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

a GILDAS working group software

February 7, 2002

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

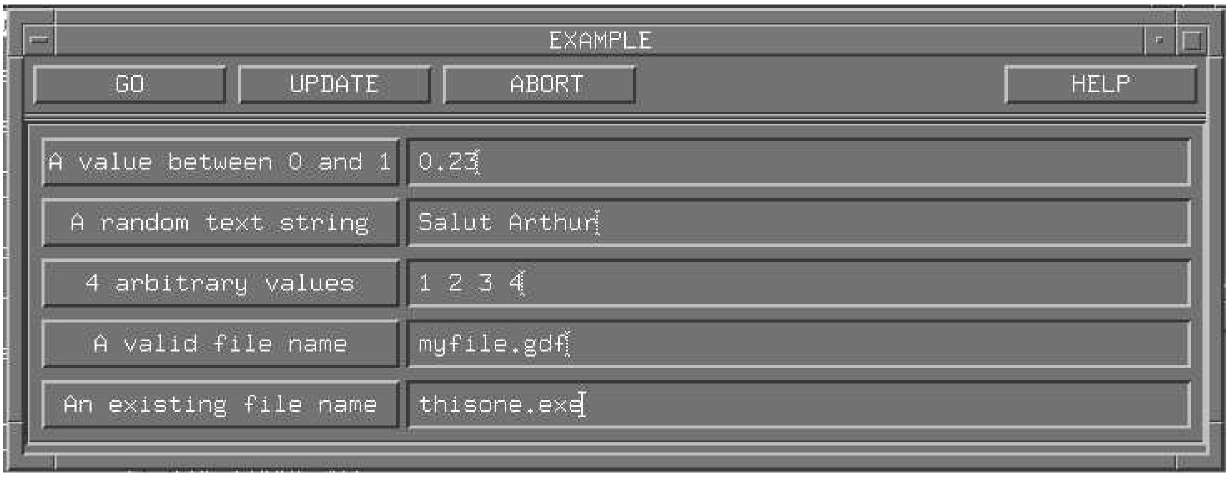
"Nobody is perfect"
(attributed to the nameless God)

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.



A value between 0 and 1 0.23
A random text string Salut Arthur
4 arbitrary values 1 2 3 4
A valid file name myfile.gdf
An existing file name thisone.exe

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 Language

```
EXPLAIN [Task] : Gives explanation about a GILDAS task
RUN Task       : Activate a GILDAS task
SPY [Task]     : Look at current status of detached Tasks
SUBMIT Task    : Submit a GILDAS Task in batch queue GILDAS_BATCH
TRANSPOSE      : Transpose data cubes
WAIT [Task]    : Wait for Task completion
```

3.3 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.
```

3.4 SPY

```
        [VECTOR\]SPY [Task_Name]

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

3.5 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.
```

3.6 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
```

3.7 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**Ṫhis 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 Language

```
    GRAPHIC\ Language Summary
ASTROMETRIC      : Plots astrometric star positions (for finding charts)
HEADER           : Give the Header of the current Image.
IMAGE [Filename]: Read the GDF map Filename. Default extension is .GDF
IRAS Band        : Select and plot IRAS sources of a given band
KILL             : Kills pixels.
PSC_IRAS         : Get characteristics of IRAS point source
REGRESSION [Val]: Computes regression lines
SPECTRUM         : Extract a spectrum from an image (to plot it).
```

5.8 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).
```

5.8.1 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
```

5.9 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.
```

5.10 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 Language

```
                    FITS\ Language Summary
            FITS to GILDAS image data format translator

This program converts GILDAS images on disk to FITS files on tape or  on
disk,  and  vice  versa.  Switching from tape to disk mode is done using
command FILE, while switching from disk  to  tape  mode  is  done  using
command MOUNT.

DISMOUNT        : Logically dismounts the tape
INIT            : Initializes the tape
HEADER          : Lists the next header
IMAGE Name      : Defines the file name for the next READ or WRITE.
LIST [Keyword]  : List headers of all files from current position
MOUNT           : Logically mounts the tape
READ            : Reads the current file to create a new GDF image
REWIND          : Rewinds the tape
SKIP N          : Skips N files on tape
SET Arg         : Customize the GFITS reading program.
WRITE           : Writes the current image at end of tape.
```

7.6 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...

7.7 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).

7.8 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.
```

7.8.1 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.
```

7.8.2 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.
```

7.9 REWIND

```
    REWIND
```

```
Rewind the tape
```

7.10 SET

```
    SET Argument Value
```

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

7.10.1 SET BLANKING

```
    SET BLANKING Breal
```

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

7.10.2 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.
```

7.10.3 SET NBITS

```
SET NBITS Value
```

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

7.10.4 SET NOPROMPT

```
SET PROMPT or SET NOPROMPT
```

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

7.10.5 SET PROMPT

```
SET PROMPT or SET NOPROMPT
```

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

7.10.6 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.
```

7.11 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.
```

7.12 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.

Index