GILDAS translator
* GFITS *     FITS (disk or tape) to|from GILDAS format translator
GILDAS_FITS   Single image GILDAS to disk-FITS translator
GRID_CUBE     A gridding task to make an image from a table.
GRID_EXTEND   A gridding task to make an image from a table.
HEADER        List and modify an image header
MAKE_CUBE     Image reconstruction for incomplete, noisy data.
MAP_AVER      Average plans of a 3-D data cube.
MAP_COMPRESS  Compress images (by Fourier truncation)
MAP_EXPAND    Expand images (by Fourier expansion)
MAP_INTER     Resample data cubes along 3rd axis.
MASK          Mask part of an image.
SUM           Add a large number of images
SWAP          Rotate or make a mirrored image of an input image
UV_MAP        Make maps (cubes) from UV data using gridding + FFT
TRANSPOSE     Transpose images
```

9.70.6 MODEL_FITTING

```
GAUSS_1D      Multi component one-dimension gaussian fitting
GAUSS_2D      Single component two-dimension gaussian fitting
REGRESSION    Compute the linear regression from a cross histogram
UV_FIT        Fit simple functions into UV data
UV_FITC       Fit centered functions into UV data
```

9.70.7 SMOOTHING

```
CV_SMOOTH     Cross Validation smoothing
DG_SMOOTH     Conjugate Gradient smoothing
GAUSS_SMOOTH  Smooth by a gaussian in the Fourier plane
INTERPOLATE   Resample and/or smooth an image along first dimension
NOISE_SMOOTH  Noise smoothing enhancement
SMOOTH        General (small) pattern smoothing algorithm
```

9.70.8 TABLE_PROCESSING

```
HISTO_TABLE   Cross histogram of column of a table
LIST          List a table
MERGE         Merging of Tables
SORT          Sort a table (real version)
TABLE         Make a Table from a formatted file
```

UVSORT	Precess and Sort a raw UV table

9.70.9 UV_PROCESSING

UV_ASCAL	Self-calibrate (in some way...) a UV data set
UV_AVERAGE	Average several channels of a UV table
UV_CENTER	?
UV_CIRCLE	?
UV_CLIP	Clip UV data
UV_COMPRESS	Compress UV data by spectral channel averaging
UV_CRASH	N.A.
UV_DFT	Compute an image using Direct Fourier Transform (no FFT)
UV_EXTRACT	Extract some planes from UV data
UV_FIT	Fit simple functions into UV data
UV_FITC	Fit centered functions into UV data
UV_FLAG	Flag UV data
UV_LIST	List UV data
UV_MAP	Makes map from UV data using gridding + FFT
UV_MERGE	Merge UV data sets
UV_MODEL	Computes UV data from input model.
UV_NOISE	?
UV_SELFCAL	Self-calibrate (in some way...) a UV data set
UV_SHIFT	Shift phase center of UV data
UV_SHORT	Compute short spacing from single-dish (obsolete)
UV_SINGLE	Compute short spacing from single-dish (recommended version)
UV_ZERO	Enter zero spacing
UVSORT	Precess and sort raw UV tables

9.71 SWAP

SWAP

This task swaps an image with respect to first or second axis (e.g. along the X dimension, pixel 1 becomes pixel NX, 2 becomes NX-1 and so on). This operation is never required for most algorithms because they work in the "User Coordinate" space (such as the contouring command in GRAPHIC...). It is only needed for a few algorithms which work in the pixel space (such as HISTO_CROSS). Pixel order can be reversed along X (first axis) or Y (second axis). The program does not handle data cubes at present..

9.72 TABLE

TABLE

This task produces a GILDAS Table from a formatted listing. The table format is recommended if you intend to use your data more than a few (4) times, i.e. in virtually any case. Moreover, the table format can be processed by more programs than the list-directed one. You should set the number of columns exactly, and the number of lines to 0 to allow the program to compute the table size, or to the real number of lines in your formatted file. Never set the number of lines or the number of columns larger than really present or the program will crash.

Task LIST does the reverse process.

9.73 TRANSPOSE

    TRANSPOSE

A non-general simple-minded routine to tranpose data cubes (or images). The only transposition codes it recognises currently are 312, 231 and 213 if I remember well.

Usually command VECTOR\TRANSPOSE Input Output Order is preferred, except possibly for large images.

9.74 TRUE_COLOR

    TRUE_COLOR

This is a dedicated routine to produce bipolar outflow maps in "true" color. It takes as input a cube (N by NX by NY) with the velocity along the first axis, and produces a pseudo cube (3 by NX by NY) containing the integrated intensities in the blue and red lobes in planes 1 2, and a specially encoded map in plane 3. The latter must be displayed in programs OVERLAY or DISPLAY with a special color Look Up Table as defined below

```
ALL\SELECT RANGE 1 225              ! Range of input values
ALL\SELECT LUT RGB                  ! Select Red Green Blue mode
SIC\LET RED[I] (I-1)|225            ! Red look up table
SIC\LET BLUE[I] (MOD(I+13,15)+1)|15.5 ! Blue look up table
SIC\LET BLUE[1] 0                   ! Index 1 is for background
SIC\LET GREEN 0                     ! No green
ALL\LUT
```

9.75 UV_ASCAL

    UV_ASCAL: "Self"-Calibration of a UV data set

UV_ASCAL does a Phase and/or Amplitude Referencing of a UV Data set, using another one representing a point source observed quasi-simultaneously. It is STRONGLY ADVISED to self-calibrate the phase first, then the amplitude using a longer smoothing time. UV_ASCAL allows in addition to subtract (part of) the input reference source, typically to provide continuum-free spectral line maps.

See also UV_SELFCAL.

9.76 UV_AVERAGE

    UV_AVERAGE: Channel averaging of UV data

```
Averages channels from CHANNEL$[1] ro CHANNEL$[2] of a UV data set
to produce a pseudo-continuum (single-channel) output UV table.
(Similar to UV_COMPRESS)
```

9.77 UV_CENTER

```
    UV_CENTER

Find the centroid of (centro-symmetric) emission from a UV table.
```

9.78 UV_CLIP

```
    UV_CLIP

Clip UV data to suppress grossly discrepant values.  All channels are
flagged if any channel between the first and last considered
(specified by CHANNELS$) has an amplitude greater than VCLIP$.  Clipped
channels are set to zero amplitude and zero weight.
```

9.79 UV_COMPRESS

```
    UV_COMPRESS:  Channel averaging of UV data

Averages NC$ channels of a UV data set to produce a smaller,
spectrally smoothed one.
```

9.80 UV_DFT

```
    UV_DFT :  map making from UV data

UV_DFT makes a map from UV data using (slow) Fourier Transform.  This
method introduces no aliasing, and no grid correction, but it is slow.

This task processes all table channels at a time.
```

9.81 UV_EXTRACT

```
    UV_EXTRACT :  Extract channels from a UV data set

Extract the spectral line channels specified by CHANNELS$ from a
bigger UV data set.
```

9.82 UV_FIT-OLD

```
    UV_FIT-OLD :  Fit spherically symmetric models to a UV data set

This task allows fitting of simple models directly from the
visibilities. The models are linear combinations of 2 radial
distributions, assumed to be centered on the current phase tracking
center. Use tasks UV_CENTER and UV_SHIFT if necessary before fitting.


OLD VERSION: there is a sign error....
```

```
The type of the first radial distribution. Available types are
    NONE      No function to fit (for 2nd function usually)
    POINT     Point source
    GAUSS     Gaussian sources
    DISK      Disk
    RING      Annulus
    EXPONENT  Exponential brightness
    POWER-1   B = 1/r
    POWER-2   B = 1/r^2
    POWER-3   B = 1/r^3
    PARABOLA  Parabolic brightness distribution
```

9.83 UV_FIT

```
    UV_FIT :  Fit point and centro-symmetric models to a UV table

This task allows fitting of models directly from the  visibilities.  The
models  are  either  simple or linear combinations of several functions.
The results of the fitting process are the position offsets in R.A.  and
Dec  (arc  seconds) of the model source from the phase reference center,
and its flux (Janskys).  Depending on the fitting  functions  additional
fit   results   are  possible.  Currently  supported  distributions  and
additional fit parameters are:

POINT     Point source             : None
E_GAUSS   Elliptic Gaussian source  : FWHP Axis (Major and Minor), Pos Ang
C_GAUSS   Circular Gaussian source  : FWHP Axis
C_DISK    Circular Disk            : Diameter
E_DISK    Elliptical (inclined) Disk: Axis (Major and Minor), Pos Ang
RING      Annulus                  : Diameter (Inner and Outer)
EXPO      Exponential brightness    : FWHP Axis
POWER-2   B = 1/r^2                : FWHP Axis
POWER-3   B = 1/r^3                : FWHP Axis
```

9.84 UV_FITC

```
    UV_FITC :  Fit spherically symmetric models to a UV data set

This  task  allows  fitting  of  simple   models   directly   from   the
visibilities.   The   models   are   linear  combinations  of  2  radial
distributions, assumed to be centered  on  the  current  phase  tracking
center. Use tasks UV_CENTER and UV_SHIFT if necessary before fitting.


OLD VERSION: there is a sign error....


The type of the first radial distribution. Available types are
    NONE      No function to fit (for 2nd function usually)
    POINT     Point source
    GAUSS     Gaussian sources
    DISK      Disk
```

```
RING      Annulus
EXPONENT  Exponential brightness
POWER-1   B = 1/r
POWER-2   B = 1/r^2
POWER-3   B = 1/r^3
PARABOLA  Parabolic brightness distribution
```

9.85 UV_FLAG

```
UV_FLAG: Flag or unflag data in a UV data set
```

```
Flag or unflag a specified time range in a UV data set, for one  or  all
baseline and a specified channel range.
```

9.86 UV_LIST

```
UV_LIST: List part of a UV data set
```

```
This task list one channel of a UV data set, by specified time steps.
```

9.87 UV_MAP

```
UV_MAP :  Fast map making from UV data, using FFT
```

```
UV_MAP makes a map from UV data by gridding  the   UV   data   using   a
convolving  function,  and then Fast Fourier Transforming the individual
channels.  UV_MAP can produce one beam per channel or a common beam  for
all  channels.  In the latter case, the gridding is done only once, thus
neglecting  frequency change between channels; this is  of  course  much
faster.

UV_MAP allows to shift the map center and rotate the image, by  shifting
the  phase  tracking center and rotating the UV coordinates of the input
UV table.

UV_MAP produces an output LMV cube (X, Y, Velocity) and a BEAM image  or
cube  from  the input  UV  table.  UV_MAP is an optimized version of the
older UVMAP.
```

9.88 UV_MERGE

```
UV_MERGE
```

```
This task merges together two tables of UV  data,  to  form   a   single
output  table.   The  two  input  tables  must  have  the  same spectral
characteristics (not checked).

Multiplicative factors may be applied  to  both   the   amplitudes   and
weights of both tables.
```

9.89 UV_MODEL

```
    UV_MODEL

Compute a UV table from an input image (3-D data cube are allowed) and a
reference UV table used to specify the UV sampling.

Related task UV_TABLE will do the same for  a  regular  output  UV  grid
rather than on the sampling of a template table.
```

9.90 UV_SELFCAL

```
    UV_SELFCAL:  "Self"-Calibration of a UV data set

UV_SELFCAL does a Phase and/or Amplitude Referencing of a  UV  Data set,
using    another   one   representing   a   gaussian   source   observed
quasi-simultaneously.  It is  STRONGLY  ADVISED  to  self-calibrate  the
phase   first,  then  the  amplitude  using  a  longer  smoothing  time.
UV_SELFCAL allows in addition to subtract (part of) the input  reference
source, typically to provide continuum-free spectral line maps.
```

9.91 UV_SHIFT

```
    UV_SHIFT

This task changes the phase tracking center of a UV table, by applying a
phase shift on the data. The coordinate system can also be rotated using
the variable ANGLE$. Rotation is easier to  apply  in  UV  tables  since
rotation  and  Fourier  transform  commute: only the U and V coordinates
have to be rotated.
```

9.92 UV_SHORT

```
    UV_SHORT

This task prepares a UV table of short spacings from a  single-dish map.
This  table  may  later  be  merged  to an interferometer UV table.  The
operations performed are:
  -  Fourier transform of the single dish map
  -  Division by the Fourier transform of the single dish  beam,  up  to
    a maximum spacing (SD_DIAM$, in meters)
  -  Inverse Fourier transform to  the  image  plane
  -   Multiplication  of  the   image   by  the  primary  beam  of   the
    interferometer elements
  -  Fourier transform back to the UV plane.
  -  Creation of the UV table, with a given  weight  SD_WEIGTH$  and  an
    appropriate calibration factor to Janskys SD_FACTOR$

Both the single-dish and the interferometer antennas are assumed to have
gaussian beams (SD_BEAM$ and IP_BEAM$, in radians).
```

9.93 UV_SINGLE

```
    UV_SINGLE

This task prepares one or more (for mosaics) UV tables of short spacings
from  a   single-dish  table  of  spectra  (as produced by CLASS command
GRID).  This table may later be merged to an  interferometer  UV  table.
Two major steps are performed for this:
  - Creation of a "well behaved" map from the spectra.
  - Extraction of UV spacings from this map.

The "well behaved" map is obtained by linear  operations  (convolutions)
from  the original spectra, in an optimum way from signal to noise point
of view. The map is extrapolated smoothly towards zero at the  map  edge
in  order  to avoid further aliasing in the Fourier transform operations
required in step 2. This extrapolation has a scale length of  twice  the
single-dish beam, in order to avoid spurious Fourier components.

To obtain it, UV_SINGLE performs the following operations:
  -  Resampling (in space) of the original spectra on a regular grid  by
    convolution  with  a  small  (typically 1/4 of the single-dish beam)
    gaussian  convolving  kernel.  In  this  process,  the  weights   of
    individual spectra is carried on a weight map.
  -  Extrapolation by zero outside the convex hull of the mapped region
  -  Convolution of the result by  a  gaussian  twice  as  wide  as  the
    single-dish beam.
  -  Within the convex hull of the mapped region, the  smoothed  map  is
    replaced by the original map.

From this map, UV_SINGLE computes one or several UV tables (one for each
pointing center in case of a mosaic) in the following way:
  -  Fourier transform of the single dish map
  -  Division by the Fourier transform of the single dish  beam,  up  to
    a maximum spacing (SD_DIAM$, in meters)
  -  Inverse Fourier transform to  the  image  plane
and then for each pointing center
  -   Multiplication  of  the   image   by  the  primary  beam  of   the
    interferometer elements
  -  Fourier transform back to the UV plane.
  -  Creation of the UV table, with a given  weight  SD_WEIGTH$  and  an
    appropriate calibration factor to Janskys SD_FACTOR$

Both the single-dish and the interferometer antennas are assumed to have
gaussian beams (SD_BEAM$ and IP_BEAM$, in radians).
```

9.94 UV_STAT

```
    UV_STAT :  Analyse a UV table for weighting parameters and map  cell
size.

UV_STAT analyses a UV table from the weighting point of view to evaluate
the  beam  size  and  the  expected  rms  noise as function of Weighting
parameters (UV_CELL, mode  WEIGHT)  or  Taper  (UV_TAPER,  mode  TAPER).
```

UV_STAT computes dirty beams for various tapers or "Robust weighting" parameters, and evaluate the expected rms noise for each of them.

9.95 UV_SUBTRACT

UV_SUBTRACT: "Self"-Calibration of a UV data set

UV_SUBTRACT subtracts a time average continuum UV table from a spectral line one, typically to provide continuum-free spectral line maps. Caution: it will not work if the continuum source is too complex, because of the time averaging.

9.96 UV_TABLE

UV_TABLE

Compute a UV table from an input image (3-D data cube are allowed). It differs from UV_MODEL in producing UV data on a critically sampled rectangular grid rather than along user specified tracks. It is typically used to analyse image plane observational data with UV tools.

9.97 UV_ZERO

UV_ZERO

This task appends zero-spacing (single-dish) data (spectrum and/or continuum flux) to an existing UV table. If zero-spacing data already exists in the UV table, they are replaced by the new data. The zero spacing spectrum is a GILDAS table, which should be created in CLASS by the command GREG. Additional inputs are: the weight to be used, an amplitude calibration factor affecting the spectrum, and a continuum flux.

The single-dish spectrum is resampled to the UV table spectral characteristics.

9.98 UVSORT

UVSORT

Sort a raw UV data set to reorder UV data in ascending, negative V values for further processing using UVMAP. In the future, will also apply differential precession which is currently neglected.

Index