National Radio Astronomy Observatory Tucson, Arizona

To: Observers

From: Simon Radford, Jeff Mangum, Mark Gordon

Subject: Gildas Software at the 12 m telescope

December 13 1995

A Little History

The Gildas software is produced by IRAM and the Groupe d'Astrophysique de Grenoble. The package includes Class, for reducing (mostly) spectroscopy data, Greg, a general plotting program, Graphic, for image construction and processing, and Astro, for preparing observations. At the NRAO 12 m telescope, Gildas is available for data post processing. Class is not directly linked to the on-line system, so you must manually convert your data to Class format before you can use Class to reduce your data.

NRAO offers only limited support for Class. Gildas is developed and maintained entirely by IRAM and GAG, so general questions about the software should be directed to them. Class was originally installed at the NRAO 12 m telescope by Phillipe André in May 1992. Questions about the local installation may be directed to the Class advisor at `gagxprt@heineken.tuc.nrao.edu`.

Getting Your Data Into Class

The on-line system at the NRAO 12 m telescope writes the data in the SDD (UniPOPS) format. There is a utility program, `uni2class`, that takes all the scans in an SDD file, translates them, and writes them to a Class file. To use this utility, simply supply the file names either in response to the prompts or on the command line. Examples:

```
% uni2class
SDD filename:  test.sdd
Class filename:  test.12m
Entry 1024 beyond end of file

% uni2class test.sdd test.12m
Entry 1024 beyond end of file
```

The `end of file` message is normal. You can convert all the data from an observing session with:

```
% uni2class /home/data/data/abc/sdd.abc_001 abc.12m
```

for the filter bank data and

```
% uni2class /home/data/data/abc/sdd_hc.abc_001 abc.12m
```

for the correlator data, where `abc` are your observing initials and `001` is the file sequence number, usually `001`. You can also convert UniPOPS SAVE and KEEP files with `uni2class`.

Class and `uni2class` can access the data file at the same time, i. e., Class can read it at the same time `uni2class` is writing it. This suggests it wouldn't be too hard to make `uni2class` into a quasi-real-time translator.

`Uni2class` will overwrite existing output files without warning. Be careful.

Peculiarities
Because the SDD and Class headers don't map onto each other exactly, only a subset of the parameters are carried over. Furthermore, some parameters aren't defined the same way, so the conversion involves some interpretation.

Subscan numbers
In SDD files, the receiver and backend configuration for each scan is indicated by a subscan number (Table 1). Each telescope scan may result in several subscans. Class doesn't support subscans, however, but does differentiate beween the scan numbers assigned by the telescope and observation numbers, which are unique, but otherwise arbitrary, keys to the data in a file. Again, each telescope scan may result in several observations. Hence `uni2class` sequentially assigns unique observation numbers to each spectrum in the Class output file. The original telescope scan number is preserved in the header. To distinguish between the different subscans, the UniPOPS Backend descriptor, which contains the subscan number, is copied into the Class Line field. This is not ideal, perhaps, but it avoids ambiguities.

TABLE 1

SUBSCAN CODES FOR DATA SCANS

Subscan Number	Parallel Configuration	Series Configuration
	Filter Banks	
01	IF 1 & First half of filter bank 1	IF 1 & Filter bank 1
02	IF 1 & Second half of filter bank 1	
03	IF 2 & First half of filter bank 2	IF 2 & Filter bank 2
04	IF 2 & Second half of filter bank 2	
	Hybrid Spectrometer	
11		IF 1
12		IF 2
...		...
18		IF 8

Other parameters
Map offsets are carried over. There seem to be small ($< 1''$) discrepancies in times and coordinates between the SDD and Class files. Perhaps roundoff error?

Only LSR and heliocentric velocities, equatorial coordinates, and the standard small field projection are handled correctly.

No distinction is made between radio ($\Delta\nu/\nu$) and optical ($\Delta\lambda/\lambda$) velocity definitions.

The Class frequency offset (`Foff`) parameter is set to zero, since I'm unsure of its interpretation.

It's not clear if bad channels are properly flagged.

Continuum data are not converted.

Starting Class

Now you have a Class format file with your spectra. How do you reduce the data? While we'll give you some starting pointers, completely describing Class is beyond the scope of this memo. The Class manual is, howver, on line. Sections 1, 2, and 3 (pages 7–25) provide a decent introduction if you have never used Class. Either type `DOC` to view the manual on the screen or you can print the postscript file:

```
/usr/local/gildas/exe/doc/class.ps
```

To start Class, type `class`. If it asks,

```
remove /home/sun/obs/class.mes ?
remove /home/sun/obs/class.log ?
```

respond `yes`. Class will then supply a prompt:

```
LAS>
```

At this, call the `proc:start` macro (see below), which sets some useful defaults:

```
LAS> @proc:start <cr>
```

This macro also opens an X window for graphics. To print out plots, type

```
LAS> hardcopy/plot <cr>
```

and the plot will be automatically sent to the QMS1725 in the control room.

Because Class incorporates a database manager, it provides criteria to select your spectra to allow easy manipulation of a large volume of data. This facility is particularly convenient for reducing mapping observations. For instance, at any time during your run at the telescope, it may be used to take a quick look at all the spectra you have obtained for a particular source (cf. "How to produce a map of 12 m spectra with Class" in the following memo).

Source selection

To avoid problems with internal conversion from lower to upper case and vice versa, enclose the source name in quotation marks when selecting scans:

```
LAS> set source "M345" <cr>
```

Macros

Data processing will be much easier if you use macros and learn how to develop them. There are some example macros in `usr/local/gildas/lcl/proc`. You can copy these to your working directory or access them from within Class prefixing `proc:` to the macro call (see example below). You can also create your own macros with any text editor. Conventionally, they are named "mymacro.class", where "mymacro" is the name used to call it from within Class. All the Gildas programs are based on the Sic command interpreter. This interpreter offers a more or less complete programming language you can use to develop sophisticated data processing routines. Consult the Sic manual (also on line) for further details and look look at the examples in `/usr/local/gildas/exe/pro` (`pr:` within Class).

A Class startup macro

```
!  /usr/local/gildas/lcl/proc/start.class
!
!  Set up the Class parameters for running on bohemia.
!
!  Installed on Bohemia 11/20/93 (Jeff Mangum; jmangum@as.arizona.edu)
!  revised 12/6/95 sr
!
sic verify on
sic help page
device xauto
set format long
set ext .12m
set format origin on
set angle s
set unit v f
set plot histo
set align velocity
!
!  Set up some Class abbreviations
!
symbol st "set telescope"
symbol smx "set mode x"
symbol smy "set mode y"
symbol fi "file in"
symbol fo "file out"
symbol ss "set source"
symbol sw "set window"
symbol sc "set scan"
symbol sn "set num"
symbol sl "set line"
symbol so "set offset"
symbol scon "set cursor on"
symbol scof "set cursor off"
symbol su "set unit"
symbol sma "set mask"
```

A Sample Class session

```
@proc:start
file in spectra
find/all
get first
plot
set window -5 10
file out spectrab new
@proc:baseit
file in spectrab
set mode x -10 15
set mode y -3 7
set plot normal
```

```
map match/grid/number
hard/pl
file out spectras new
@proc:addspe
file in spectras
smx -10 15
smy -1 7
map match/grid
hard/pl
exit
```

Other programs

Besides Class, Gildas includes Greg, for general purpose two dimensional plotting, Graphic, for image processing, and Astro, which can precess coordinates, plot rise–set charts, predict planetary positions and fluxes, calculate *uv* coverages, and several other useful things. There are also programs for interferometer and bolometer data reduction. These programs are all described in their (on line) manuals and are started the same way, namely typing the program name, e. g. `greg`, at the system prompt. They then respond with their own prompts, e. g. `GreG>`. All the programs are based on the Sic command interpreter, so they have similar syntax.

This memo expands upon a series of earlier memos by P. André, M. A. Gordon, and J. Mangum.

Copies:

D. Emerson
T. Folkers
R. Freund
M. Gordon
J. Hagen
P. Jewell
J. Mangum
W. Latter
J. Neighbours
S. Radford

file: /usr/lcl/gildas/lcl/gaginfo.tex