The future of the GILDAS software

The GILDAS System Group

April 21, 1999

The GILDAS System Group includes:
Dominique Broguière, Gilles Duvert, Thierry Forveille, Stéphane Guilloteau, Roberto Neri, Alain Perrigouard, Robert Lucas, Pierre Valiron

Preamble

This document is intended to be used in keeping the history of the GILDAS software evolution. It contains proposed evolutions, summary of meetings, and current status in the realisation of the approved evolution scheme.

Contents

1 Introduction

The GILDAS software has evolved quite substantially since the original "SAS" program (later renamed LAS & CLASS) was written at the Groupe d'Astrophysique de Grenoble for the analysis the spectra from the POM-2 telescope.

GILDAS is now about 10 years old (some parts are 15 years old, others are only 2 years old), and has been ported to 5 operating systems (simultaneously or successively): RSX-11M, VAX-VMS, several Unix, MAC-OS and Windows-95/98. Because of short (Plateau de Bure antenna 6), medium (software upgrade of the 30-m, VLTI data reduction software), and long (LSA) term projects, it is time to re-evaluate the future of GILDAS.

2 History

For completeness, here are some of the major upgrades in this software, with some of the key authors. Please note that the list is not exhaustive: it is intended to illustrate the diversity of contributors and complex history of the software. Dates are quite approximate too, since the list is reconstructed from non-ECC astronomer memory.

- Initial development of the SIC monitor and SAS program on a PDP 11-34, in 1983, using PDP Fortran-77 language, by S.Guilloteau and R.Lucas
- Development under VAX-VMS of the GTLIB low-level graphic library by P.Valiron. This was a vector library supporting Tektronix compatible devices and pen plotters.
- Development of the GREG program by S.Guilloteau and P.Valiron. The initial GREG program only knew about plotting 1-Dimensional data.
- Incorporation of 2-Dimensional data contour facility in GREG, and introduction of the (in)famous "RGDATA" format for maps.
- 1985: LAS is adopted as spectral line package for the 30-m.
- circa 1986 (?): definition of the Gildas IMAGE data format (GDF). Access to the GDF images uses VAX-VMS memory mapping technique.
- ibid: development of the GILDAS Tasks and of the VECTOR language. Initial code used the RDPAR package for the task parameters.
- 1987: development of the mathematical formula handling in SIC by T.Forveille
- 1988: development of the CLIC package for the Plateau de Bure interferometer by R.Lucas
- 1988: development of the OVERLAY bitmap display program, based on the proprietary bitmap.
- 1989: Development of the imaging tasks.
- 1991: Support of GDF images in SIC (command DEFINE IMAGE & DEFINE HEADER).
- 1991: Start conversion to UNIX. Discard the RDPAR
- 1991: Development of the GTVIRT graphic library, by P.Valiron R.Gras, P.Begou and co-workers. The GTVIRT graphic library supersedes the GTLIB library, offering structured plots, integrated bitmap and vector plots, colour handling, and X-Window support.
- 1992: End support for VMS. Numerous upgrades by S.Guilloteau, P.Monger, G.Duvert in the GTVIRT library.
- 1993: Development of the MAPPING program by S.Guilloteau
- 1994: Porting to 64 bit environment.

- 1994 (?): small SIC syntax modification in character string handling.
- 1994: Development of the NIC software for the MPIfR bolometer at the 30-m, by D.Broguiere and R.Neri. Introduction of shared memory for the NIC data set.
- 1995: Porting to MAC-OS operating system by P.Rabou
- 1996: Development of the X-Window interface of SIC by F.Badia.
- 1996-1997: Implementation of new imaging techniques by F.Gueth and K.Bouyoucef
- 1997: Porting to Windows-95 operating system.
- 1998: Generalized use of X-Window interfaces for Plateau de Bure calibration and imaging.
- 1998: Incorporation by G.Duvert of "Structure"-like variables, which could simplify FITS interface.
- ...: ?

GILDAS is essentially coded in Fortran-77 (98 % of the code), with just a few C routines for system interface (including with the X-Window system).

3 Evolution or Revolution ?

3.1 Possible Options

GILDAS was developped and handled by a relatively small group of people. It is estimated between 0.5 to 1.5 Full Time Equivalent persons were working on the development of GILDAS. This manpower was devoted mostly to the application domain (CLIC, Tasks) and only very seldom to the "system" domain, where support for portability has been the most time consuming task. The total man-years is thus presumably of order 15.

Other interferometry/single-dish packages like MIRIAD and GIPSY, are being abandonned by their support observatories to be eventually replaced by AIPS++. In comparison with GILDAS, the AIPS++ project now has 17 FTEs working on the project, and has been on-going since early 1991. The estimated number of man years exceeds 100, and the first public release is foreseen in March 1999.

It is clear that the two softwares cannot be compared. However, the future of GILDAS, and of the IRAM data processing software in general, must be adressed now. IRAM faces several options: continuing support to GILDAS, joining AIPS++, or re-investing into other data handling packages, either commercial (like IDL) or non commercial (like IRAF). Several observatories have used IDL extensively for data reduction, mostly in Space programs (e.g. ISO), but have faced compatibility issues between different releases of the IDL suite.

Based on such experience, and given IRAM field of applications, we feel only the AIPS++ or GILDAS(++) options are realistic.

3.2 Revolution: the AIPS++ option

IRAM could decide to join on the AIPS++ project. IRAM should be able to devote 1 to 1.5 FTE to that activity. Given the limited resources at IRAM, this would imply almost complete suppression of support for the current GILDAS package. Moreover, since AIPS++ is not yet fully operational, this would open a time window in which no substantial development could be undertaken. We assume here that an initial period of 1 to 2 years would be required to get acquainted with AIPS++ before being able to add useful tasks. If AIPS++ stands up to its promise of "ease of programmability", this may be reduced.

The following areas of IRAM activities would most likely suffer from such lack of development:

- Multibeam observations with the 30-m
- Self-calibration with the 6-antenna Plateau de Bure interferometer
- New generation 30-m software might be deferred to the end of the training period.

The list is of course not exhaustive.

3.3 Evolution

In contrast, IRAM could decide a moderate evolution of the GILDAS software. The risk is then to commit manpower to this evolution, and become decoupled from the AIPS++ suite which might become **the** radio astronomy package of the future.

On the long term, with only IRAM & associates (Observatoire de Grenoble, MPIfR, other French observatories), it is conceivable that the IRAM community produces and maintain a radio-astronomy package well suited to the IRAM instruments (30-m, Plateau de Bure), and also to "Standard" imaging with ALMA (Atacama Large Millimeter Array, ex LSA/MMA). More fancy applications such as very large mosaics, full polarisation measurements, super-computer class images, would most likely be beyond reach.

Obviously, the IRAM community should express its view point on the future of IRAM software.

3.4 January 1999 meeting conclusion

A meeting was held in January 1999, with all GILDAS authors present.

D.Broguière summed up its evaluation of AIPS++. Besides installation problems, AIPS++ was found relatively slow on Plateau de Bure applications. The slowness was later tracked back to ordering of the UV data, although GILDAS software is still quite faster. Although AIPS++ is evolving positively, it is still not mature.

AIPS++ command language is non-declarative, variables being (re)-typed and (re)-dimensionned according to their content. This was disliked by most of the attendent (if not all).

P.Valiron described its interest in SIC as one of the command languages allowing mathematical operations, independent of any tailored applications by opposition to AIPS++. GILDAS is also widely used in various astronomical institutes (including NRAO).

Based on these premises, it was decided to go on a substantial refurbishment of the GILDAS software.

4 The Proposal for SIC evolution

4.1 Medium Term

The SIC command monitor should be improved along the following lines:

- more rigorous syntax and grammar
- compatibility as far as possible with the old syntax
- Keep the "Fortran" interface for mathematics. C-like mathematics would cause problems because of array ordering, and C is less suited for mathematics and array handling than Fortran.
- Better mathematic handling, including more functions.
  - "Extensible dimension":
    support for formulas like `C = A/B` where `A` and `C` are N-Dim arrays and `B` is a N-1 Dim array.
  - Suppression of the COMPUTE command by operators, e.g. `COMPUTE B FFT A` is replaced by `B = FFT(A)`
  - support for reduction operations, e.g. `COMPUTE C MAX A` is replaced by `C = MAX(A)`
  - More functions, e.g. `DIM(A) SIZEOF(A)` etc... are desirable
  - Character sub-string extraction would be useful

  Whether inner and outer product operators should be introduced may be considered.

This improvement could be a subject for a student work. Gilles Duvert will write up the scope, goals and limits of the subject. Depending on whether a student can be found for this spring/summer or not, the timescale for this upgrade could be 6 months to 18 months.

Multi-tasking was mentionned as a possible development line, by introduction of `events`, perhaps in a AIPS++ style. This could simplify the handling of interactive graphic events, such as cursor operation. This could also simplify the GUI interface programming. No proposal was put forward at this stage.

Action Gilles Duvert to write up working plan.

4.2 Short Term

On the short term, the following points were mentionned:

- Structuration of command procedures
- Handling of double quotes in character string
- Better handling of upper case / lower case problem

Such short term evolution were important mainly from IRAM point of view because of the existence of complex command procedures. Evolution of SIC was agreed upon, with a prototype to be developed by S.Guilloteau.

5 Status of SIC evolution

5.1 Syntax

- **Case conversion**
  SIC would no longer convert the command line in upper case. Commands, keywords, variables and mathematics expressions would be recognized whatever the case, but the rest of the command string would be taken "as typed"

  Example:
  `SIC EXTENSION .class` and `Sic EXtension .class` are equivalent
  but not `SIC EXTENSION .CLASS`

- **Separators**
  The only separator would be the space (" "). Slashes ("/") would no longer be a separator, but could either be an option delimiter or just part of a string. This allows the use of / as the division token.

  A pending question is whether a slash can begin a string without double quotes. This would allow file names to be typed directly, but would prevent the user of being warned by an error message in case of mistyped option. Robert Lucas and Stephane Guilloteau propose that
  - Single slash necessarily begins an option
  - Double slash can be used to begin an absolute path/file name

  Example

  ```
  file out /usr /new        ! Returns error "No such option /usr"
  file out /usr/toto/new    ! Returns error "No such option /usr/toto/new"
  file out //usr/toto /new  ! Creates file /usr/toto  (option /new)
  file out "/usr/toto" /new ! Creates file /usr/toto  (option /new)
  ```

  4 voted in favor, 2 against (noting that double slash creates unecessary exception, quotes can be used instead), 2 did not comment. This issue is still pending (there is no programming problem).

- **Double Quotes**
  Double quotes will be handled in character strings as follows: two consecutive double quotes will produce one double quote in the string. (The current behaviour was to produce TWO separate strings by inserting a space between the two double quotes)

```
SAY "toto = ""toto""!"
```

will result in

```
toto = "toto"!
```

- **Better Syntax Checking**
  SIC now tries to complain on `‘‘invalid syntax’’` as much as possible at parsing time. De facto, the only allowed characters before a " are `( ' .` and `space`, and after a " are `) ' .` and `space`.

  This cleans up a previously strange behaviour, which gave error message, but still produced some (clumsy) output.
  Example:

  ```
  command "toto"TATA"titi"     !
  command "toto"'TATA'"titi"   !
  command "toto"/titi          !
  command "toto"+titi          !
  command "toto"titi           !
  ```

  are all invalid, and recognized as such at parsing time.

  ```
  command "toto"'titi'         ! TITI is a known variable
  ```

  is valid (because `titi` is a known variable), but

  ```
  command "toto"'titi'         ! TITI is NOT a variable
  command "toto".eq.           !
  ```

  are both invalid, but recognized only as such at execution time.

5.2 Search Path for procedures

SIC may define a path (by opposition to a directory as now) for procedure. After tests, the following syntax is proposed:

```
SIC MACRO_PATH "./;GAG_CURRENT:pro/greg/;GAG_CURRENT:pro/clic/;"
```

The initial proposal (to allow logical names to define a path) would have caused logic problems when creating files. Unix style syntax was discussed, but rejected (the two proposals for Unix-like syntax were different).

A.Perrigouard suggested that if a machine hosting a ressource has to be specified a double slash may introduce it, like //iraux2/usr/local. This possibility is not implemented so far (see 5.1).

5.3 Procedure Structuration

Procedures can be structured. We introduce the following new possibilities to structure procedures, their help files, and data files.

```
BEGIN PROCEDURE procedure_name
...
END PROCEDURE
BEGIN HELP procedure_name
...
END HELP
BEGIN FILE file_name
...
END FILE
```

For symmetry, the `ENDIF` command is renamed in `END IF` A symbol `ENDIF` is defined for compatibility with previous syntax. Other alternatives were unanimously discarded.

5.4 New Commands

The following commands have been introduced

- `SIC MACRO Path`
  defines the macro search path
- `LET Variable Expression /UPPER (resp /LOWER)`
  Assign a character variable with conversion in Upper (resp Lower) case. This is required to properly interpret keywords, since there is no longer any automatic case conversion.
- `LET Variable Expression /FORMAT String`
  Assign a character variable by formatting another variable or expression. In case the expression defines the name of a character variable, this character variable is formatted. Example:
  ```
  LET I 1
  LET B1 "Toto Tutu"
  LET A B'I'                 A = B1
  LET A B'I' /FORMAT A12     A = Toto Tutu
  ```
- `BEGIN PROCEDURE File` and `BEGIN HELP File`
- `END PROCEDURE` and `END HELP`
- `END IF`
  which replaces the previous `ENDIF` command.

5.5 New Subroutines

To implement the changes, the following subroutines have been created:

```
SIC_CH(LINE,IOPT,IARG,ARGUM,LENGTH,PRESENT,ERROR)
```

returns character string (as before)

```
SIC_KE(LINE,IOPT,IARG,ARGUM,LENGTH,PRESENT,ERROR)
```

returns a keyword (upcase converted).

All programs have been converted to use `SIC_KE` whenever necessary. This guarantees the compatibility of the programs with both versions (old and new) of SIC.

```
SUBROUTINE SIC_DESC (LINE,IOPT,IARG,DESC,DEFAULT,PRESENT,ERROR)
```

returns a SIC descriptor (DESC) corresponding to the IARGth argument of option IOPT. The expected argument type (R*4, R*8, I*4, or C*(*)) is derived from the supplied DEFAULT descriptor. The argument must thus be either a constant or a variable of the appropriate type, but cannot be a mathematic expression. If the argument is "*" (wildcard), the programmer supplied DEFAULT descriptor is returned.

```
SUBROUTINE SIC_INCA (LINE,IOPT,IARG,DESC,DEFAULT,PRESENT,ERROR)
```

returns a SIC descriptor (DESC) corresponding to an incarnation of the IARGth argument of option IOPT. The expected argument type (R*4, R*8, I*4, or C*(*)) is derived from the supplied DEFAULT descriptor. The argument can thus be either a constant, a variable, or a mathematical expression. Proper incarnation in the desired type is performed if needed. If the argument is "*" (wildcard), the DEFAULT descriptor is returned.

```
INTEGER FUNCTION SIC_FINDFILE (NAME,FILE,PATH,EXT)
```

which searches for an existing `FILE` of given `NAME` on path `PATH`, and with default extension `EXT`. Returns 0 if the file is found, 1 otherwise.

Another possible subroutine which could be useful:

```
SUBROUTINE SIC_FILE (LINE,IOPT,IARG,NAME,FILE,PATH,TYPE,PRESENT,ERROR)
```

would combine the current sequence of `SIC_CH` and `SIC_PARSEF`, or two variants, one for new files, and one for existing files (combining `SIC_CH` and `SIC_FINDFILE`).

5.6 FITS Support

FITS support is being tested by Gilles Duvert. FITS support makes use of the DEFINE STRUCTURE command previously implemented by Gilles. The notion of structured variable, like `A%Item%Subitem`, allows easier reading of FITS files. Because the FITS format was/is too flexible, reading FITS files has always been a major problem since FITS keywords are not fully normalized. The definition of structured variables allow procedural mapping of FITS keywords to internal variables without recompiling the application.

The agreed syntax is

```
DEFINE FITS Generic_name File
```

by analogy with `DEFINE IMAGE, DEFINE REAL, DEFINE HEADER, etc...` (unanimous agreement, although the main person concerned has not replied...).

The syntax was easily understood to read existing FITS file. However, what happens to create an new FITS file was unclear, and needs some description.

6 Graphic Display Evolution

The GTVIRT graphic library works (and works fast), but is not fully satisfactory. Its maintenance is difficult, the use of the plot hierarchy not straightforward, color support not clear, etc...

To simplify it, it was suggested to drop off old code supporting old peculiar devices. However, the multi-protocol aspect of the library was considered as essential. It could allow ultimately to display in Java for example. We should also keep the DEVICE TEKTRO X for slow links.

The following short term improvements were mentionned:

- Addition of "Expandable rectangular box" dragged by mouse.
- Addition of "Expandable vector" dragged by mouse
- Addition of "Mouse defined polygon". This would be faster and simpler than the current ways of defining polygons.
- Addition of "Mouse defined region", basically limited by the screen resolution. This would essentially be a large polygon with corners defined by the screen resolution.
- Better support for 24-bit or 32-bit colour display should
- Re-writing of the GTVIRT to use multi-thread programming to separate window display from internal plot handling. This would allow resizing and refresh events to be handled much more conveniently. SIC events (as mentionned before) would be practicle in this respect.

Evolutions in GreG would be nil or very limited, e.g. a zoomed box (by opposition to a zoom window) could be easily implemented once the "Expandable rectangular box" is available in the GTVIRT.

A. Perrigouard declared to be interested by these subjects. A proper description of the GTVIRT library concepts (goals ?) would be required to get started (Stephane Guilloteau ?).

7 Programming - Portability

In programming style, it is now reasonable to convert the package to Fortran-90, since the language is mature and compilers can be found on most machines.

Porting could most likely be limited to a smaller number of operating systems. Since LinUX is becoming popular, this should become THE basic system for GILDAS software. Removing support for other flavors of Unix would free substantial manpower.

We have improved portability by suppressing the use of the NAG library, and by using public domain libraries. **GILDAS** now requires two public domain libraries: **LAPACK** and **SLATEC**. Robert Lucas wrote a new version of `UV_FIT` using the NETLIB SLATEC library, and Dominique Broguiere modified NIC to use it for 2-D gaussian fitting.

8 Documentation and Help

It would be convenient to generalize the use of HTML, through the LaTeX2html translator, to all parts of the documentation and help system. This requires a text browser, for simpler access.

It has been decided to write a true programmer manual, and remove from the general user documentations the corresponding sections. This has been done, although the programming guide is still preliminary.

9 Progress Report

The new SIC evolution has been implemented in a full scale test. CLIC, CLASS, GRAPHIC, NIC and MAPPING have been tested. A number of bugs in SIC have been discovered and corrected while doing so.

A tool to convert semi-automatically command procedures has been written.

The "Programming Manual" is underway.

There is no significant progress on the GTVIRT case so far.

10 Pending Question and Action Items

- Is the TASK language still necessary ? Can we simplify that by more simple procedures ?
- Documentation: Should we move all help files to html format? Can we install the "lynx" text browser to test this capability ? Who does that ?
- Fortran 90 usage:
  Using Fortran 90 would simplify some programs. There is a free linux compiler. We can buy a f77-f90 converter (cost depending on number of line codes converted see http://www.psrv.com/vast77to90.html).

  Implications on our general coding practices, in particular the pre- processor should however be evaluated. Remember, we are using some special f90 constructs like INTEGER(KIND=4) and POINTER as having very different meanings in our pre-processor.

  - Who evaluates in detail the impact on portability / reconstruction ?
  - Can we start writing new code in f90, at least in tasks ?
  - Is it worth to automatically convert our old f77 code to f90 code ?