IRAM Memo 2015-4
Introducing Associated Arrays in **CLASS**

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

1. IRAM (Grenoble)
2. Observatoire de Paris

July, 12^{th} 2016
Version 1.2

Abstract

The concept of Associated Arrays is being added in the **CLASS** data format. It is an additional section that will store a set of data arrays whose spectral axis matches the brightness spectral axis, *i.e.*, one-to-one channel correspondance. At start, this will be used in two contexts. For Herschel/HIFI, it will store a logical array of blanked channels and a logical array to define the channels where the lines appear (see IRAM Memo 2015-3). At some point, it will be used at the IRAM-30m to store the results of the calibration scans, *i.e.*, the sky counts in the `RY` array and the hot/cold counts, and the receiver, system, and calibration temperatures in a set of Associated Arrays.

This document introduces the detailed specifications of the concept of an Associated Array, its implementation/integration in **CLASS**, and it gives an example of use, namely how to baseline a spectra using a `LINE` associated array to define the baseline windows. The specifications of the Associated Arrays is foreseen to evolve following feedback from its use at IRAM and elsewhere.

Keywords: **CLASS** Data Format
Related documents: **CLASS** documentation, Importing Herschel-FITS into **CLASS**

Contents

1 Nomenclature reminder

It is reminded that:

1. A **CLASS** observation is used to store and retrieve a flux array for analysis into **CLASS**. This flux array is called `RY` hereafter.
2. In a **CLASS** observation, there are 3 ways to define the X axis associated to `RY`:
   (a) the spectroscopic section for regularly sampled (in frequency and velocity) spectroscopic data,
   (b) the drift section for regularly sampled (in time and angular position) continuum drift data,
   (c) the so-called "X coordinate section" for irregularly sampled spectroscopic or continuum drift data.

   One of these 3 sections is used to define the `RX` array describing the X axis associated to `RY`. In the cases (a) and (b), `RX` in memory is built so that it has as many values as `RY`. In the case (c), the **CLASS** Data Format ensures that `RX` has as many values as `RY`. For simplicity hereafter, we call `RX` the X axis coming from one of these sections, and `Nchan` its number of values.

2 Specifications

The concept of Associated Arrays is introduced in **CLASS**. They are specified as follows:

1. The Associated Arrays section is added to store 1 or more arrays of data directly *related* to `RY`.
2. The Associated Arrays must be of size `Nchan` (1st dimension). A second dimension can be used. Their X axis associated to the 1st dimension is **exactly** described by `RX`. In other words, there is a one-to-one channel relationship between the flux array `RY` and any of the Associated Arrays. This also means that any modification on the spectroscopic section has a similar and consistent impact on `RY` and on the Associated Arrays. This includes resizing the axis (`EXTRACT`) and shift/stretch of the axis (`MODIFY VELO|FREQ`).
3. The type and kind of data stored as an Associated Array is flexible, in the limit of what the Gildas internal engines are used to (*i.e.* numeric data types). Expected types are integer or real (floating point) types. See details in Table 1.
4. An Associated Array is identified by its name (12 characters maximum). It must be unique in one observation. The name may contain only characters valid for Sic variable names (typically the alphanumeric character set), so that the arrays can also be used as Sic variables. The Associated Array may also provide a unit string (12 characters maximum), left blank if not used. And it must provide a *bad* (blanking) value, valid in the supported range of values corresponding to the data format.
5. There is no predefined/limited list of supported Associated Arrays. Some may be built and saved by the **CLASS** internal engines with a precise definition (so-called Reserved Associated Arrays, see section 4), others may be created and saved by the user on his own choice. Examples could be:
   - a weight array
   - a mask array
   - wavelets
   - fitted model and residuals
   - etc.

One typical counter example which does not satisfy the rule in item #2 is:

- the Fourier transform: its X axis (inverse frequency) is not described by the spectroscopic section, `EXTRACT`ing the same subset of `RY` and FFT would be incorrect (the FT of a subset is not the subset of the FT).

6. Resampling and its other flavors (`RESAMPLE`, `AVERAGE`, ...) is a particular problem since **CLASS** has to know how the values can be *mixed* (*e.g.* what is the meaning of resampling a flag array?). There is no straightforward answer. Basic ideas are:
   - the usual `RY` resampling engines can be opened for all floating point arrays, and can be extended to integers (floating point averaged then rounded back).
   - a nearest neighbour algorithm could also be offered (may be better suited for integers like boolean arrays).
   - for Reserved Associated Arrays, the **CLASS** programmer can offer custom resampling engines, suited exactly for a given array,
   - one can think in *hooks* so that the users can describe how the resampling works. Implementation is unclear at this stage.
   - if **CLASS** does not have any solution for resampling, the ultimate solution is to remove the Associated Array. Leaving it non-resampled would violate the rules in item #2.

   As of today, **CLASS** uses the first proposition as a generic solution.

Table 1: Type and kind of data supported for Associated Arrays. Note that 4-bits and 2-bits integers are provided for storage on disk with low space comsuption. In particular, 2-bits integers provide 4 possible values: they are well suited for storing a boolean (0 and 1) and a bad value (e.g. -1). Note also that in practice, integers smaller than 32 bits are always loaded as INTEGER*4 in memory, so that Sic variables can be mapped on them and the **GILDAS** tools can be used.

Internal code	Equivalent Fortran type and kind	Nbits (disk)	Range of values	Type and kind in memory
fmt_r8	REAL*8	64		REAL*8
fmt_r4	REAL*4	32		REAL*4
fmt_i8	INTEGER*8	64	-2^{63} to $+2^{63}-1$	INTEGER*8
fmt_i4	INTEGER*4	32	-2^{31} to $+2^{31}-1$	INTEGER*4
fmt_i2	INTEGER*2	16	-2^{15} to $+2^{15}-1$	INTEGER*4
fmt_by	INTEGER*1	8	-2^{7} to $+2^{7}-1$	INTEGER*4
fmt_b4	N/A	4	-2^{3} to $+2^{3}-1$	INTEGER*4
fmt_b2	N/A	2	-2^{1} to $+2^{1}-1$	INTEGER*4

3 Implementation details

The Associated Arrays support was officialy introduced in the oct15 version of **CLASS**, together with the Herschel-HIFI FITS reader. They are available in a new section (identifier code -19) in the **CLASS** observation header. The size of this section is variable, as there can be several arrays in the section, and the size of each array is the same as `RY`. The section provides the number of arrays stored, then for each array its description and its data. See Table 2 for details.

A basic example using 2 arrays is shown in Appendix A.

Table 2: Description of the section Associated Arrays in the **CLASS** observation header. Each array is described by its format, 2nd dimension, name, and unit. The *bad* (blanking) value is stored in the same format as the data, then the data array itself follows. Order matters here.

	Name	Format (disk)	Description
Once in the section	`N`	INTEGER*4	Number of arrays
Repeated per array	`NAME`	CHARACTER*12	Array name (identifier)
”	`UNIT`	CHARACTER*12	Unit (blank if not relevant)
”	`DIM2`	INTEGER*4	Second dimension (0 for 1D array)
”	`FMT`	INTEGER*4	Data format (internal code)
”	`BAD`	FMT	Blanking value
”	*DATA*	FMT	The data array (Nchan values, times `DIM2`)

4 Reserved Associated Arrays

The Associated Arrays are a generic concept offered to store any kind of numeric data with minimal restrictions. However, **CLASS** will be the first one to offer such arrays. This will be done by reserving some array names. The reserved arrays have a strict definition and, in some cases, **CLASS** knows how to use them directly.

4.1 Pre-reserved array names

The names `X, Y, C, V, F, I, T` and `A` are pre-reserved because they refer to the `RX` or `RY` arrays in the different possible units in **CLASS** (command `SET UNIT`). The use of these names is restricted in order to avoid confusion. In the future they could also be used as real arrays saved on disk.

4.2 Reserved arrays

As of today, **CLASS** reserves with a special meaning the arrays named `BLANKED` and `LINE`. The name `W` is also reserved for use as a channel-based weight array. See Table 3 for their specifications.

Table 3: Specifications of the Reserved Associated Arrays.

Name	Format (disk)	Dim2	Unit	Bad	Description
`W`	fmt_r4	0		-1000	Weight array
`BLANKED`	fmt_b2	0		-1	Flag for bad channels
`LINE`	fmt_b2	0		-1	Flag for spectral line

4.2.1 W

The `W` Associated Array is a single-precision floating point array providing a channel-based weight array of the spectrum. It can be used as input and output of the `AVERAGE` command (and its other flavors) thanks to the switch `SET WEIGHT ASSOC`. The user can define the weight array of his choice for the input spectra thanks to a sequence like:

```
FILE IN ...
FIND
FILE OUT ... SINGLE
FOR I 1 TO FOUND
  GET NEXT
  ! Fill MYARRAY as desired here
```

```
  ASSOCIATE W MYARRAY
  WRITE
NEXT I
```

and then reuse the output file as input for `SET WEIGHT ASSOC + AVERAGE`.

4.2.2 BLANKED

The `BLANKED` Associated Array is an integer array of 0 or 1 indicating if the `RY` data should be flagged out (1) or not (0). The purpose is similar to the bad values in `RY`, except that the *actual* (non-bad) values are still available by default, and it is the choice of the user to apply the `BLANKED` flags or not[1]. The user can apply these flags to `RY` *e.g.* with the following command:

```
LAS> LET RY R%HEAD%SPE%BAD /WHERE R%ASSOC%BLANKED%DATA.EQ.1
```

4.2.3 LINE

The `LINE` Associated Array is an integer array of 0 or 1 indicating if the `RY` channels are in (1) or out (0) a spectral line, i.e. in signal or noise windows. This array is intended to be used by the command `BASE` in order to exclude the given channels. Here is the specific integration in **CLASS**:

- if `SET WINDOW` is invoked with the option `/ASSOCIATED`, the command `BASE` will ignore the channels where the `LINE` array is 1. The usual windows (defined by their min-max range) are not used in this context. As a result, the element `R%HEAD%CAL%NWIND` is set to -1 (code for "Associated Array `LINE` was used") in the baseline section.
- if `SET WINDOW` is invoked with the argument `AUTO` (*i.e.* take windows from data), and if the element `R%HEAD%CAL%NWIND` is -1, the `LINE` Associated Array will be used by the command `BASE`.
- `SHOW WINDOW` will display a specific message if the current `SET WINDOW` tuning refers to the Associated Array `LINE`.
- `DRAW WINDOW` will plot the Associated Array `LINE` if the current `SET WINDOW` tuning refers to the Associated Array `LINE`.

5 Generic integration in CLASS

We describe here what are the generic tools supporting the Associated Arrays in **CLASS** and how the end-user can interact with them.

- The command `ASSOCIATE` allows the user to attach or detach an Associated Array to/from the current R buffer. Some other commands may also attach Associated Arrays implicitly, *e.g.* the Herschel-HIFI FITS reader.
- The command `DUMP` can be used for a quick overview of the Associated Arrays attached to the `R` buffer, if any.
- The Associated Arrays are also instantiated in the Sic structure `R%ASSOC%` (read-only) for direct use by the end-user. For each array a sub-structure of the same name is provided, with the form:

```
R%ASSOC%NAME%DIM2 =         0      ! Integer GLOBAL RO
R%ASSOC%NAME%UNIT =                ! Character*12 GLOBAL RO
R%ASSOC%NAME%BAD =       -1        ! Integer GLOBAL RO
R%ASSOC%NAME%DATA is an integer Array of dimensions  8257
```

[1] This array was actually introduced in the context of Herschel-HIFI data filler to **CLASS**.

where `NAME` is to be replaced by the array name.

- The arrays can be plotted thanks to the commands `PLOT` and `SPECTRUM`, passing their identifying name as first argument.
- The command `EXTRACT` (spectrum and index mode) supports extracting a subset of `RY` and the same consistent subset of all Associated Arrays.
- The commands `SMOOTH`, `RESAMPLE`, and `FOLD` apply the same exact engines on the Associated Arrays as the ones which are used for RY. This is straightforward for floating point Associated Arrays (reasonable solution for most physical quantities). For an integer Associated Array, the values are smoothed (resp. resampled) as floating point values and then rounded back to nearest integer (reasonable solution for integer physical quantities, e.g. counts). This solution is generic but may not be satisfying in all cases (e.g. flag arrays). No custom solution is possible yet.
- `AVERAGE` and its other flavors (`ACCUMULATE`, `STITCH`, `RMS`) support averaging the Associated Arrays of the current index, as long as all the observations have the same number of Associated Arrays, with the same name and in the same order. They are also averaged using a floating point engine as a generic solution (see above). All arrays are averaged except `W`, as it makes no sense to average the weight arrays (the resulting weight array is actually saved if `SET WEIGHT` is `ASSOC`.
- Finally, some commands may interpret a specific Associated Array when one is found. See the related Associated Array description for details.

6 Evolution

This document describes the first version of the Associated Arrays. However, the **CLASS** team evaluates the feasibility of a new version of the arrays in order 1) to provide a more efficient access to the arrays, and 2) to ensure a better integration of the arrays in **CLASS**.

A Associated Arrays section example

```
N                [I*4]  = 2  ! 2 arrays in the section

### Array 1 ###
  # Description
  FMT(1)         [I*4]  = fmt_b2  ! i.e. 2-bits integer
  DIM2(1)        [I*4]  = 0       ! i.e. array is 1D of size Nchan
  NAME(1)        [C*12] = "LINE"
  UNIT(1)        [C*12] = ""
  BAD(1)         [B*2]  = -1      ! 2-bits value
  # Data
  DATA1(1)       [B*2]           ! 2-bits-value
   ...
  DATA1(Nchan)   [B*2]           ! 2-bits value

### Array 2 ###
  # Description
  FMT(2)         [I*4]  = fmt_r4
  DIM2(2)        [I*4]  = 2          ! i.e. array is 2D of size Nchan x 2
  NAME(2)        [C*12] = "WAVELET"  ! All wavelet orders in a 2D array
  UNIT(2)        [C*12] = ""
  BAD(2)         [R*4]  = -1000.0
  # Data (column-major)
  DATA2(1,1)     [R*4]
   ...
  DATA2(Nchan,1) [R*4]
  DATA2(1,2)     [R*4]
   ...
  DATA2(Nchan,2) [R*4]
```