IRAM Memo 2011-1

Preparing GILDAS for large datasets
I - GREG 2011

S. Bardeau[1], E. Reynier[1], J. Pety[1,2]

1. IRAM (Grenoble)
2. Observatoire de Paris

26-jun-2013
version 1.4

Abstract

The size of the datasets produced by the IRAM instruments experience a tremendeous increase (because of multi-beam receivers, wide bandwidth receivers, spectrometers with thousands of channels, and/or new observing mode like the interferometric on-the-fly). Visualizing these datasets in a fluent way is a challenge, which requires the best use of the available hardware and operating systems (multi-cores processors and multi-window environments). This prompted a large rewriting of the part of the GILDAS kernel (known as GTV) in charge of the interface between the plotting facilities and the system. The main guidelines of this rewriting were 1) the backward compatibility when possible, 2) the use of modern standards as the multi-threading or the GTK+ toolkit, 3) the factorization of the source code for different OS (Linux, Mac OSX and MS Windows), 4) the implementation of new facilities like a PNG output or an interactive lens. This document thoroughly describes the improvements for the end-users and the programmers.

Keywords: multithreading, multiwindowing, the GIMP ToolKit (GTK+), Portable Network Graphics (PNG)
Related documents: *GreG documentation*, *Programming in GILDAS*

Contents

1 Why a new implementation of the GTV library? 3

2 The new cool things for the enthousiastic users 3

3 Requirements 4
3.1 Widget and graphic library 4
3.2 Fortran compilers 5

4 Changes for end-users 5
4.1 Behavior 5
4.1.1 Removed 5
4.1.2 Changed 6
4.1.3 New 7
4.2 Commands, variables, SIC logicals and symbols 7

5 Changes for programmers 11
5.1 The new GTV overview 11
5.2 Entry points 12
5.2.1 Removed 12
5.2.2 New 13

A Exhaustive description of the changes in GTV 14
A.1 Commands 14
A.1.1 Removed 14
A.1.2 Changed 16
A.1.3 New 17
A.2 Variables 19
A.2.1 Removed 19
A.2.2 Changed 19
A.2.3 New 19
A.3 SIC logicals 20
A.3.1 Removed 20
A.3.2 New 20
A.4 Symbols 20
A.4.1 Removed 20

1 Why a new implementation of the GTV library?

The `GreG` program lies on the so-called `GTV` library, a low level segment library. This library was written two decades ago, and not updated for several years. This was resulting in old-fashion graphical aesthetics for nowadays users, with some annoying constraints from other ages, e.g. windows which where not refreshed as often as the user would like.

For the maintainers point of view, the lack of experts of the source code was a big problem to fix bugs or improve any feature, especially with an old-fashion source code. Finally the status was usual rule in such a case: *"If you don't touch it you don't break it"*.

It was decided in 2007 to act before reaching the non-return point. In particular it was time to cut the dependency with the old widget library `Motif` (and its various implementations giving more or less satisfying results). The choice was made to use GTK instead, a well-known widget library for Linux users (but also supported on MS Windows and Mac OSX), and actively developed by the community. This was a first part of the work.

Another key piece is that GTV could only do one action at a time. By *action* one have to understand: storing a new plot coming from a user command, or drawing in one window, or plotting an hardcopy, etc. In particular this explains why the windows could not all be refreshed when desired. From the developers point of view, it was due to the fact that a large amount of the drawings were performed with global variables (some of them were internal to the library, but some others were the user property!). Thanks to modern Fortran features (namely derived types and pointers), the majority of the variables are not global anymore, but are passed to the drawing routines through a small number of structures describing the plot and its destination.

Finally, thanks to the above point, it was possible to implement GTV as a multi-threaded library: the user can work on its data while all its X windows are automatically refreshed or updated.

In this new context, the field is open for new and modern graphical tools. For example, a *lens* is already available: a popup window can be opened to zoom and browse a plotting window with the cursor. And more tools are already considered for the future.

2 The new cool things for the enthousiastic users

Will your life change with the new GTV? Well, not exactly, but consider these points:

- the GTV is now **multithreaded**: this means basically that the drawings and the flow of commands (including the data processing) are done in two different *threads*. With the efficient rewriting of the code and your multi-core machine, you will experience much more fluid graphics: the drawing events won't be slowed or postponed by the data processing, and vice-versa.
- **GTK+** is now used as the widget library. No more old-fashion windows, forms, or file browsing popups, they will look like what you are used to.
- **multi-windowing** is more user friendly: you can have several plots in several windows, all are refreshed in real time. No more `ZOOM REFRESH`: windows are always up-to-date. You

can also define the windows size and position with much more flexibility. The first window is not anymore forced to open in the top right corner of your screen: now YOU decide if your windows should open or not at a specific position.

- **zooming** is now easier and more interactive: a lens is now available at any time when center-clicking in any plotting window, and zooming/unzooming is done with the mouse wheel.
- tired of PostScript captures, rotations, and even color conversions? Now you can convert (hardcopy) your plot natively to the **PNG image format**. In particular, the background transparency is supported for an enhanced cosmetic of your presentations.

There also other things which will ease your life:

- **Encapsulated PostScripts** of landscape-oriented plots are not rotated anymore to portrait orientation. It is assumed that you will include it in another document (Latex source, presentation, etc): you will not have do derotate it back to landscape orientation.
- the **Look-Up-Table dynamics** has been increased from 128 to 65536 levels. You will not see anymore staircasing effects on smoothly-varying images.
- the so-called ***static* Look-Up Table mode** works now: you can use one Look-Up Table per image in a single window.
- the **GTV variables** have been merged under a few SIC structures: this avoids diluting your own variables into a large number of program variables.
- some **command and options** have been renamed for clarity, and to get rid of obsolete nomenclature. Among other things, you will not have anymore to figure out which one of the nine `CLEAR` flavors you need. For a smooth transition, GTV will recognize the old syntax commands and let you know what they should be replaced with. Some **variables, SIC logicals, and symbols** have also changed for the same reasons: you will find a summary of all the changes in the tables 2, 3, 4 and 5.
- N^2 **algorithms** in the source code are fixed. You think that `GO SPECTRUM` or `GO BIT` are taking ages on a big data cube, slowing down at a point you feel they will never finish? Try again now!
- no more **weird error messages** like *"Not enough colors"* or *"Too many images plotted"*: the internal limitations have been removed and you are now only limited by the physical memory of your computer.

And more will come in the future!

3 Requirements

3.1 Widget and graphic library

The GIMP ToolKit (GTK+) is now used as the widget library, while the GIMP Drawing Kit (GDK) is used as the graphic library. This choice ensures portable code and similar rendering on the three major sytems Gildas supports: Linux, Mac OSX, MS Windows. GTK+ and GDK are thus now a requirement in order to compile the Gildas kernel. They are provided together

on the system repositories as a unique package. Its presence on your system can be checked with the following command:

```
pkg-config --exists gtk+-3.0; echo $?
```

If a 0 is returned (i.e. no error), then you will be able to compile the new Gildas kernel. If another value is returned, you will need first to install the package.

The GTK+/GDK package has been tested between versions 2.10 and 2.22. Beyond this range, you can encounter GTK warnings or errors. Please report these to `gildas@iram.fr`, with your GTK version which you can find in the file `gtk/gtkversion.h` installed on your system.

In addition, the support for the old widget library `Xforms` is removed. Its support was still present but was not tested for years. The support for the `MOTIF` library (the default used with the previous kernel) is obsolete and is considered for complete removal soon.

3.2 Fortran compilers

At the time this documentation is written (14-dec-2010), here is the status of the various Fortran compilers supported by Gildas:

- Intel Fortran Compiler (ifort): versions 9.0 and 11.1 have been tested. As far as we know it seems fine to compile the new kernel with ifort. There is no bug directly related to the new kernel, but remember there can be problems in old version of ifort (e.g. a memory leak with ifort 9.0).
- g95: none of the stable versions of g95 (i.e. up to 0.92) is now able to run correctly the new Gildas kernel. The Fortran runtime libraries it provides do not seem to be thread-safe, i.e. there are conflictual access to its internal routines. With the implementation of the Fortran co-arrays, the current development version of g95 (0.93) should be thread-safe. The problem is that this version is not stable enough to compile the Gildas kernel, and developments seem frozen since August 2010.
- the GNU Fortran Compiler (gfortran): recent versions of gfortran are able to compile and run correctly the new Gildas kernel. The lower limit 4.3.0 required for the gfortran version and introduced 2 years ago is kept unchanged. The annoying bug which enforce the user to type `RETURN` after clicking in a widget is **not** related to the new kernel. A patch for this bug has been submitted to the development versions of gfortran, and it will be part of the releases 4.4.6, 4.5.2 and 4.6.0.

4 Changes for end-users

All the obsolete, changed or new concepts are exhaustively described in this section and in the appendix A.

4.1 Behavior

4.1.1 Removed

There is a number of *devices* officially supported by GTV which have been removed. In the GTV context, a *device* is a destination for the plot stored in the GTV tree (e.g. X-Windows or PostScripts). The following ones have been removed, mostly because they are just obsolete:

- HPGL
- REGIS
- TEKTRO

This includes all the variants which were available in the Gildas `kernel/etc/` sources, but not compiled by default in the standard Terminal Definition File.

The device `SVG INTERACTIVE` is also disabled, at least as long as its need is not clearly required. The device `SVG [LOCAL]` is kept for hardcopies.

4.1.2 Changed

- The EPS (Encapsulated PostScript) hardcopies are not automatically rotated anymore. The reason is that we assume now that EPS are intended to be included in another document rather than being printed. The new option `/FITPAGE` for command `HARDCOPY` enables automatic rotation and scaling (i.e. the historical behavior). You can also set the global variable `GTV%FITPAGE` to `YES` for an identical result. The option `/PRINT` (formerly `/PLOT`) implicitely activates `/FITPAGE`. Finally, the standard (non-encapsulated) PostScripts have not been changed. The table 1 summarizes the former and new rotation rules.
- The default Look-Up Table (LUT) is now a continuous variation of colors, from black to white. It is similar to the LUT which can be loaded from the file `rainbow3.lut`, probably the most popular LUT in GTV. It can be reloaded at any time with the command `LUT DEFAULT`. The previous default LUT was also a continuous variation of colors (circular hue value), but from red to red, resulting in confusing low and high levels. It still can be loaded with the command `LUT COLOR`.
- The so-called "LUT static" mode has been fixed and works correctly now. "LUT static" mode can be used to define one LUT per GreG image. It also applies to pens 8 to 23 used by *e.g.* `RGMAP /GREY`. See `HELP GTVL\CHANGE LUT` for details.
- Now the pen is a global atribute, e.g.:

```
GREG\PEN 0  ! Black
CREATE DIRECTORY SUB
CHANGE DIRECTORY SUB
GREG\PEN 1  ! Red
CHANGE DIRECTORY ..
```

  With the above sequence of commands, the next segments created in the first directory would have been black, since there was a memory of the last pen in use when leaving the directory. Now, the pen used *everywhere* is the last pen defined *anywhere*.

Table 1: How the PostScripts and Encapsulated PostScripts are rotated or not depending on the GTV version and the plotting page. *Condition1*: Yes if the X range of the drawings is larger than its Y range (*bounding box*), else No. This table reflects how the printers or the Latex inclusions behave. It does not reflect how some viewers can display the PostScript, in particular *ghostview*.

	Old		New					
Kind	EPS	PS	EPS	PS	EPS	PS	EPS	PS
					`HARD /FITPAGE`		`GTV%FITPAGE YES`	
Landscape	Yes	Yes	No	Yes	Condition1	Yes	Condition1	Yes
Portrait	No	No	No	No	Condition1	No	Condition1	No

4.1.3 New

- An interactive "lens" tool: browsing and zooming in your figures is now enhanced thanks to a new tool: i) open the lens with a center-click on any plotting window, ii) zoom in or out with the mouse wheel, iii) enlarge or reduce the lens with CTRL+wheel, iv) close the lens with center-click again.
- PNG hardcopies: in the context of modern use of figures (e.g. digital presentations, webpages, etc), Greg plots can now be transfered natively to PNG images. See `HELP GTVL\HARDCOPY` for details. In particular, these images support background transparency

4.2 Commands, variables, SIC logicals and symbols

There has been an important effort to clarify the names of the various GTV *objects*, to fix or improve their behavior, and to clean out the obsolete ones. This can imply here and there an action from the user to make its procedures compatible with the new GTV. The new GTV knows about the old commands: it will detect these and suggest to the user what they should be replaced with. It can run in 2 modes, tolerant or strict, which can be toggled with the SIC variable `GTV%STRICT2011`. All the changes regarding these objects are exhaustively described in the appendix A. The tables 2, 3, 4 and 5 summarize these changes.

Table 2: Old GTV and GREG commands which are removed or modified, their purpose, and their equivalent with the new GTV syntax. Refer to section A.1 for a detailed description.

Old command	Behavior	Equivalent
CLEAR	perform a CLEAR PLOT	see CLEAR PLOT
CLEAR ALPHA	bring the plotting window in front	not yet defined
CLEAR GRAPHIC	bring the terminal in front	none
CLEAR PLOT	perform a CLEAR WHOLE, or perform a CLEAR TREE	DESTROY ALL, or CHANGE DIRECTORY + CLEAR DIRECTORY
CLEAR TREE	destroy the current tree	CHANGE DIRECTORY + CLEAR DIRECTORY
CLEAR WHOLE	destroy all the trees	DESTROY ALL
CLEAR WINDOW [Num]	close the active/numbered window	DESTROY WINDOW [Dir [Num]]
CHANGE CLEAR TREE|WHOLE	set the behavior of CLEAR PLOT	none
CHANGE DRAW ON|OFF	toggle the drawing state to active or sleeping	none
CHANGE ZOOM NEW|CURRENT	toggle the default behavior of ZOOM	none
CREATE DIRECTORY /PIXEL	set the window size of a new top directory	CREATE DIRECTORY /GEOMETRY
CREATE DIRECTORY /SIZE	set the plot page of a new top directory	CREATE DIRECTORY /PLOT_PAGE
DISPLAY CLEAR	show the CLEAR PLOT behavior	none
HARDCOPY /PLOT	send the hardcopy to the printer	HARDCOPY /PRINT
ZOOM REFRESH	force the redraw of the active window	REFRESH [Dir [Num]]
ZOOM /REGION	define the zoom area with the cursor	none
ZOOM /CURRENT	zoom is performed in the current window	none
ZOOM /NEW	zoom is performed in a new window	ZOOM without argument
GREG1\SET DRAW ON|OFF	toggle the drawing state to active or sleeping	none
GREG1\SHOW DRAW	display the drawing state	none

Table 3: GTV and GREG variables which are removed, modified, or added. Their read-write status is indicated in the third column. Refer to section A.2 for a detailed description.

Old name	New name	Read-Write	Usage
GTV%SLEEP	none	RO	the active/sleeping drawing state
GTV%IMAGES[4,15]	none	RO	internal values associated to the first 15 images
GTV%IDENT	none	RW	the number of the device currently in use
LUT	none	RO	the hue values of the current LUT in use
REXTR	none	RO	the extrema of the last segment created
REXTR_D	none	RO	the extrema of the current working directory
none	GTV%PWD	RO	the current working directory
none	GTV%FITPAGE	RW	activate option /FITPAGE for HARDCOPY
GTV%LUT_STATIC	LUT%STATIC	RO	the dynamic/static LUT state
LUT_SIZE	LUT%SIZE	RO	the number of levels of the current LUT
LUT_MODE	LUT%MODE	RW	the HSV or RGB mode for the user-defined LUT
HUE	LUT%HUE	RW	the hue values of the current LUT (input/output)
SATURATION	LUT%SATURATION	RW	same for the saturation values
VALUE	LUT%VALUE	RW	same for the intensity values
RED	LUT%RED	RW	same for the red intensity values
GREEN	LUT%GREEN	RW	same for the green intensity values
BLUE	LUT%BLUE	RW	same for the blue intensity values
B_HUE	LUT%BLANKING%HUE	RW	the hue value of the blanking color (input/output)
B_SATURATION	LUT%BLANKING%SATURATION	RW	same for the saturation value
B_VALUE	LUT%BLANKING%VALUE	RW	same for the intensity value
B_RED	LUT%BLANKING%RED	RW	same for the red intensity value
B_GREEN	LUT%BLANKING%GREEN	RW	same for the green intensity value
B_BLUE	LUT%BLANKING%BLUE	RW	same for the blue intensity value
P_HUE	LUT%PEN%HUE	RW	the hue values of the user-defined pens (input/output)
P_SATURATION	LUT%PEN%SATURATION	RW	same for the saturation values
P_VALUE	LUT%PEN%VALUE	RW	same for the intensity values
P_RED	LUT%PEN%RED	RW	same for the red intensity values
P_GREEN	LUT%PEN%GREEN	RW	same for the green intensity values
P_BLUE	LUT%PEN%BLUE	RW	same for the blue intensity values
LCUT	CURIMA%SCALE[1]	RO	low cut for last image PLOT'ted
HCUT	CURIMA%SCALE[2]	RO	high cut for last image PLOT'ted
SCALING	CURIMA%SCALING	RO	scaling mode for last image PLOT'ted
EQUAL_NLEV	CURIMA%EQUAL%NLEV	RO	number of levels in EQUAL mode
EQUAL_LEV	CURIMA%EQUAL%LEV	RO	levels in EQUAL mode
EQUAL_HIST	CURIMA%EQUAL%HIST	RO	values per level in EQUAL mode

Table 4: SIC logicals in GTV which are modified or added. Refer to section A.3 for a detailed description.

Old name	Equivalent name	Usage
LUT_DIR:	LUT#DIR:	where the LUT files should be searched in
none	WINDOW_GEOMETRY	the default geometry of new windows
none	WINDOW_POSITION	the position on screen of the first window

Table 5: Old GTV symbols which are removed, and the command which can be used to define them in user's ∼/.gag.dico . Refer to section A.4 for a detailed description.

Old symbol	Equivalent
CD	SYMBOL CD "GTVL\CHANGE DIRECTORY"
PWD	SYMBOL PWD "GTVL\DISPLAY DIRECTORY"
MKDIR	SYMBOL MKDIR "GTVL\CREATE DIRECTORY"

5 Changes for programmers

5.1 The new GTV overview

In the previous version of the GTV, the metacode describing the plots was stored in several chunks of memory. The current chunk was contiguously filled with all the incoming data: directory descriptors, segments descriptors, segment data, or image descriptors. When the current chunk was (nearly) full, it was copied to a newly allocated chunk, and reset for new use. This had 3 major limitations:

- it was a permanent worry to make sure that the incoming data will fit in the remaining place of the current chunk,
- the tree size and the numbers of images in the tree were limited by hard coded parameters in the source code,
- the various links in the tree between the directories, the segments, and their data, were remembering the chunk number and the position in the chunk. Deleting an object and freeing its associated memory was just a nightmare: links could not be updated easily.

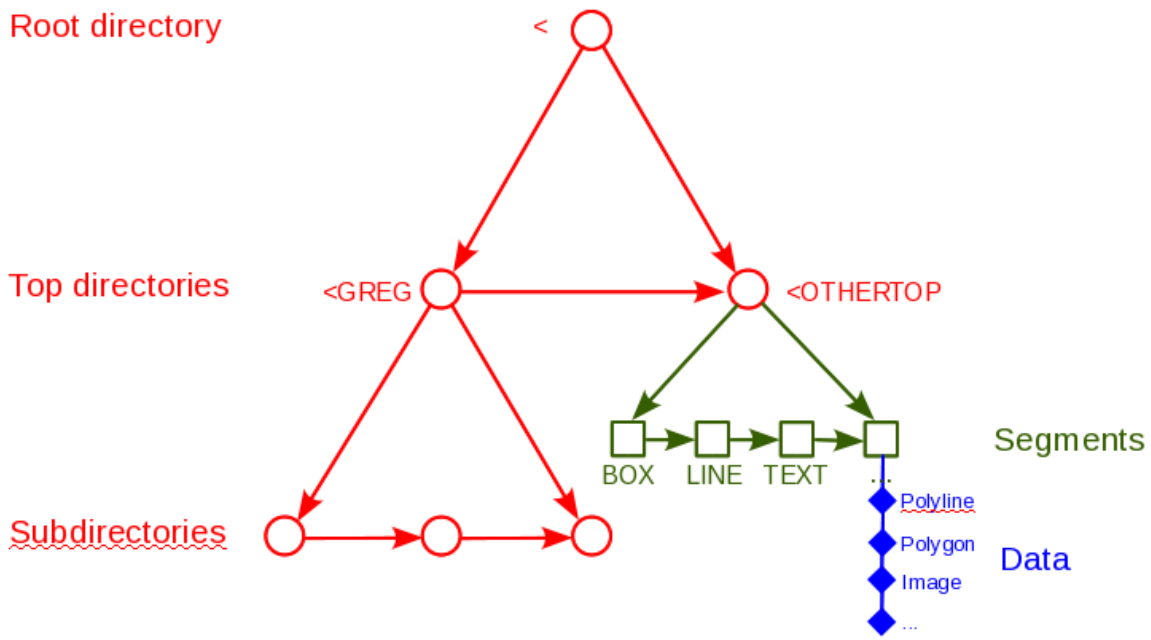


Figure 1: The tree and linked lists structure of the metacode storing the plots in the GTV, with a basic example of top directories, subdirectories, and segments. The relationship presented here between the various elements is not exhaustive.

Given these limitations, the choice was made to fully rewrite the way the metacode data is stored in memory. Appropriate Fortran derived types were defined: they describe the various links an object can have with others in the tree. Figure 1 shows an example of the various kind

of data which can be encountered in the metacode.

All the links are made with allocatable pointers, which are allocated in real time, on-demand. The tree is thus now growing dynamically in memory. Deleting an object is just as easy as deallocating a pointer to it, after having taken care that the surrounding elements in the tree are correctly updated. Furthermore, deallocation can be delayed (i.e. put in the event stack) after all the events which need it are done, without altering the normal life of the tree (as ruled by the commands executed).

5.2 Entry points

5.2.1 Removed

The so-called "immediate pen" support and associated entry points are removed:

- `gti_where`
- `gti_pen`
- `gti_out`
- `gti_draw`
- `gti_reloc`

The "immediate pen" was used to plot lines directly on the default window, without storing them in the metacode. This was an unsatisfying mechanism since: i) these drawings (probably useful) were lost as soon as the window was redrawn, ii) the window manager could ask for a window redraw at any time.

The `Sleep` and `Wake_Up` modes for `gtview` are removed. They were intended to delay the drawing of new segments being added to the metacode. This behavior has been removed because i) the drawing is performed in another thread i.e. it does not slow down the main thread execution, ii) the drawing routines were speed up to avoid re-reading the whole tree in order to search for undrawn segments, iii) user always wants to see the segments being added as soon as possible. As a consequence the following entry points are removed:

- `gtstat` (it was returning the sleeping status of the `GTV`)
- `gtview('Sleep')` and `gtview('Wake_Up')` (they were setting the sleeping status of the `GTV`).

The following obsolete entry points are removed:

- `gtpaus`
- `gtg_charsiz`
- `gtg_open`

5.2.2 New

Any segment opened with `gr_segm` must now be closed with the following new entry point after all the data it contains has been sent to the metacode:

- `gr_segm_close(error)`

The following conditions applies:

- at close time (`gr_segm_close`), a segment is implicitely drawn by `GTV`,
- attempting to send segment data to the metacode while there is no segment currently opened is an error,
- attempting to open (`gr_segm`) a new segment while the previous one is not closed (`gr_segm_close`) is an error,
- attempting to close (`gr_segm_close`) twice a segment is an error.

Given the first point above, the necessity for an `Append` mode for `gtview` is severely reduced. Basically, what you will probably have to do is to replace all the statements `gtview('Append')` by `gr_segm_close(error)`. The only purpose to call `gtview('Append')` is now when you are filling a unique segment, and want the user to see in real-time this segment being filled (e.g. a curve showing the convergence of a computation, or an interactive drawing with the cursor, ...).

Instead of calling `gr_exec('CLEAR SEGMENT FOO')`, programmers can invoke the public subroutine `gt_clear_segment(name,present,error)` where `name` is the segment(s) name to be destroyed (e.g. if `name` is `FOO` it will destroy all the segments named `FOO:*`), `present` is a logical indicating if an error should be raised if no segment with such a name is found, and `error` is an `inout` logical error flag (i.e. it keeps its entering value if no error occurs).

A Exhaustive description of the changes in GTV

A.1 Commands

A choice has been made to clarify the various commands. In particular, there were some keywords which were refering to some obsolete devices (e.g. `CLEAR ALPHA` and `CLEAR GRAPHIC`).

There was a also a confusion with the behavior command `CLEAR`. The rule is now: `CLEAR FOO` erases the content of the object designated by `FOO`, while the new command `DESTROY` kills this object.

A.1.1 Removed

- `CHANGE CLEAR TREE|WHOLE, DISPLAY CLEAR`
  - commands were controling and displaying the behavior of `CLEAR PLOT`
  - reason: associated `CLEAR TREE` and `CLEAR WHOLE` are removed.
- `CHANGE ZOOM NEW|CURRENT`
  - command was controling the default behavior of `ZOOM`, i.e. if the zoom should be performed in the main or in a new window.
  - `CHANGE ZOOM` is removed since the zoom is now always performed in a new window.
- `CHANGE DRAW ON|OFF`, and `GREG1\SET DRAW ON|OFF`
  - commands were both disabling (`OFF`) the real-time drawing of the new segments being added to the metacode, and drawing them (`ON`) later on.
  - reason: associed mode is removed in the Fortran API (see comments on `gtview` at section 5.2.1.
- `CLEAR GRAPHIC`
  - command was supposed to bring to front (or focus) the terminal (`CLEAR GRAPHIC` means *hide the graphic window*).
  - reason: not supported on any known modern window managers + too many keywords accepted by `CLEAR` + keywords have no meaning for those who did not know the old devices like Tektro.
- `CLEAR PLOT`
  - reason: there was a confusion between plot clearing and windows destruction.
  - in the simplest case, you have one window, and you are working in the top directory (probably `<GREG`): just type `CLEAR` to erase the content of the current directory and clear the window.
  - if you have several windows: i) if you want to *clear* one window, type `CLEAR TopDir` where `TopDir` is the directory associated to the window you want to clear, ii) but if you want to *kill* one directory and its associated window, type `DESTROY TopDir`
  - finally if you want to reset all your plots i.e. get back the startup status, type `DESTROY ALL`.

- `CLEAR TREE`
  - reason: too many keywords associated to the command `CLEAR`, can be replaced in the new GTV context
  - use `CLEAR DIRECTORY TopDir` instead, or `CHANGE DIRECTORY` (with no argument, it goes to the top directory) + `CLEAR DIRECTORY` (with no argument, it erases the content of the current directory).
- `CLEAR WHOLE`
  - reason: too many keywords associated to the command `CLEAR`, command unused
  - use `CLEAR` without argument instead
- `CLEAR WINDOW [Num]`
  - reason: name is ambiguous, too many keywords associated to the `CLEAR` command
  - use `DESTROY WINDOW [Dir [Num]]` instead. No way to destroy (from the command line) the current window only. For the old `CLEAR WINDOW ZOOM`, use `ZOOM OFF` or `DESTROY WINDOW ZOOM` instead.
- `CREATE DIRECTORY|WINDOW /PIXEL`
  - option `/PIXEL Nx Ny` is removed. It was overiding the default size of the new window opened by the command.
  - reason: new option `/GEOMETRY` does the same and more (see detailed description hereafter)
  - use `CREATE DIRECTORY|WINDOW /GEOMETRY Nx Ny` for an identical result.
- `CREATE DIRECTORY /SIZE`
  - option `/SIZE` is renamed `/PLOT_PAGE` for symmetry with command `GREG\SET PLOT_PAGE`) which does the same afterwards. These commands define the physical size of the plotting area of a top directory.
- `HARDCOPY /PLOT`
  - option `/PLOT` for command `HARDCOPY` is renamed `/PRINT`.
  - reason: nowadays we use printers, not plotters.
- `GREG1\SHOW DRAW`
  - command was displaying the current real-time or delayed drawing status.
  - reason: associated command `GREG1\SET DRAW ON|OFF` is removed.
- `ZOOM REFRESH`
  - command is renamed `REFRESH`
  - reason: we may want to redraw any kind of window, not only zooms.
- `ZOOM /NEW|/CURRENT`

  - options were indicating if the zoom had to be performed in the main or in a new window
  - these options are removed since it not possible anymore to zoom in the main window (i.e. `/NEW` is implicit now)

- `ZOOM /REGION`
  - reason: not available on MS Windows, unused, duplicate of the new lens.

- `ZOOM buttons`
  - "A" (Clear alphanumeric screen) is removed
  - reason: same as `CLEAR ALPHA`

A.1.2 Changed

- `CHANGE WINDOW WinNum`
  - change the active window to the given window number. The active window is the one in which the interactive cursor will appear when it is invoked.
  - window numbering starts now from 1 (Fortran like) instead of 0 (C like). So in most of the cases the main window of the current directory has now number 1, and the zoom window (if any) has number 2.

- `CLEAR`
  - was performing a `CLEAR PLOT`
  - now performs a `CLEAR ALL` (see after).

- `CLEAR ALPHA`
  - commands is renamed (new name to be defined). It is supposed to bring to front (or focus) the graphic window (`CLEAR ALPHA` means *hide the alphanumeric terminal*).
  - reason: too many keywords accepted by `CLEAR` + keywords have no meaning for those who did not know the old devices like Tektro.
  - remark: this feature is poorly supported on Linux window managers (too many applications must have excessively used this, so that the window managers are very restrictive now).

- `CLEAR DIRECTORY [DirName]`
  - the old behavior was to make the directory and its content invisible for later destruction by `COMPRESS`.
  - command behaves differently: `CLEAR DIRECTORY DirName` empties the directory named `DirName`. As a consequence, this clears the windows seeing this directory. Without argument the content of the current working directory is removed. Segments are now really deleted.
  - reason: need a command which clears the content of a window i.e. of a top directory (usually). Need also to minimize the number of keywords accepted by `CLEAR` for clarity.

  - the old behavior (destruction of the directory) is ensured by the new command `DESTROY DIRECTORY DirName`.

- `CLEAR SEGMENT`
  - without argument, command was deleting the last segment created *in the metacode*.
  - without argument, command behaves slightly differently: it deletes the last created segment *in the current working directory*, which can be different from the old behavior.
  - reason: the metacode is now growing dynamically and the segments have lost their "temporality". Actually the new behavior seems better than the previous one.
  - no way to mimic exactly the old behavior
  - NB: `CLEAR SEGMENT Name` behaves as before, except it really deletes the segment now (in the past it could have only made the segment invisble for later deletion by `COMPRESS`).

- `METACODE IMPORT|EXPORT`
  - Backward compatibility is broken.

A.1.3 New

- `CHANGE POSITION`
  - support for `Offx[Unitx] Offy[Unity]` is added. See `CREATE WINDOW /POSITION` above for the syntax.

- `CLEAR ALL`
  - clear the content of all the top directories, which clears the content of their windows. Top windows are not destroyed.

- `CREATE WINDOW [/GEOMETRY] [/POSITION]`
  - new options `/GEOMETRY` and `/POSITION` are added:

```
CREATE WINDOW [/GEOMETRY Sx[u] Sy[u]]
              [/POSITION Offx[u] Offy[u]]
```

    where `Sx` and `Sy` are the window size, `Offx` and `Offy` the offset on the screen. "u" (unit) can be "p" for pixels, "%" for percent of the screen, or "r" for the aspect ratio in unit of the other dimension. Default unit is pixel.
  - if `/GEOMETRY` is omitted, the values are read from the logical `WINDOW_GEOMETRY` defined in user's ∼`/.gag.dico` with the same syntax as above. If the logical `WINDOW_GEOMETRY` is not defined, internal defaults are used for the window size.
  - if `/POSITION` is omitted, the choice of the window position is let to the window manager. "r" unit is a non-sense and is forbidden for this option.

- `CREATE DIRECTORY [/GEOMETRY] [/POSITION]`
  - new options `/GEOMETRY` and `/POSITION` are added: they apply to the creation of the window of a new top-directory. See `CREATE WINDOW` above for the syntax.

- `DESTROY DIRECTORY DirName`
  - destroys the input directory and its windows attached. Destroying `<GREG` is allowed (as long as user has moved before to another directory).
- `DESTROY WINDOW [DirName [WinNum]]`
  - destroys all the windows attached to the directory `DirName`, i.e. should be all the windows named `DirName Number`. Destroys all the windows of the current working directory by default. If a window number is given, only this window is destroyed.
- `DESTROY ALL`
  - destroys all the top directories and their windows
  - since the user can not be in the root directory (`<`) and can not create segments in it, he would have to create right after a new top directory. A first top directory is thus created implicitely.
  - As a consequence of these two points, the command resets the tree and windows to the startup status.
- `HARDCOPY /DIRECTORY DirName`
  - this new option allows to plot the given GTV directory. Default remains the current working directory.
- `HARDCOPY /FITPAGE`
  - this new option allows to adjust at best the plot on the hardcopy physical size. This is suited for EPS harcopies which are not automatically rotated and scaled anymore.
  - this option is exclusive with the historical option `/EXACT` (which disables the automatic scaling).
- `LUT Name /REVERT`
  - the look-up-table `Name` is loaded and reverted on the fly.
- `REFRESH [DirName [WinNum|ZOOM]]`
  - redraw all (default) or some of the windows (identified by their directory and number/name).
  - Note that all windows are always up-to-date (drawn in real-time thanks to multi-threading). The need of this command is thus very limited. It makes sense only when using segments at different depths (e.g. images) which may be redrawn in an order different from the first drawing (first drawing is done in chronological order in order to remain efficient).
- `GREG1\PEN [Ipen] /DASH 8`
  - an 8th dashing style is added. It contains regular dashes, shorter than the 2nd style dashes, and larger than the 3rd (dot) style. Execute `@ gag_demo:demo5.greg` for an overview of all the pen attributes.

A.2 Variables

A.2.1 Removed

The following GTV variables are removed:

- GTV%SLEEP: the real-time or delayed drawing status, but this functionality is removed.
- GTV%IMAGES[2,15]: size and memory address of the images stored in the metacode, for debugging purpose.
- GTV%IDENT: the number in Terminal Definition File of the device currently in use.
- the LUT array is removed: it was describing the hue values of the current Look-Up Table in use. But since the hue is not enough to describe a LUT (saturation and value are also needed), the two triplets (LUT%RED, LUT%GREEN, LUT%BLUE) or (LUT%HUE, LUT%SATURATION, LUT%VALUE) should now be used to get and set the Look-Up Table values.
- REXTR: the extrema (in plot_page units) of the last segment created in the metacode.
- REXTR_D: the extrema (in plot_page units) of the current working directory.

A.2.2 Changed

The look-up table related variables are merged in a new structure named LUT%. See table 3 for details. This structure has three major groups of variables related to the images look-up table (under LUT%), to the blanking color (under LUT%BLANKING%) and to the pen look-up table (under LUT%PEN%).

The variables describing the current RG image plotted by the command GREG2\PLOT are merged under the structure CURIMA%. The equalization mode, if in use, is described under the substructure CURIMA%EQUAL%. See table 3 for details.

A.2.3 New

- GTV%PWD: this new variable contains the current working GTV directory as a character string. It is aimed to help writing plotting procedures which can work in the directory they are started in without altering the other plots (trees) already created by the user.
- GTV%FITPAGE: if YES, implies an implicit /FITPAGE option for the command HARDCOPY. In particular, this enables the historical behavior regarding Encapsulated PostScripts (if landscape, they will be rotated to match the portrait print page). Default is NO.
- GTV%STRICT2011: this new variable will be available *temporarily* to rule how the interpreter should behave when encountering an old GTV command. If its values is “yes”, an error is raised. If “no”, it is only a warning. In both cases an explanatory message is displayed. Default value is “yes”. This variable is present to help the users and programmers to speed up the transition, but without breaking abruptly their procedures and programs.

A.3 SIC logicals

A.3.1 Removed

- `LUT_DIR:`
  - logical is removed. It was controlling the directory where the LUT files have to be searched in.
  - reason: new logical `LUT#DIR:` is better.

A.3.2 New

- `GAG_LUT:`, `LUT#DIR:`
  - new logical `GAG_LUT:` is the path to the internal LUT files. User should not change it.
  - new logical `LUT#DIR:` is *a collection* of paths where the LUT files should be searched in. User can customize it as its will. Default value is `./;GAG_LUT:;`.
- `WINDOW_GEOMETRY`, `WINDOW_POSITION`
  - if defined, these new logicals control respectively the size of any new window, and the position on the screen of the first window when the device is opened.
  - user can define (one of) them in its ∼`/.gag.dico` for all its sessions, or later during a session.
  - the option `/GEOMETRY` of the command `CREATE DIRECTORY|WINDOW` override the values in WINDOW_GEOMETRY.
  - see `CREATE DIRECTORY|WINDOW` above for the syntax to be used.

A.4 Symbols

A.4.1 Removed

- `CD`
  - reason: should not be provided by GREG
  - use `CHANGE DIRECTORY` or define the same symbol instead
- `PWD`
  - reason: unused / should not be provided by GREG
  - use `DISPLAY DIRECTORY` or define the same symbol instead
- `MKDIR`
  - reason: should not be provided by GREG
  - use `CREATE DIRECTORY` or define the same symbol instead