MRTCAL Programmer Manual

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

Documentation
In charge: S. Bardeau\textsuperscript{1}.
Active developers: J. Pety\textsuperscript{1,2}, A. Sievers\textsuperscript{3}.

Software
In charge: S. Bardeau\textsuperscript{1}, A. Sievers\textsuperscript{3}.
Active developers: J. Pety\textsuperscript{1,2}.

1. IRAM (Grenoble)
2. Observatoire de Paris
3. IRAM (Granada)

Related information is available in:

- MRTCAL User Manual
- CLASS Manual
- GREG: Graphical Possibilities
- SIC: Command Line Interpreter
Contents

1 Chunks and chunksets .......................... 1
   1.1 Chunks slicing .................................. 1
   1.2 Polarimetry .................................. 4

2 Reading the DATA column .................. 7
   2.1 Discarding bad time dumps ............... 7
   2.2 Dump cycle identification ............... 7
   2.3 Reading the DATA column by pieces ....... 8

3 Positions ...................................... 9
   3.1 Reference position ......................... 9
   3.2 Scan offsets .................................. 9
   3.3 Antenna offsets .............................. 9
      3.3.1 Projection system ....................... 10
      3.3.2 Interpolation .......................... 10
      3.3.3 Wobblor switching ..................... 11
      3.3.4 Slow and Fast traces ............... 11
   3.4 Offsets of multi-pixel receiver .......... 11
      3.4.1 Handling the derotator .......... 11
      3.4.2 Computing the pixel coordinates . 13
      3.4.3 Description of the radio projection 14
      3.4.4 In practice: Effect on Polaris observations 19
   3.5 Guessing ON and OFF positions .......... 24

4 Interpolation of calibration products .... 27

5 Memory index ................................. 29
   5.1 Strategy ..................................... 29
   5.2 Contents ................................... 29

6 Scan date vs observation date: Midnight issue 33

7 Patching IMB-FITS at read time .......... 35
   7.1 Added elements when missing ........... 35
   7.2 Modified elements ......................... 36
# Contents

8 IMB-FITS version 2.13

8.1 Primary .................................................. 37
8.2 IMBF-scan .................................................... 38
8.3 IMBF-frontend ............................................... 39
8.4 IMBF-backend ............................................... 40
8.5 IMBF-backendFTS ........................................... 41
8.6 IMBF-antenna ............................................... 41
Chapter 1

Chunks and chunksets

For the processing of the IMBFITS files, MRTCAL introduces the concept of chunks (see Fig 1.1). One dump of the IMBFITS DATA column is a collection of such chunks. MRTCAL can make no assumption on the way they are ordered (this comes from ???. The PART column indicates to which spectrum each chunk belongs (spectra are identified with integer numbers). Each “set of chunks belonging to the same final spectrum” is called a chunkset in the MRTCAL nomenclature. From one chunkset, MRTCAL will produce one spectrum.

The chunks in each DATA row can belong to several spectra (e.g., 8 for FTS), and by definition to as many chunksets. MRTCAL maps those chunksets internally as a chunkset_1D Fortran structure, i.e., an array of chunksets.

The DATA column has as many rows as the number of pixels of the receiver times the number of time dumps. This introduces 2 new dimensions which are used to map all[2] the DATA column into a chunkset_3D Fortran structure. The 2 new dimensions here are pixels and time (see Fig 1.2).

For its internal needs, MRTCAL has to refer to a single time dump in the 3rd dimension. It can also time-average this 3rd dimension. In those 2 use-cases, the final product is then available through a chunkset_2D Fortran structure.

1.1 Chunks slicing

In order to be able to make calibration at a bandwidth smaller than the hardware chunks, MRTCAL is able to map the raw data block read with CFITSIO into smaller memory chunks. The main advantage is that this approach has very well defined boundaries: When the IMBFITS BACKEND table is read, it can be replaced on-the-fly with its sliced version. All the rest of the code does not have to know this happened. The downside is that the more memory chunks are defined, the more overheads they carry (e.g., description of their spectroscopic axis, calls to chunk-by-chunk subroutines, stitching, etc). Experience shows that slicing the 1400 MHz of FTS chunks into 14 chunks of 100 MHz introduces a penalty of 10% in the whole computation time (reading, slicing, calibrating, writing).

In practice, user can request the bandwidth he wishes. In order to slice the USED channels into comparable pieces, MRTCAL will choose the nearest bandwidth which gives an integer number of slices

\[ N = \text{nint} \left( \frac{\text{USED} \times |\text{SPACING}|}{W} \right) \]  

For debugging purpose, MRTCAL can also produce one spectrum per chunk.

MRTCAL can also map a fraction of the DATA column, process it, and then iterate on the next fraction. This can be useful on memory-limited machines.
Figure 1.1: How one dump in the IMBFITS DATA column is divided into chunks, gathered by family (chunksets), and reordered to produce the spectra.
Figure 1.2: How all the dumps in the **IMBFITS** DATA column is divided into 3D chunksets. After this, they can be time-averaged, or can be accessed one by one.
where \( W \) is the requested bandwidth. \( N \) must also be corrected so that there is at least 1 and at most \( \text{USED} \) channels per subchunk

\[
\text{USED} \geq N \geq 1. \quad (1.2)
\]

While the \texttt{CHANS} value (total number of channels) is often a multiple of 2, \( \text{USED} \) channels have no special value. In particular, it may not be a multiple of \( N \). Their relationship can be written as

\[
\text{USED} = q \times N + r, \quad (1.3)
\]

where \( q \) and \( r \) are respectively the integer quotient and the remainder of the division \( \text{USED}/N \). Since \( 0 \leq r \leq N - 1 \), \( \text{USED} \) can be divided in \( N \) subchunks, \( r \) of them having one extra channel

\[
\text{USED} = [q \times (N - r)] + [(q + 1) \times r]. \quad (1.4)
\]

This gives the best channel division and a negligible difference of subchunks bandwidth.

\texttt{ZZZ} Say a word about the “famous” special channel with its half weight. Slicing does not have any special problem with it.

\section*{1.2 Polarimetry}

The presence of polarimetric measurements in the \texttt{IMBFITS} file is described in the \texttt{POLAR} column of the \texttt{BACKEND} table. For a non-polarimetric scan, all the chunks in the \texttt{POLAR} column have the \texttt{NONE} value. For a polarimetric scan, values can be \texttt{NONE}, \texttt{AXIS}, \texttt{REAL} or \texttt{IMAG} (note that non-polarimetric chunks -\texttt{NONE}- can be mixed with polarimetric ones).

In addition, calibration scans for polarimetric measurements must provide a \texttt{calGrid} subscan. If absent, \texttt{MRTCAL} will mark this scan failed as it can not be used to calibrate polarimetric science scans.
Figure 1.3: Hardware chunks can be virtually divided into smaller pieces just by redefining the \texttt{BACKEND} table. In this example, the table is sliced from 1 chunk to 3 subchunks (3 times more lines in the table). The same raw data block read with \texttt{CFITSIO} will be mapped by \texttt{Mrtcal} with a larger number of \textit{memory} chunks.
Chapter 2

Reading the DATA column

2.1 Discarding bad time dumps

The BackendDATA table provides the data dumps collected during the subscan integration. In particular, it shows the dump MJD, its integration time INTEGTIM, its phase number through cycles (ISWITCH) and the data itself (channels) in the DATA column.

It may happen that a dump is flagged out by the control system. In this case, its ISWITCH value is 0. Such a dump should be discarded during the calibration process. This is achieved with the command MSET CALIBRATION BAD YES|NO (default NO, i.e. discard bad time dumps).

In practice, this is not just a matter of skipping undesired rows when dealing with the DATA column. The first issue is that the values in the associated columns are unreliable. This is particularly true with the MJD value which is not set to the actual MJD of the dump. This introduces inconsistent values in this column, resulting in an unsorted column, breaking the dichotomic search engine. In order to solve this issue, MRTCAL compresses at read time the MJD, INTEGTIM, and ISWITCH columns, discarding the rows where ISWITCH is null. The size of those columns is decreased from $N_{tot}$ to $N_{good}$ (with $N_{tot} = N_{good} + N_{bad}$).

MRTCAL keeps also track of the discarded dumps thanks to two new columns added. One column named FOREPOIN (size $N_{tot}$) provides a forward pointer, i.e. $\text{FOREPOIN}(i)$ is the position of the $i^{th}$ dump (as found in the original columns) in the compressed columns. If the dump is bad, the corresponding position is set to 0 (i.e. dump not available in the compressed columns). It also adds a column named BACKPOIN (size $N_{good}$) which provides a backward pointer: $\text{BACKPOIN}(j)$ is the position of the $j^{th}$ dump (as found in the compressed columns) in the original columns.

At this stage, MRTCAL has patched the BackendDATA columns so that the bad dumps are removed, but this is not fully transparent for later use.

2.2 Dump cycle identification

From a general point of view, the dumps have to be calibrated by cycles, each cycle repeating the same sequence of several phases. In order to do this, MRTCAL identifies those sequences in the ISWITCH column. As this column may have been compressed, the identification also takes into account the BACKPOIN column: a cycle is valid if the expected sequence is found in the compressed column AND if the associated back pointers are contiguous in the original columns.
2.3 Reading the DATA column by pieces

MRTCAL is designed to be able to process the subscans within an arbitrary limited buffer size (MSET BOOKKEEPING SPACE VALUE, default 512 MBytes). If the buffer size is large enough, the whole DATA column can be loaded at once in memory and its dumps can be treated in a row. If the buffer is not large enough, the DATA column has to be load by pieces, and each piece has to be processed separately from the other.

In practice, the calibration routine asks the reading routine to load the piece of DATA which contains at least a desired range of dumps (usually a cycle of a few dumps). At least means that the reading routine ensures that the range will be present entirely in the buffer, but more dumps will also be loaded in order to fill at best the buffer. On subsequent requests, if the desired range is already present in the buffer, nothing will be reloaded, in order to save calls to CFITSIO and I/O accesses.

At this stage, one should consider that CFITSIO is asked to read a block of rows from the DATA column. CFITSIO is not able to skip some bad rows here and there, and it would be inefficient to ask for reading the desired rows one by one. So CFITSIO reads a piece of the DATA column, including good and bad dumps. But right after, MRTCAL sets a 3D chunkset ($N_{set} \times N_{pix} \times N_{time}$) pointing to the good dumps only. Then the calling routine has just to work with the 3D chunkset, with no need to care for bad dumps.
Chapter 3

Positions

3.1 Reference position

For each spectrum, the Class Data Format expects a reference (usually source) position. In MRTCAL, this reference is taken from the `LONGOBJ` and `LATOBJ` (Source longitude and latitude in basis frame) in the Scan HDU. The coordinate system of these values are taken from header cards `CTYPE1` and `CTYPE2` (RA and DEC for equatorial system, `GLON` and `GLAT` for galactic system). Note that the trailing characters describing a projection system (e.g. -SFL) are useless: they do not apply to any data which can be found in the Scan HDU.

3.2 Scan offsets

From the absolute reference defined at 3.1, the IMBFITS format can define an intermediate position, through the `SYSOFF`, `XOFFSET` and `YOFFSET` columns in the `SCAN` table:

- if only the Nasmyth row is present, an implicit projection row is added at read time with (0,0) offsets,
- if the projection row is available, the associated offsets are added to the antenna offsets (see next section) and used into the CLASS Data Format,
- if something else than projection is available, e.g. horizontalTrue, a specific support is not yet available in MRTCAL and an error is raised. However, if the associated offsets are (0,0), Mrtcal will tolerate this system of offsets. This is a temporary solution, waiting for a full support of all systems.

3.3 Antenna offsets

The antenna offsets are found in the `LONGOFF` and `LATOFF` columns in the Antenna Slow HDU. These antenna offsets are to be added to the scan offsets defined at 3.2. Note that these offsets describe the primary dish pointing position. In case of wobbler switching, the beam does not point to this position (see section 3.3.3 for details).

\footnote{This can happen e.g., if the user has commanded `OFFSETS 0.0000000E+00 0.0000000E+00 /SYSTEM "trueHorizon" in PAKO}
3.3.1 Projection system

The antenna offsets are expressed in a system described by the card SYSTEMOF in the Antenna HDU. If its value is:

- **projection**, this means the offsets are radio-projected in the current coordinate system, i.e. the system found at 3.1; the offsets can be used “as is” in CLASS,

- something else, e.g. horizontalTrue for calSky, the offsets are not expressed in the correct frame and should be recomputed to the desired sky coordinate system (CLASS does not offer the possibility to express the reference in one system and the offsets in another). ZZZ this is not yet implemented, warning for now.

3.3.2 Interpolation

The antenna offsets are derived from the Antenna Slow table together with other position values. Namely the 5 following elements have to be computed:

- lambda offset from reference position,
- beta offset,
- azimuth,
- elevation,
- local sidereal time.

All these elements are available in the ANTSLOW table, at a typical sampling rate of 1 Hz. Since the spectra dumps can be produced at different time sampling (no assumption is made on the rate or its variations), each spectrum positions are interpolated from the ANT table thanks to their respective Modified Julian Day values

\[
f = \frac{\text{mjd}_S - \text{mjd}_A(j)}{\text{mjd}_A(j+1) - \text{mjd}_A(j)}
\]  

where \(\text{mjd}_S\) is the spectrum MJD value, and \(\text{mjd}_A(j)\) is the MJD value of the \(j^{th}\) trace in the ANTSLOW table. \(j\) is computed thanks to a dichotomic search in the table such as

\[
\text{mjd}_A(j) \leq \text{mjd}_S < \text{mjd}_A(j+1).
\]

\(f\) being the interpolation fraction between the \(j^{th}\) and \(j+1^{th}\) trace, the positions are interpolated by

\[
\text{l}_S = \text{l}_A(j) + f \times [\text{l}_A(j+1) - \text{l}_A(j)],
\]

where \(\text{l}_S\) and \(\text{l}_A\) are the spectrum and antenna lambda offsets respectively. Same formula applies for the beta, azimuth, and elevation values.

If \(\text{mjd}_S\) is found beyond the ANTSLOW table limits, the boundary values are applied without extrapolation. However, this is not expected to happen since such spectra should be rejected since they are out of the on-track range.
3.3.3 Wobbler switching

In case of wobbler switching, the LONGOFF and LATOFF columns describe the primary dish direction on sky. Because the secondary mirror switches with additional offsets between the ON and OFF positions, MRTCAL assumes that the ON phases always have (0,0) antenna offsets. For simplicity, the OFF phases offsets are also set to (0,0) (instead of ±Δ/2) but the CLASS switching section will describe properly the throw when saving the ON-OFF result, or when saving the ON and OFF separately. Remember that the wobbler throw is usually along azimuth with a given projection, while the CLASS offsets are described in equatorial or galactic system with radio projection: their addition is not straightforward.

3.3.4 Slow and Fast traces

The IMBFITS files provide 2 antenna streams and associated tables: The slow (1 Hz) and the fast (128 Hz) traces. For each of those traces, some antenna position parameters are available, plus the associated time as MJD values.

As of today (IMBFITS version 2), the 2 traces are in the same FITS extension. Both have the same number of rows, but the fast trace has 128 values per column and per row, while the slow trace has 1 value per column and per row. However, even if they have the same number of rows, they are INDEPENDENT: For example, their MJDs are misaligned by a ∼2 seconds shift (2 rows). They should not be correlated! In the IMBFITS version 3, the 2 traces will be clearly separated in 2 FITS extensions, which will avoid any misinterpretation.

Note that the caveat exposed above explains why MRTCAL selects more dumps than MIRA in on-the-fly maps: MIRA looks at the antenna slow column TRACEFLAG to flag out MJD values in the antenna fast. Because of the ∼2 seconds shift between the 2 traces, more time dumps are discarded.

3.4 Offsets of multi-pixel receiver

In the IMBFITS files, the backend table describes how the chunks cover the bandwidth and from which receiver they come from (e.g. H or V polarization). In this table, the column PIXEL also describes from which pixel the signal comes. This is useful in the context of multi-pixel receivers. In such a case, there are several pixels which do not look at the same position on sky. In the frontend table header, the derotator mode (sky, frame, or horizontal) and angle describe the commanded inputs from the user. The derotator table, which is present in the IMBFITS files only if relevant, describes the commanded and actual derotator position through time, at typical intervals of 5 seconds. We describe below how MRTCAL makes use of these informations.

3.4.1 Handling the derotator

If the subscan has no derotator table, the current pixel (obviously the unique pixel) is left centered on the current position (reference + position offset) computed in the section [3.3.2]. On the other hand, if a derotator table is present, MRTCAL will apply a per-pixel offset.

First of all, MRTCAL tries to guess the actual derotator angle on sky. The strategy is the following:

- angle on sky will be chosen from the column sAct (“sky Actual”), as opposed to fAct (“frame Actual”) and hAct (“horizontal Actual”). This assumes sAct values are reliable even if the current derotator mode is not sky.
Figure 3.1: Asymmetrical wobbler switching illustration. For each subscan, the antenna (primary dish) points at $\pm \Delta/2$ away from the source, usually in the azimuthal direction (blue line). This is called *antenna noding*. During the subscan, the secondary mirror switches (red line) from the ON position (at offset 0) to one OFF position (at offset $\pm \Delta$). $\Delta$ is the *wobbler throw*. This pattern, which alternates between OFF1 and OFF2 at each subscan, is called *asymmetrical wobbler switching*, as opposed to the *symmetrical wobbler switching* (see details in Fig 3.2).

Figure 3.2: Symmetrical wobbler switching illustration. Same as Fig 3.1 except that the subscans are observed with symmetry around a given time. Thanks to this, and assuming that the atmospheric properties drift linearly through time, the atmospheric variations between the 2 OFF1 subscans and between the 2 OFF2 subscans average at the same level.
• if the column $s\text{Act}$ is empty (this can happen around midnight), a warning is raised and the returned value is the commanded value as found in the frontend table header. In this case, the derotator mode must be sky.

• if the column $s\text{Act}$ provides no value within the subscan range, a warning is raised.

• if the column $s\text{Act}$ has a single row, or if the chunk MJD is beyond the table limits, the unique/nearest value is used.

• if the chunk MJD lies within the column $s\text{Act}$ limits, the derotator angle is interpolated according the associated MJD column and to the chunk MJD computed earlier.

• if the chunk MJD is further than 5 seconds\footnote{5 seconds is the derotator sampling rate under normal conditions.} from the nearest point in the column $s\text{Act}$, an additional warning is raised.

• finally, if the actual angle is more than 0.5 degrees away from the commanded value (e.g. derotator has reached its rotating limit), a warning is raised.

### 3.4.2 Computing the pixel coordinates

Given the actual position of the derotator, \texttt{MRTCAL} can compute the pixel position on sky from the current telescope position. This position has been pre-computed using the radio projection (\texttt{LONGOFF} and \texttt{LATOFF} columns in the IMB-FITS), only the per-pixel offsets are to be added. \texttt{MRTCAL} checks first if it knows the offsets for the current receiver (as found in the column \texttt{RECNAME} in the frontend table). The pixels offsets are not self-described in the \texttt{IMBFITS}: they must be hardcoded in the program, with a correct correspondance between the pixel number found in the backend table and the hardcoded offsets\footnote{See the Fig.4 in the \textit{HERA user manual} for this receiver}. Once the offset of the current pixel is found, the strategy is the following.

• The current position (reference + position offset) is unprojected to absolute spherical coordinates.

• The pixel position angle on sky is computed, given the pixel position in the array and the derotator angle.

• The spherical distance from the array center is added using an inverse Haversine formula, i.e., the pixel position in absolute spherical coordinates is found by adding an arc of great-circle towards the correct direction.

• The resulting position is reprojected again as a reference + position offset in the current projection system.

The deprojection, offset in spherical coordinates, and reprojection are done by the Gildas internal engines.
3.4.3 Description of the radio projection

The unprojection-reprojection approach to add the pixel offsets is more accurate than simple offset addition to projected position, in particular far away from the equator of the coordinate system. If we look at the radio projection (its standard FITS name is Global Sinusoidal, abbreviated into GLS) of the full sky shown in Fig. 3.3, we can see that one can expect strong distortions near the pole. Note that the distortions depend on the declination of the object and NOT on the declination of the projection center. In other words, the plane of projection is not tangent to the projection center, but to the equator instead. This is directly linked to the mathematical formula of the radio projection

\[
\begin{align*}
    x &= (A - A_0) \times \cos(d), \\
    y &= d - d_0,
\end{align*}
\]

where \((A_0, d_0)\) is the projection center, \((A, d)\) the object absolute coordinates, and \((x, y)\) its offsets in the projected map. The key point in this projection is that \(x\) is a function of \(d\) and not \(d - d_0\). Hence, the deformation for wide fields of view will depend both on the distance to the projection center and the source declination! That’s why the radio projection is deprecated and IAU recommends to replace it by the Sanson-Flamsteed projection (abbreviated in SFL).

The distortions near the pole may thus have a non-negligible impact on observations. Figures 3.4 to 3.6 show the effect in different conditions. The geometry of the multi-beam array shows no visible distortions, even one degree away from the source position, when the source position is located on the equator. However, the distortions of the array geometry increase with the distance to the projection center when the source is located at high declination. Moreover, for the same source coordinates, the deformation at high source declination depends on the right ascension of the projection center (cf. Fig. 3.6), but they are independent of the projection center declination (cf. Fig. 3.5)!

\footnote{... while most of the other projections offer minimal distortions near the projection center!}
Figure 3.3: Radio projection of the full sky (right ascension from -12 to 12 hours, declination from -90 to 90 degrees), for four projection centers (marked with a blue cross) at 0 right ascension and 0, 30, 60, 90 degrees declination. The parallels (resp. the meridians) are spaced by 30 degrees (resp. 2 hours).
Figure 3.4: Effects of the radio projection on observations with HERA for source centers at 4h, 12h, 20h Right Ascension and 0, 44, 88 degrees declination (blue crosses). In this case, the projection center is aligned on the source center. The green polygon shows the shape of a 1 square degree field in sky spherical coordinates. In addition, the red parallelograms show the resulting shape of a 480” × 480” square multi-beam array as a function of the distance from the source center. Finally, the parallels (resp. meridians), shown as black dashed lines, are separated by 0.5 degree (resp. 1 degree).
Figure 3.5: Same as Fig. 3.4, except that the projection center is located 15 degrees south the source center. The deformations are independent of the declination of the projection center!
Figure 3.6: Same as Fig. 3.4, except that the projection center is located 1 hour angle west from the source center. The deformations depend on the right ascension of the projection center.
3.4.4 In practice: Effect on Polaris observations

For reference, we first remind the observing strategy advised in the HERA manual. Two main points will help us explain the behavior on Polaris.

- The array is rotated by a given angle that ensures Nyquist sampling and optimal sky coverage in a single observation (i.e., each portion of the sky is observed only by one of the nine pixels during a scan). This requires two contiguous subscans.

- However, the observing strategy usually set up at Pico Veleta samples regularly the projected sky and not the sky itself as naive users tries to obtained rectangular projected maps. Moreover the HERA receiver array is a perfect square in focal plane and on the sky, but not in the projected map!

Under observer-friendly conditions, namely relatively small maps near the projection center and maps near the equator, these rules deliver the expected result. However, if those conditions are not met (large field of view and the source is located at high declination), the sky and the projected map will both be incorrectly sampled!

Maps near Polaris (declination 87:42:04.6) where observed during the project 219-07 (PI: P.Hily-Blant). The maps extend typically from (0,0) to (-1500,2000) arcsec (projected offsets). We focus here on the scans 26 and 40 covering the projection center, and scans 98 and 115 observed at the largest distance from the projection center.

Figure 3.7 shows the scans observed near the projection center. They do not show any particular effect visible by eye. The array receiver is still square (rotated by 9.6 degrees) on the projected map, and the coverage of the 2 scans is as expected. Figure 3.8 shows the scan coverages in spherical coordinates (i.e., without projection). The spherical sky is correctly sampled. On the other hand, the scans observed far away from the projection center (−1500′′, 2000′′) are highly affected by the projection distortions. The projected receiver array is not square anymore. It is parallelogram-shaped in the projected map. As a consequence, the rows scanned in declination are shifted. Figure 3.9 shows that the vertical scanning is not correctly sampled in the projected sky. This can be understood by looking at the scan in absolute coordinates (Fig. 3.10): The scan does not follow a South-North direction suited for the 9.6 degrees derotator angle on the sky. To first order, the angle should have been different (the exact value is not computed here). On the other hand, the scan along the right ascension is correctly sampled, but the start and end points of the rows are actually shifted in projected coordinates compared to the scan 26 near the projection center. This leads to different edge effects in right ascension according to the declination of the source center.

---

5 See HERA Manual Fig.7
6 To second order, the pattern is an arc and the derotator angle could have to be modified along the scan.
Figure 3.7: Scans 26 (top) and 40 (bottom) relative coordinates (radio projection) of project 219-07 observing Polaris (declination 87:42:04.6). The observing strategy is the usual one, i.e. there were 2 OTF subscans (red: first subscan, black: second subscan), with a 9.6 degrees derotator angle, and a slight shift from one subscan to another to fill the gaps. These 2 scans cover the (0,0) reference position.
Figure 3.8: Same as Fig.3.7 (scans near the projection center) but showing absolute coordinates. In this case, the uniform mapping of the projected map results in uniform mapping of the sky.
Figure 3.9: Same as Fig. 3.7 but showing scans 98 (top) and 115 (bottom) relative coordinates (radio projection). The same observing strategy is used. These 2 scans cover the (-1500, 2000) offset position, far from the reference position. Note parallelogram shape of the receiver array in the projected map, and as a consequence the unexpected coverage of the vertical scan.
Figure 3.10: Same as Fig.3.9 (scans far from the projection center) but showing absolute coordinates. In this case, the attempt of uniform mapping of the projected map results in unexpected mapping of the sky. In particular, the 9.6 degrees derotator angle on the sky is obviously not suited here.
3.5 Guessing ON and OFF positions

Depending on the observing mode, the telescope is expected to observe 1 or 2 positions on the sky:

- Position switch: 2 positions (ON + OFF)
- Wobbler switch: 1 or 2 positions
- Frequency switch: 1 position

Those positions are not described directly in the IMB-FITS. Instead, the positions of each dump is available in the \textsc{LONGOFF} and \textsc{LATOFF} columns of the antenna slow table. By using equivalence classes of the X and Y offset pairs, it should straightforward to group the dumps by position.

For projection offset system, \textit{i.e.} when offsets are described along a sky system of coordinates (equatorial or galactic), this approach is correct: the offsets are all strictly identical for a given position and for both coordinates. All digits are the same, there are no numerical round-off errors.

![Figure 3.11: LONGOFF and LATOFF vs MJD for iram30m-fts-20150429s199-imb.fits (whole scan). Observing mode is tracked wobbler switch, using horizontalTrue offset system. During the scan, elevation changes from 82.3 to 83.6 degrees.](image)

On the other hand, for \texttt{horizontalTrue} offset system, the offsets drift. In the \texttt{horizontalTrue} offset system, \texttt{LONGOFF} are actually offsets along azimuth, corrected by
cos(elevation), and LATOFF are offsets along elevation. If we look at the example in Fig. 3.11, we can see that:

- LATOFF offsets are all null: this means that all the dumps were sharing the same elevation as the reference (source) position. The telescope is tracking the source, so that the elevation changes (this can be seen in the CELEVATIO column), but not the offset in this direction.

- LONGOFF offsets are drifting, slowly decreasing for both positions.

Because of this drift, MRTCAL uses a centi-arcsecond tolerance when the equivalence classes are computed. If we reuse the example in Fig. 3.11, this tolerance is still not enough and results in 6 different positions:

<table>
<thead>
<tr>
<th>E-SUBSCAN&gt;LIST&gt;BUILD,</th>
<th>Position #1:</th>
<th>Position #2:</th>
<th>Position #3:</th>
<th>Position #4:</th>
<th>Position #5:</th>
<th>Position #6:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan has 6 positions (expected 1 or 2):</td>
<td>-119.82903552 0.00000000</td>
<td>119.82643276 0.00000000</td>
<td>-119.81881083 0.00000000</td>
<td>119.81592996 0.00000000</td>
<td>-119.80748135 0.00000000</td>
<td>119.80419607 0.00000000</td>
</tr>
</tbody>
</table>
Chapter 4

Interpolation of calibration products

MRTCAL computes one set of calibration products _per chunk_. These products are:

- the receiver temperature,
- the calibration temperature,
- the system temperature,
- the precipitable water vapor,
- and the zenithal opacity.

Their exact values are evaluated for the frequency at the middle of the chunk. The values at other frequencies in the chunk can be evaluated using one of the two available modes:

1. in the _flat_ mode, the central values are reused over all the chunk bandwidth.

2. the second mode is _interpolation_. In this case, the chunks are reordered by ascending frequency, and the values at non-central frequencies are interpolated between the current chunk and its nearest neighbour, so that it is computed at the frequency of the desired channel. This has the advantage to avoid discontinuities along the whole bandwidth and thus to avoid calibration platforming. For the two half-chunks at the boundary of the whole bandwidth, where no neighbour chunk is available, the values are extrapolated from the boundary chunk and its neighbour on the other side.

---

1 Remember that the chunk bandwidth is ruled by \texttt{MSET\_CALIB\_BANDWIDTH}
Chapter 5
Memory index

5.1 Strategy

MRTCAL indexes IMBFITS files in the so-called index files. A raw index file named index.mrt can be produced with the command INDEX BUILD (see command help for more advanced uses). When one or more index files are then reopened for reading, the command INDEX OPEN builds a memory index named the “Input indeX” (IX). At this stage, IX gathers the summary of all entries in all opened index files. From this full set, the command MFIND can be used to make a selection of desired entries: This is the “Current indeX” (CX).

While the primary purpose of the index files is to reference IMBFITS files, their other advantage is to store other products that can be derived from the IMBFITS files. In particular, after the calibration, the commands CALIBRATE and PIPELINE will save new versions of each calibrated entry. These new versions have a modified calibration status (from none to failed, empty, or done), and optionally a calibration section added (storing the calibration results). These modified versions of the entries are themselves saved in an index file (either the original one, or a new one, depending on the user choice), AND are implicitly appended to IX. After the calibration process, it is then easy for the user to select from IX the calibrated entries (or failed, or pending, etc). The figure 5.1 shows a basic example of these steps.

Figure 5.1: Basic example showing how the IMBFITS files are first indexed, and then how the calibration command (here, PIPELINE) adds a new version of each (tentatively) calibrated IMBFITS.

5.2 Contents

When the index file is loaded, its index content is duplicated in memory for all the entries. These elements are described in the Table 5.1.
Table 5.1: Memory index elements coming directly from the index file.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Unit</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>bloc</td>
<td>records</td>
<td>Entry position in index file</td>
</tr>
<tr>
<td>word</td>
<td>words</td>
<td>Entry position its this record</td>
</tr>
<tr>
<td>version</td>
<td>—</td>
<td>Entry version</td>
</tr>
<tr>
<td>telescope</td>
<td>code</td>
<td>Telescope code</td>
</tr>
<tr>
<td>projid</td>
<td>—</td>
<td>Project id</td>
</tr>
<tr>
<td>source</td>
<td>—</td>
<td>Source name</td>
</tr>
<tr>
<td>dobs</td>
<td>gag_date</td>
<td>Observation date</td>
</tr>
<tr>
<td>ut</td>
<td>rad</td>
<td>Observation time</td>
</tr>
<tr>
<td>lst</td>
<td>sec</td>
<td>Local Sideral Time</td>
</tr>
<tr>
<td>az</td>
<td>rad</td>
<td>Azimuth</td>
</tr>
<tr>
<td>el</td>
<td>rad</td>
<td>Elevation</td>
</tr>
<tr>
<td>frontend</td>
<td>—</td>
<td>Receiver names</td>
</tr>
<tr>
<td>scan</td>
<td>—</td>
<td>Scan number</td>
</tr>
<tr>
<td>backend</td>
<td>code</td>
<td>Backend code</td>
</tr>
<tr>
<td>obstype</td>
<td>code</td>
<td>Observation type</td>
</tr>
<tr>
<td>switchmode</td>
<td>code</td>
<td>Switching mode</td>
</tr>
<tr>
<td>filstatus</td>
<td>code</td>
<td>Can read file or not?</td>
</tr>
<tr>
<td>calstatus</td>
<td>code</td>
<td>Calibration status</td>
</tr>
<tr>
<td>filename</td>
<td>—</td>
<td>IMBFITS file (no path)</td>
</tr>
<tr>
<td>itime</td>
<td>nanosec</td>
<td>Last indexing time (from 01-jan-1970)</td>
</tr>
</tbody>
</table>

However, these elements alone are not enough to deal with each IMBFITS file referenced in the index. Extra numbers are associated on-the-fly, at load time. The Table 5.2 summarizes those numbers. In details, here is the exact meaning of those numbers.

**num** The observation number is a unique number associated to each IMBFITS file. Uniqueness is ensured by a unique combination of the observing date, the scan number, and the backend. In a memory index, there can be several version of the same observation, meaning that the same IMBFITS file is indexed several times (e.g., a first time when the index is built, a second time after the calibration). This number (and optionally its version) is intended to be used as a frontend to the end-user. However, it is volatile: It can change from one session (or one INDEX OPEN) to another, depending on the index contents, and their number and order if several indexes are loaded in memory. In practice, the observation numbers are contiguous, starting from 1, and ordered by observing date, then by scan number, then by

Table 5.2: Memory-only index elements associated to each entry

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Unit</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>num</td>
<td>—</td>
<td>Observation number</td>
</tr>
<tr>
<td>mnum</td>
<td>—</td>
<td>Entry position in the Input indeX (IX)</td>
</tr>
<tr>
<td>fnum</td>
<td>—</td>
<td>Entry position in the index file</td>
</tr>
<tr>
<td>idir</td>
<td>code</td>
<td>Associated index file and IMBFITS directory</td>
</tr>
<tr>
<td>sort</td>
<td>—</td>
<td>Sorting array</td>
</tr>
</tbody>
</table>
backend code.

\textbf{mnum} The \textit{memory number} is the entry position in the input index (IX), and thus it provides a unique identifier for each entry indexed in memory. This means that the array \texttt{ix%mnum(:)} runs contiguously from 1. Since IX is the absolute reference for all sub-indexes (\textit{e.g.}, the current index CX), each entry of any index can easily be identified thanks to this back pointer (\textit{e.g.}, \texttt{cx%mnum(ient)}).

\textbf{fnum} The \textit{file number} is the entry position in its associated index file. This number can not be implicit, because several index files can be loaded in memory, and because the entries are reordered by date, scan and backend at load time.

\textbf{idir} the \textit{directory code} identifies each index file loaded in memory, so that each entry knows easily to which index file it comes from. This code also identifies the directory the \textbf{IMBFITS} can be found in (remember the index file may not be hosted in the same directory; this is controled by the user through the command INDEX /FILE).

\textbf{sort} The \textit{sorting array} is an indirection array which would order the memory index by date, by scan, and by backend. At load time, the entries are actually ordered in memory, so this array is not needed at this time. But later on, commands like \textbf{CALIBRATE} or \textbf{PIPELINE} can add new versions of the entries. These new entries are appended at the end of the memory indexes, in order to avoid breaking the cross references between indexes (in particular, the \texttt{cx%mnum(:)} back pointers to \texttt{ix}). Each application which needs to access the entries sorted by date, scan, and backend, should then use the sorting array. Typically, one has to loop on \texttt{cx%mnum(cx%sort(ient))} (with \texttt{ient} 1 to \texttt{cx%next-1}) to access the entries ordered correctly.
Chapter 6

Scan date vs observation date: Midnight issue

When a new scan is prepared\(^1\) at the telescope, the associated \texttt{IMBFITS} file name is constructed with the current date. Later in this process (up to several minutes after), the scan is actually started\(^2\), and the current date and time are used to build the \texttt{MJD-OBS} and \texttt{DATE-OBS} in the FITS Primary header. If those two steps are performed exactly before and after midnight respectively, the file name and the Primary header refer to 2 different dates, e.g.,

\begin{verbatim}
iram30m-fts-20120809s323-imb.fits:
  MJD-OBS = 56149.0038078704 / MJD at observation start
  DATE-OBS= '2012-08-10T00:05:29.000'
\end{verbatim}

For several purposes, \texttt{MRTCAL} saves the \textit{observing date and time} of each file in the index. In order to deal correctly with the above issue, it is decided

1. to save the date as found in the file name, so that there is no difference for the user between the file name and the date exposed by \texttt{MRTCAL};

2. the time value saved for this file refers to the above date.

The direct consequence of those 2 rules is that, in case of the midnight issue exposed above, the UT value will be larger than 24h. Note that the UT value associated to each file is saved for bookkeeping purpose, e.g. for analysis of the calibration though date and time. It is considered as a typical value for the whole scan. Each individual spectrum produced by \texttt{MRTCAL} uses its own date and time.

In details, the observing date saved in the index is used

- to check the entry uniqueness (date, scan and backend triplet should be unique),
- to sort the entries (by date, scan and backend),
- through the command \texttt{MFIND /DATE},
- in the output of the command \texttt{MLIST},
- in the Sic variables \texttt{MDX\%DOBS} and \texttt{MHEAD\%KEY\%DOBS}

\(^1\) scan prepared or scan loaded in the NCS nomenclature.
\(^2\) scan started or subscan 1 started in the NCS nomenclature.
and the associated UT value is used

- to sort the entries,
- in the output of the command `MLIST`,
- in the Sic variables `MDX%UT` and `MHEAD%KEY%UT`

The user should be ready to encounter UT values larger than 24h in the `MLIST` output and larger than $2\pi$ in the Sic variables.
Chapter 7

Patching IMB-FITS at read time

MRTCAL is divided in 2 main libraries:

1. the libimbfits reads the IMB-FITS files and fills a memory structure from it,

2. the libmrtcal gets the libimbfits memory structure, and use it to index the files and calibrate them.

When MRTCAL has to deal with several versions of the IMB-FITS (e.g. V1.35 for HERA and V2 for EMIR), the variations are hidden in the libimbfits so that the libmrtcal does not have to worry about those details. This means that the memory structure is unique while the disk structure differ.

Note that each patch is saved as a warning in the comment field of each element. Those warning are visible when using the command MDUMP.

7.1 Added elements when missing

The purpose of adding elements in the HDUs when they are missing is that the libmrtcal does not have to worry about this element existing or not, and and doing different things (different code) in one case or the other. The following elements are added:

- POLEX and POLEY are absent under V1.35: they are defined and default to 0,
- SYSOFF, XOFFSET, and YOFFSET columns in the Scan table can have 0, 1, or 2 rows. They are forced to 2 rows providing Nasmyth and projection offsets (default to 0 if they were absent),
- the LINENAME column in the BackEnd HDU is built from the LINENAME column in the FrontEnd HDU for IMB-FITS version lower than 2, or for 4MHz backend,
- MJD_BEG and MJD_END keywords are added as convenient duplicates of DATE-OBS and DATE-END keywords in all HDUs of each subscan. They offer MJD values instead of ISO strings,

[Note that this strategy is in use since a long time in GILDAS to ensure the backward compatibility of the various formats.]
• the column TSTAMPED is added to BackendDATA tables in IMB-FITS version lower or equal to 1.35. Its defaults to 1. Remember that this value is sensitive (it introduces a shift on the time stamps) and is actually backend-dependent.

• an IFRONT column is added as a convenient backpointer from the BackEnd table to the FrontEnd table, i.e. each chunk knows easily to which receiver it is associated.

7.2 Modified elements

• IFCENTER, FRQOFF1, and FRQOFF2 columns from the FrontEnd table are reversed for each row where IFFLIPS is true. This factorizes the sign issues when dealing with IF2 instead of IF1,

• The column BAND is read under the name PART for IMB-FITS version lower than 2. Same for REC1 and REC2 read under the names RECEIVER and BAND respectively,

• REFCHAN is patched to correct values (instead of 1) when dealing with continuum backend (1 channel per chunk),

• all the BackEnd columns, namely PART, REFCHAN, CHANS, DROPPED, USED, PIXEL, RECEIVER, BAND, POLAR, REFFREQ, SPACING, and LINENAME are sliced into memory to more values than from the file in order to factorize at once the MSET CALIBRATION BANDWIDTH tuning. NAXIS2 is updated accordingly. See section 1.1 for details,

• the columns MJD, INTEGTIM and ISWITCH are compressed if MSET CALIBRATION BAD is NO (this is the default). In this case all rows where ISWITCH is 0 are removed. Two new columns FOREPOIN and BACKPOIN are added as cross pointers between the original and compressed columns. See section 2.1 for details.

• the DATE-OBS and DATE-END stamps of each subscan are patched if MSET CALIBRATION MJDINTER is YES. See HELP for details.
IMB-FITS version 2.13

For single pixel receiver, the IMB-FITS data format provides the following FITS extensions:

0. **Primary**: general description of the observation

1. **IMBF-scan**: scan description (source, antenna offsets, etc)

2. **IMBF-frontend**: frontend description (which receiver was connected, etc)

3. **IMBF-backend**: backend configuration i.e. how the chunks translate to sky frequency.

Then, the 3 following extensions are repeated for each subscan (example with FTS backend):

4. **IMBF-backendFTS**: provides the data dumps through time,

5. **IMBF-antenna**: provides the antenna position through time at slow and fast rates.

6. **IMBF-subreflector**: not used by MRTCAL.

Each of these extensions are detailed below from an example scan, showing the output of the command **MDUMP**.

### 8.1 Primary

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMPLE (L)</td>
<td>T</td>
<td>file does conform to FITS standard</td>
</tr>
<tr>
<td>BITPIX (I4)</td>
<td>32</td>
<td>number of bits per data pixel</td>
</tr>
<tr>
<td>NAXIS (I4)</td>
<td>0</td>
<td>number of data axes</td>
</tr>
<tr>
<td>EXTEND (L)</td>
<td>T</td>
<td>FITS dataset may contain extensions</td>
</tr>
<tr>
<td>TELESCOP (C)</td>
<td>IRAM 30m</td>
<td>Telescope Name</td>
</tr>
<tr>
<td>ORIGIN (C)</td>
<td>IRAM</td>
<td>Organisation or Institution</td>
</tr>
<tr>
<td>CREATOR (C)</td>
<td>Python IRAM MBFITS Wri</td>
<td>Software</td>
</tr>
<tr>
<td>INSTRUME (C)</td>
<td>fts</td>
<td>Backend</td>
</tr>
<tr>
<td>OBJECT (C)</td>
<td>XXXXXXXX</td>
<td>Source Name</td>
</tr>
<tr>
<td>LONGOBJ (R8)</td>
<td>00.0000000000000000</td>
<td>[deg] Source longitude in basis frame</td>
</tr>
<tr>
<td>LATOBJ (R8)</td>
<td>00.0000000000000000</td>
<td>[deg] Source latitude in basis frame</td>
</tr>
<tr>
<td>TIMESYS (C)</td>
<td>UTC</td>
<td>time system (TT,TAI,UTC ...)</td>
</tr>
<tr>
<td>MJD-OBS (R8)</td>
<td>57841.4547916667</td>
<td>MJD at observation start</td>
</tr>
</tbody>
</table>

---

1. **MDUMP** slightly modifies the header elements at read time for convenience. Some remarks are added accordingly in the output.
8.2 IMBF-scan

XTENSION (C) = BINTABLE / binary table extension
BITPIX (I4) = 8 / 8-bit bytes
NAXIS (I4) = 2 / 2-dimensional binary table
NAXIS1 (I4) = 23 / width of table in bytes
NAXIS2 (I4) = 2 / number of rows in table
PCOUNT (I4) = 0 / size of special data area
GCOUNT (I4) = 1 / one data group (required keyword)
TFIELDS (I4) = 3 / number of fields in each row
EXTNAME (C) = IMBF-scan / name of this binary table extension
TELESCOP (C) = IRAM 30m / Telescope Name
TELSIZE (R8) = 30.00000000000000 / [m] Telescope Diameter
SITELONG (R8) = -3.39875641986850 / [deg] observatory longitude (EAST)
SITELAT (R8) = 37.0684132670517 / [deg] observatory latitude (NORTH)
SITEELEV (R8) = 2851.50000000000 / [m] observatory elevation
PROJID (C) = 102-16 / Project ID
OBSID (C) = XX / Observer initials
OPERATOR (C) = Pako / Operator initials
SCANNUM (I4) = 138 / Scan number
DATE-OBS (C) = 2017-03-29T10:54:54.00 / scan start in TIMESYS system
DATE (C) = 2017-03-29T10:55:18 / Date of FITS file creation
MJD (R8) = 57841.4547916667 / [day] Scan date/time (Modified Julian Date)
LST (R8) = 83367.1998821900 / [s] Local apparent sidereal time (scan start)
N_OBS (I4) = 3 / Number of observations in this scan
EXPTIME (I4) = 15 / Total netto integration time [s]
TIMESYS (C) = UTC / time system (TT, TAI, UTC ...)
LONPOLE (R8) = 0.00000000000000 / [deg] Native longitude of celestial pole: range
LATPOLE (R8) = 0.00000000000000 / [deg] Basis latitude of native pole
LONGOBJ (R8) = 0.00000000000000 / [deg] Source longitude in basis frame
LATOBJ (R8) = 0.00000000000000 / [deg] Source latitude in basis frame
SWITCHMOD (C) = totalPower / Switch mode
NOSWITCH (I4) = 1 / no. of switch phases in a switch cycle
PHASETIM (R8) = 0.500000000000000 / [s] Integration time per phase
PHOBTHROW (R8) = 0.000000000000000 / [deg] wobbler throw
PHOBDIR (I4) = 0 / wobbler throw direction
PHOBCYCLE (R8) = 0.000000000000000 / [s] wobbler period
PHOBMODE (C) = / wobbler mode (SQUARE/TRIANGULAR))
NFEBE (I4) = 8 / \nfebe\ number of FEBEs
PRESSURE (R8) = 728.500000000000 / [hPa] Atmospheric pressure (hPa)
TAMBIENT (R8) = -3.00000000000000 / [deg C] Outside temperature (C)
HUMIDITY (R8) = 36.6000000000000 / [%] Relative Humidity (%)
WINDDIR (R8) = 187.300000000000 / [deg] Wind direction (deg)
WINDEVL (R8) = 1.500000000000000 / [m/s] Wind velocity (m/s)
WINDEVL (R8) = 2.500000000000000 / [m/s] Wind max. velocity (m/s)
DATE-WEA (C) = 2017-03-29T10:55:15.58 / Time of weather station par.
REFRACTI (R8) = 0.00000000000000 / [none] Refraction Correction, as a function of
THOT (R8) = 295.150000000000 / [K] Hot Load Temperature in K
DATE-HOT (C) = 2017-03-29T10:54:44.75 / Time of Hot Load Temp. meas.
TIPTAUZ (R8) = 0.136721000000000 / \] Tau meter zenith opacity
TIPTAUC (R8) = 0.999226000000000 / \] Tau meter goodness of fit
DATE-TIP (C) = 2017-03-29T10:54:50.85 / Time of Tau from Taumeter
SYSOFF (C) = Nasmyth projection / WARNING! Added a dummy 'projection' value
XOFFSET (R4) = -1.9150140E-04 0.000000 / WARNING! Added a dummy 'projection' value
YOFFSET (R4) = 2.6664753E-05 0.000000 / WARNING! Added a dummy 'projection' value

8.3 IMBF-frontend

XTENSION (C) = BINTABLE / binary table extension
BITPIX (I4) = 8 / 8-bit bytes
NAXIS (I4) = 2 / 2-dimensional binary table
NAXIS1 (I4) = 139 / width of table in bytes
NAXIS2 (I4) = 1 / number of rows in table
PCOUNT (I4) = 0 / size of special data area
GCOUNT (I4) = 1 / one data group (required keyword)
TFIELDS (I4) = 22 / number of fields in each row
EXTNAME (C) = IMBF-frontend / name of this binary table extension
SCANNUM (I4) = 138 / Scan number
DATE-OBS (C) = / observing date (Y2K format with time) in TIMESY
DEWANGMOD (C) = / Dewar tracking system
DEWANG (R8) = 0.000000000000000 / [Deg] Dewar angle
FEBEBAND (I4) = 1 / \nbd\ number of basebands for this febe
FEBEFEE (I4) = 8 / \nfd\ total number of feeds
NUSEFEED (I4) = 8 / \nch\ Number of feeds in use.
VELOSYS (R8) = 6.700000000000000 / [km/s] Source Radial Velocity in Reference Fram
SPECSYS (C) = LSR / Reference Frame
VELOCONV (C) = optical / Convention for doppler correction
EMIRBEAM (C) = right / EMIR beam used
RECMAME (C) = E230 / Receiver Name
LINENAME (C) = L220510 / Line Name

39
8.4 IMBF-backend

XTENSION (C) = BINTABLE / binary table extension
BITPIX (I4) = 8 / 8-bit bytes
NAXIS (I4) = 2 / 2-dimensional binary table
NAXIS1 (I4) = 81 / width of table in bytes
NAXIS2 (I4) = 24 / number of rows in table
PCOUNT (I4) = 0 / size of special data area
GCOUNT (I4) = 1 / one data group (required keyword)
TFIELDS (I4) = 12 / number of fields in each row
EXTNAME (C) = IMBF-backend / name of this binary table extension
SCANNUM (I4) = 138 / Scan number
DATE-OBS (C) = 2017-03-29T10:54:54.00 / observing date (Y2K format with time) in T
FEEBAND (I4) = 1 / \nbd\ number of basebands for this febe
FEBEFEED (I4) = 8 / \nfd\ total number of channels
NUSEFEED (I4) = 8 / \nch\ Number of channels in use.
PART (I4) = 2 ... 6 / Identification
REFCHAN (I4) = 1 ... 376833 / Reference Channel
CHANS (I4) = 16384 ... 16384 / Number of Channels
DROPPED (I4) = 1843 ... 1843 / Channels dropped
USED (I4) = 12494 ... 12494 / Channels used
RECEIVER (C) = E2HUI ... E2VLO / Receiver connected
IFRONT (I4) = 1 ... 1 / Backpointer to frontend table (memory only)
BAND (C) = E2HUI ... E2VLO / Sub-band specification
POLAR (C) = NONE ... NONE / Polarization
PIXEL (I4) = 1 ... 1 / Pixel No. (arrays)
REFFREQ (R4) = 7200.000 ... -8480.000 / [MHz] IF reference frequency
SPACING (R4) = -4.8828125E-02 ... -4.8828125E-02 / [MHz] Spacing
LINENAME (C) = L220510 ... L220510 / Additional Line Name
8.5 IMBF-backendFTS

The DATA column is omitted in this output:

XTENSION (C) = BINTABLE / binary table extension
BITPIX (I4) = 8 / 8-bit bytes
NAXIS (I4) = 2 / 2-dimensional binary table
NAXIS1 (I4) = 1572888 / width of table in bytes
NAXIS2 (I4) = 11 / number of rows in table
PCOUNT (I4) = 0 / size of special data area
GCOUNT (I4) = 1 / one data group (required keyword)
TIELDS (I4) = 5 / number of fields in each row
EXTNAME (C) = IMBF-backendFTS / name of this binary table extension
SCANNUM (I4) = 138 / Scan number
OBSNUM (I4) = 1 / Observation number
BASEBAND (C) = / [-->] Baseband number
DATE-OBS (C) = 2017-03-29T10:54:54.41 / observation start in TIMESYS system
MJD_BEG (R8) = 57841.4547964699 / Duplicate of DATE-OBS (memory only)
DATE-END (C) = 2017-03-29T10:54:58.42 / observation end in TIMESYS system
MJD_END (R8) = 57841.4548428241 / Duplicate of DATE-END (memory only)
CHANNELS (I4) = 393216 / \nch\ Number of channels for this baseband
NPHASES (I4) = 1 / no. of switch phases in a switch cycle
PHASEONE (C) = ON / First phase is ON or OFF source
TSTAMPED (R8) = 1.00000000000000 / Where the time stamps apply
MJD (R8) = 57841.4547911713 ... 57841.4548490417 / [day] MJD at integration start
INTEGTIM (R8) = 0.498954000000000 ... 0.498954000000000 / [s] Integration time
ISWITCH (I4) = 1 ... 1 / Integration type
FOREPOIN (I4) = 1 ... 11 / Forward pointer to compressed columns
BACKPOIN (I4) = 1 ... 11 / Backward pointer to uncompressed columns

8.6 IMBF-antenna

MRTCAL splits the antenna extension in the antenna slow and antenna fast descriptions.

Antenna slow:

XTENSION (C) = BINTABLE / binary table extension
BITPIX (I4) = 8 / 8-bit bytes
NAXIS (I4) = 2 / 2-dimensional binary table
NAXIS1 (I4) = 6228 / width of table in bytes
NAXIS2 (I4) = 8 / number of rows in table
PCOUNT (I4) = 0 / size of special data area
GCOUNT (I4) = 1 / one data group (required keyword)
TIELDS (I4) = 19 / number of fields in each row
EXTNAME (C) = IMBF-antenna / name of this binary table extension
SCANNUM (I4) = 138 / Scan number
OBSNUM (I4) = 1 / Observation number
DATE-OBS (C) = 2017-03-29T10:54:54.41 / observation start in TIMESYS system
MJD_BEG (R8) = 57841.4547964699 / Duplicate of DATE-OBS (memory only)
DATE-END (C) = 2017-03-29T10:54:58.42 / observation end in TIMESYS system
MJD_END (R8) = 57841.4548428241 / Duplicate of DATE-END (memory only)
OBSTYPE (C) = calibrate / Observation type
SUBTYPE (C) = calSky / Subscan type
SUBSTIME (R8) = 5.00000000000000 / [s] Subscan time
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEMOFF</td>
<td>projection</td>
</tr>
<tr>
<td>SUBSXOFF</td>
<td>-2.9088820000000000E-03 [rad] Subscan x offset</td>
</tr>
<tr>
<td>SUBSYOFF</td>
<td>0.000000000000000000 [rad] Subscan y offset</td>
</tr>
<tr>
<td>SETYPE01</td>
<td>Segment type</td>
</tr>
<tr>
<td>SETIME01</td>
<td>0.0000000000000000 [s] Segment time</td>
</tr>
<tr>
<td>SEXOFF01</td>
<td>0.000000000000000000 [rad] Segment x offset</td>
</tr>
<tr>
<td>SEYOFF01</td>
<td>0.000000000000000000 [rad] Segment y offset</td>
</tr>
<tr>
<td>SEXSTA01</td>
<td>0.000000000000000000 [rad] Segment x start</td>
</tr>
<tr>
<td>SEYSTA01</td>
<td>0.000000000000000000 [rad] Segment y start</td>
</tr>
<tr>
<td>SEXEND01</td>
<td>0.000000000000000000 [rad] Segment x end</td>
</tr>
<tr>
<td>SEYEND01</td>
<td>0.000000000000000000 [rad] Segment y end</td>
</tr>
<tr>
<td>SESPE01</td>
<td>0.000000000000000000 [rad/s] Speed at start of segment</td>
</tr>
<tr>
<td>SESPEE01</td>
<td>0.000000000000000000 [rad/s] Speed at end of segment</td>
</tr>
<tr>
<td>DOPPLERC</td>
<td>0.999881490337000000 / Doppler correction factor</td>
</tr>
<tr>
<td>OBSVELRF</td>
<td>28.83251400710000 / [km/s] Observer velocity in rest frame</td>
</tr>
<tr>
<td>MJD</td>
<td>57841.4547916667 ... 57841.4548726852 / [Julian day] MJD at integration start</td>
</tr>
<tr>
<td>LST</td>
<td>83367.2657212536 ... 83374.28861940 / [s] Local apparent sidereal time (integration time)</td>
</tr>
<tr>
<td>LONGOFF</td>
<td>-2.908881986513734E-03 ... -2.908881986513734E-03 / [rad] long. offset from source in user frame</td>
</tr>
<tr>
<td>LATOFF</td>
<td>0.0000000000000000 ... 0.000000000000000000 / [rad] lat. offset from source in user frame</td>
</tr>
<tr>
<td>CAZIMUTH</td>
<td>1.31577241984522 ... 1.31600523032287 / [rad] Commanded Azimuth</td>
</tr>
<tr>
<td>CELEVATI</td>
<td>0.630288975389516 ... 0.630683181202881 / [rad] Commanded Elevation</td>
</tr>
<tr>
<td>TRACEFLA</td>
<td>0 ... 0 / Trace Flag</td>
</tr>
</tbody>
</table>

Antenna fast:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>XTENSION</td>
<td>BINTABLE</td>
</tr>
<tr>
<td>BITPIX</td>
<td>8 / 8-bit bytes</td>
</tr>
<tr>
<td>NAXIS</td>
<td>2 / 2-dimensional binary table</td>
</tr>
<tr>
<td>NAXIS1</td>
<td>6228 / width of table in bytes</td>
</tr>
<tr>
<td>NAXIS2</td>
<td>8 / number of rows in table</td>
</tr>
<tr>
<td>PCOUNT</td>
<td>0 / size of special data area</td>
</tr>
<tr>
<td>GCOUNT</td>
<td>1 / one data group (required keyword)</td>
</tr>
<tr>
<td>TFIELDS</td>
<td>19 / number of fields in each row</td>
</tr>
<tr>
<td>EXTNAME</td>
<td>IMBF-antenna</td>
</tr>
<tr>
<td>DATE-OBS</td>
<td>2017-03-29T10:54:41.41 / observation start in TIMESYS system</td>
</tr>
<tr>
<td>MJD_BEG</td>
<td>57841.4547964699 / Duplicate of DATE-OBS (memory only)</td>
</tr>
<tr>
<td>DATE-END</td>
<td>2017-03-29T10:54:58.42 / observation end in TIMESYS system</td>
</tr>
<tr>
<td>MJD_END</td>
<td>57841.4548428241 / Duplicate of DATE-END (memory only)</td>
</tr>
<tr>
<td>TRACERAT</td>
<td>128 / No. of fast traces per slow trace</td>
</tr>
<tr>
<td>MJDFAST</td>
<td>57841.45475694444 ... 57841.4548494466 / [Julian day] MJD at integration start</td>
</tr>
<tr>
<td>AZIMUTH</td>
<td>1.315504000027680 ... 1.31576626838095 / [rad] Encoder Azimuth</td>
</tr>
<tr>
<td>ELEVATIO</td>
<td>0.6317572341984522 ... 0.630683181202881 / [rad] Encoder Elevation</td>
</tr>
</tbody>
</table>