A quick introduction to GILDAS

Last revised in October 2008

Questions? Comments? Bug reports? Mail to: `gildas@iram.fr`

The GILDAS team welcomes an acknowledgment in publications
using GILDAS software to reduce and/or analyze data.
Please use the following reference in your publications:
`http://www.iram.fr/IRAMFR/GILDAS`

Up-to-date information on credits and responsabilities
may be found on the GILDAS web page

Related information are available in

- SIC: Command Line Interpretor
- GREG: Graphical Possibilities
- ASTRO: Ephemeris and observation preparation
- MIRA: Calibration of the raw format (IMBFITS) and the writing of the CLASS data format of IRAM-30m spectra obtained in 2006 and after
- CLASS: Processing of single-dish spectra
- CLIC: Continuum and Line Interferometric Calibration
- MAPPING: Imaging and deconvolution of aperture synthesis data
- MIS: Millimeter Interferometry Simulation Tools
- Programming in GILDAS

Contents

1 Introduction **3**

2 Organisation **3**

3 Concepts and Useful Hints **4**
3.1 Images 4
3.2 Customizing 4
3.2.1 Logical Names 4
3.2.2 File Names 5
3.2.3 Initialisation Files 5

4 Getting Started **6**

5 The Data Format **7**
5.1 Description 7
5.2 Drawbacks 7

6 Communication with the Outer World **8**
6.1 From CLASS 8
6.2 From a Program 8
6.3 From other Packages: the ACCEPT command 8
6.4 From and To other Packages: FITS 9

1 Introduction

GILDAS is one of the numerous image processing systems used in Astronomy. GILDAS was born sometime ago in the Groupe d'Astrophysique de Grenoble (now LAOG, Laboratoire d'Astronomie de l'Observatoire de Grenoble), and has been adopted as the IRAM standard data reduction package. GILDAS is jointly maintained by IRAM & LAB (Laboratoire d'Astrophysique de Bordeaux). GILDAS contains many facilities, most of which are oriented towards spectral line mapping, or in fact all kind of 3-dimensional data.

GILDAS has grown on top of two pre-existing programs, CLASS and GREG, which seemed to please their users. As many other packages, GILDAS had to move out of its original VAX-VMS environment towards the growing Unix area. The original software was modified in order to run on VMS and Unix operating systems, with (near) transparent file sharing even between heterogeneous computers. The current implementation even allows remote task submission on a compute server from a workstation. VMS is no longer supported, but a Windows version is available and supported.

This guide is a general introduction to the GILDAS package. It does not describe any utility, but gives cross-reference to other documentations and many usefull hints on how to customize your GILDAS environment.

2 Organisation

GILDAS consists of five major parts :

- The documentations, a set of PostScript files. An HTML documentation is also available.
- Dedicated utilities, which are programs using the SIC (Sympathetic Interpretor of Commands) user-friendly interface. Each utility normally has its own manual, accessible through the general documentation system.
  - GREG an all purpose 1-D and 2-D graphic program
  - CLASS (Continuum and Line Analysis Single-dish Software), for single-dish data processing.
  - **ASTRO**, an astronomical tool, very useful to prepare an observing session, or for amateur astronomy...
  - **CLIC** (Continuum and Line Interferometric Calibration), to calibrate interferometer data from Plateau de Bure.
  - **Mapping**, an interactive imaging and deconvolution package.
- Dedicated small programs to perform non interactive time consuming processing : smoothing, transpositions, fitting, etc... These programs will be called "Tasks". They are not intended for interactive use (although you may do it if you are an expert), but require instead a monitor program to activate them.
- Additional applications programs, such as FLUX or POINT to handle pointing and monitoring data for the IRAM PdB array.

3 Concepts and Useful Hints

3.1 Images

Images and *Tables* are the two most useful concepts in GILDAS. Practically all data used by GILDAS are stored as *Images* (or *Tables*). An *Image* is a data file containing an array of up to 4 dimensions, and a small but comprehensive header to store the array dimensions, associated coordinates, etc... *Tables* are just 2-d *Images* with only the dimensions indicated in the header.

Images are used everywhere in GILDAS. The SIC command monitor directly manipulates *Images* through the `DEFINE IMAGE` command. The GREG program is able to display *Images* as contour plots with overlaid bitmaps. CLASS and **CLIC**, after working with their own data files, produce *Images* for further processing and display. And finally, all *Tasks* use *Images* for input and output.

Because of this importance of *Images*, we recommend the GILDAS users to read carefully the corresponding section in the SIC manual. Very efficient use of *Images* is possible within SIC, but it is also possible to do things a 1000 times more slowly. The ability of SIC to perform mathematics on *Images* can solve many problems, and avoid many "on-purpose" programs.

Images can even be initialized easily from external files of many different formats using the `ACCEPT` command of SIC, which allows the user to read in a totally flexible way data files to set the content of SIC variables.

Conversion from FITS format to *Images* is possible using several tools (because of the inherent complexities of the FITS format). Simple FITS files can even be automatically translated into *Images*.

3.2 Customizing

Although your system manager will provide reasonable defaults, you may wish to customize a few things to your own taste. Customizing the GILDAS environment can be done at two levels: Logical Names, and Initialization files for the SIC based utilities.

3.2.1 Logical Names

All logical names are placed in a file named `.gag.dico` in your home directory `($HOME)`. The format of the `.gag.dico` file is

```
LOGICAL_NAME1    equivalent_name1
LOGICAL_NAME2:   Equivalent_Name2
```

where the logical names should always be in capital letters, while case matter for equivalent names. The ``:'' indicates that the logical name is a directory (pathname). Besides logical names which you can use to define special files, or directories, a few peculiar names are used to customize your GILDAS environment:

- `GAG_EDIT`
  the name of your preferred text editor (e.g. vi, emacs, vuepad, ved, ...). With an & at the end, it will be launched in background.

- `GAG_HARDCOPY`
  the default type you wish for graphic hardcopies in GREG. Major possibilities are

```
EPS FAST      "Fast" grey-scale postscript
EPS GREY      "Nice" grey-scale postscript (clipping more accurate)
EPS COLOR      Color postscript
HPGL           HP-GL language
```

See GREG command DEVICE for details.

- GAG_PRINTER
  The (queue) name of the printer you prefer to use by default. Beware that it should be compatible with the description given in GAG_HARDCOPY. Funny results can be obtained sending HPGL commands to a PostScript printer.
- GILDAS_LOCAL:
  The path for your own GILDAS tasks. For advanced users only.
- GAG_TMP:
  A path where you whish to store temporary files which may be created by some applications. For optimum performance, use a local disk of the computer.

3.2.2 File Names

Two problems may arise when running GILDAS under Unix. The first one is due to the use of the as an option separator in SIC, and as a tree/subtree separator in pathnames under Unix. The second is that SIC is normally case unsensitive, while Unix is case sensitive. Placing the file name within double-quote will prevent making case conversion.

- SIC "Logical names" such as gag_log: are expanded in filenames.

```
  Typed named                  Expanded Filename
gag_log:toto.log            /users/me/.gag/logs/toto.log
```

- DOS-like names can also be typed and will be translated to Unix-like names.

```
  Typed named                  Expanded Filename
\users\me\toto.log            /users/me/toto.log
"sub\DIRE\Toto.Dat"           sub/DIRE/ToTo.dat
```

Filename handling becomes very simple when filenames are kept simple and lower case, as often happens.

3.2.3 Initialisation Files

All SIC based interactive programs read a initialisation file before starting. This file is a standard command procedure for the corresponding program. For example, for GREG it could contain:

```
SIC\DEFINE DOUBLE SEC /GLOBAL    !
SIC\LET SEC PI/180/3600         ! Second in Radian
SIC\SIC HELP CONTENT            ! Use PostScript file for HELP
GTVL\DEVICE                     ! Prompt for a graphics device
```

The initialisation file is located in your home directory, and its name is ‘‘init.program’’ where ‘‘program’’ is the default file extension for the application; this is usually just the application name.

4 Getting Started

The normal starting point is then to use the GREG graphic program, which will give you a first experience with the SIC command monitor. It is recommended to read first the SIC and GREG *Cookbooks* (15 and 5 pages respectively).

To get really started with image processing, read the GILDAS chapters 2 and 3 (*Running Tasks*, and *Displaying Images*) trying to run some of the image processing tasks (8 pages). However, since GILDAS is an image processing system, some practice with the `GREG2\` GREG language will be soon necessary. This practice can be done using the GRAPHIC program, but reading the GREG *Manual* will help (12 more pages).

Reference to more specific sections of the SIC and GREG documentations become only necessary for advanced users, and obviously for programmers. While the user become more and more familiar with GILDAS he (or she) will find that constant reference to the documentations is hardly ever necessary. The internal HELP is usually sufficient.

5 The Data Format

5.1 Description

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.

5.2 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.

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 256 Mbytes corresponds to data cubes of 256 by 256 by 256 (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.

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\TABLE` produces a pseudo-table which can further be processed to produce a spectral line data cube using `MAP\XY_MAP`.

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

6.2 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`.

6.3 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.

6.4 From and To other Packages: FITS

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

Disk FITS conversion can be a non-trivial task is the input FITS file has a complex structure. Several tools are thus available

- `FITS_GILDAS` (FITS to GILDAS) and `GILDAS_FITS` (GILDAS to FITS) Tasks can process simple FITS files
- `VECTOR\FITS FitsFile.fits FROM|TO GildasImage.gdf`
  provide a command line equivalent to the above tasks. Since FITS keywords are not fully normalized, although a standard subset is recommended, this command 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.
- `SIC\DEFINE FITS Var FitsFile` is similar to DEFINE IMAGE, and will create a SIC structure which will copy the FITS file content. The LET command can be then used to copy (part of) this structure into an Image variable if needed, and thus convert to the image format. It only works on existing FITS file however, and cannot be used to create a FITS file from a SIC structure.
- `SIC\DEFINE IMAGE` will automatically attempt to convert a FITS file into the Gildas equivalent, using the same method as FITS_GILDAS. This will work transparently on simple FITS images (with no FITS XTENSION).

Index

The Tasks, 3