Current SMA data file format

Taco, Attila
tags: data:analysis
last updated: 2021-01-21 11:49:27

Contents:

When taking interferometer data, the SMA records data in a set of files all contained in a single directory. This document describes the contents of those files. This data file format, which we refer to as "MIR format", was initially developed at OVRO. That's why you'll see that some of the data fields are unused, and a few variables have slightly odd name choices for the use we are making of them. This is the result of us shoe-horning our data into a format that was defined at a different observatory.

It is important to remember these files were written by a little endian processor! You will not need to worry about this if you process the data on an x86 architecture machine, but if you read the data on a big endian machine, like a PowerPC or SPARC, the integers, floats and doubles will need to be byte-swapped. Also note that the byte order used is not network byte order.

The files are listed here alphabetically, by file name.

antennas

This file contains the X, Y and Z coordinates of each antenna, in meters. The coordinate system is defined, with these letters, in Thompson, Moran and Swenson's text "Interferometry and Synthesis in Radio Astronomy" (Chapter 4, page 87 in the 2nd Edition).

This file is written to once, immediately after the data directory is opened. If the antenna coordinates are updated during a track, those new coordinates will **NOT** appear in this file.

autoCorrelations

This file contains the autocorrelation spectra from the SWARM correlator. The format is

```
typedef struct __attribute__((packed)) autoCorrDef {
  int antenna;
  int nChunks;          /* Number of chunks from SWARM-style correlators  */
  int scan;             /* Should agree with cross-corr scan number       */
  double dhrs;          /* Should agree with integration header dhrs      */
  float amp[N_SWARM_CHUNKS][2][16384]; /* [chunk][pol][channel]           */
} autoCorrDef;
```

Note. N_SWARM_CHUNKS is defined in smaglobal.h, currently with a value of 8.

bl_read

This file contains the baseline information. There is one fixed-length record written per receiver per sideband per baseline per scan.

The order of the loops writing out these records is:

```
sideband
   receiver
      polarization
         baseline
```

Each record contains:

```
typedef struct __attribute__((packed)) blhDef {
  int   blhid     ; /*  proj. baseline id #  This is a unique identifier for this record        */
  int   inhid     ; /*  integration id #                                                         */
  short isb       ; /*  sideband int code                                                        */
  short ipol      ; /*  polarization int code                                                    */
                    /*  0 = Unknown                                                              */
                    /*  1 = RR                                                                   */
                    /*  2 = RL                                                                   */
                    /*  3 = LR                                                                   */
                    /*  4 = LL                                                                   */
  short ant1rx    ; /*  0 = RxA (230/345)                                                        */
                    /*  1 = RxB (240/400)                                                        */
  short ant2rx    ; /*  same as ant1rx (above)                                                   */
  short pointing  ; /*  pointing data int code                                                   */
                    /*  = 1 if pointing is active (i.e., ants may have offset)                   */
```

```
                    /*  = 0 if antenna is pointed normally                                    */
                    /*  Used to indicate off source ipoint scans.   This flag is not used for  */
                    /* mosaicing offsets, etc.                                                */
  short irec      ; /*  receiver int code                                                     */
                    /*  0 = 230                                                               */
                    /*  1 = 345                                                               */
                    /*  2 = 400                                                               */
                    /*  3 = 240                                                               */
                    /* -1 = Rx unknown                                                        */
  float u         ; /*  u coord. for bsl (m) -- was (klambda) before v.3                      */
  float v         ; /*  v coord. for bsl (m) -- was (klambda) before v.3                      */
  float w         ; /*  w coord. for bsl (m) -- was (klambda) before v.3                      */
  float prbl      ; /*  projected baseline (m) -- was (klambda) before v.3                    */
  float coh       ; /*  coherence estimate - This is the ratio of the vector average (over    */
                    /*  spectral channels) to the scalar average, over all chunks (weighted by */
                    /*  channel bandwidth to give equal weight to every Hz interval).         */
  double avedhrs  ; /*  This is the midpoint time for the scan, in hours.                      */
  float ampave    ; /*  ave continuum amp (vector average of all channels)                    */
  float phaave    ; /*  ave continuum phase                                                   */
  int   blsid     ; /*  physical baseline id #  This is an integer label for this baseline    */
  short iant1     ; /*  antenna number of first antenna in baseline                           */
  short iant2     ; /*  antenna number of second antenna in baseline                          */
  int ant1TsysOff ; /*  Byte offset to start of Tsys information for first antenna of this    */
                    /*  baseline and this scan.  This is a byte offset to the data in the     */
                    /*  tsys_read file, for the data corresponding to the first antenna on this */
                    /*  baseline, for one particular scan (identified by inhid).              */
  int ant2TsysOff ; /*  Byte offset for Tsys data for the second antenna in the baseline      */
  short iblcd     ; /*  baseline int code                                                     */
  float ble       ; /*  bsl east vector (klambda)                                             */
  float bln       ; /*  bsl north vector klambda)                                             */
  float blu       ; /*  bsl up vector   klambda)                                              */
  int spareint1   ; /*  Spare 32-bit integer space for future use                             */
  int spareint2   ; /*                                                                        */
  int spareint3   ; /*                                                                        */
  int spareint4   ; /*                                                                        */
  int spareint5   ; /*                                                                        */
  int spareint6   ; /*                                                                        */
  double fave     ; /*  [v.3] (reserved for use by mir) center freq (GHz) for ampave/phaave.  */
  double bwave    ; /*  [v.3] (reserved for use by mir) bandwidth (MHz) for ampave/phaave.    */
  double wtave    ; /*  [v.3] (reserved for use by mir) weight for Real/Imag of ampave/phaave. */
  double sparedbl4; /*                                                                        */
  double sparedbl5; /*                                                                        */
  double sparedbl6; /*                                                                        */
} blhDef;
/* the size of blhDef is 158 bytes */
```

codes_read

The various header blocks that comprise the header information for each scan generally do not contain string values for such things as the name of the source being observed. Instead, this file contains all such strings, along with integer index values. The header blocks then contain the integer index values, rather than the strings themselves. As soon as the data files are created, certain strings, such ans "l" and "u" used to name sidebands, are written to this file. After that, new strings are added on an as-needed basis, for example, when a new source is observed.

The file consists of fixed length record with the format

```
typedef struct __attribute__((packed)) codehDef {

  char v_name[12];/* label                          */
  short  icode  ; /* index for a code word          */
  char code[26] ; /* the code word                  */
  short ncode   ; /* optional integer code value    */
} codehDef;
/* the size of codeh is 42 bytes */
```

The following codes are always written to the file, regardless of the array's configuration:

v_name	icode	code (string)	ncode	Comments
"tq"	0	"v01"	1	OVRO intended to use this to specify the correlator configuration. We do not use this at all.
"vctype"	0	"vlsr"	1	Velocity definition We use this value, always.
	1	"cz"	1	But these other codes
	2	"vhel"	1	exist
	3	"pla"	1	and are written.
"sb"	0	"l"	1	Sideband designator - LSB
	1	"u"	1	USB
"aq"	0	" "	1	These are amplitude qualifiers
	1	"1"	1	and we don't use them at all
	2	"2"	1	
"bq"	0	" "	1	Baseline quaifier - we don't
	1	"b"	1	use this at all.
"cq"	0	" "	1	Coherence qualifier - we don't
	1	"c"	1	use this at all.
"oq"	0	" "	1	Offset qualifier - we don't use
	1	"o"	1	this at all.
"rec"	0	"230"	1	Receiver code - 230 insert
	1	"345"	1	345 Rx
	2	"400"	1	400 Rx

	3	"240"	1	240 Rx
"ifc"	0	"1"	1	IF channel - we don't use this
	1	"2"	1	
"tel1"	0	"0"	1	First antenna on a baseline
	...	...	...	
	9	"9"	1	Antenna 9 = JCMT
	10	"10"	1	Antenna 10 = CSO
"tel2"	0	"0"	1	Second antenna on a baseline
	...	...	...	
	9	"9"	1	Antenna 9 = JCMT
	10	"10"	1	Antenna 10 = CSO
"gq"	0	" "	1	Gain qualifier
	1	"g"	1	
"pq"	0	" "	1	Passband qualifier
	1	"p"	1	
"band"	0	"c1"	1	Band qualifier - only this one is always written. Bands Sxx are written later, if needed.
"pstate"	0	"0"	1	OVRO may have intended to use this for some kind of polarization information. We don't use it.
"taper"	0	"u"	1	Uniform taper? We don't use this.
	1	"h"	1	Or this either.
"trans"	0	"unspecified"	1	We don't use this.
"pos"	0	"unspecified"	1	We don't use this.
"offtype"	0	"ra-dec"	1	Offset type, we don't use this.
	1	"az-el"	1	

The following codes are written when observing in dual Rx polarization mode:

"pol"	0	"Unknown"	1	
	1	"RR"	1	
	2	"RL"	1	
	3	"LR"	1	
	4	"LL"	1	
	5	"LH"	1	
	6	"LV"	1	
	7	"RH"	1	
	8	"RV"	1	
	9	"HR"	1	
	10	"HL"	1	
	11	"HH"	1	
	12	"HV"	1	
	13	"VR"	1	
	14	"VL"	1	
	15	"VH"	1	
	16	"VV"	1	

The following codes are written when we are not in dual Rx polarization mode:

"pol"	0	"hh"	1	
	1	"vv"	1	
	2	"hv"	1	
	3	"vh"	1	

Additional codes introduced in **version 2**:

"filever"	0	"4"	1	(NEW) Introduced in 2019 Sep (OP-556) to allow to allow identifying format changes. At introduction the current value is "2", which will be incremented each time a format change is introduced. For older files that do not contain this code, a value of "1" may be assumed.

Additional codes introduced in **version 4**:

"ddsmode"	0	"ER+dEO"	1	(NEW) Introduced in 2021 Jan (OP-1228) to indicated what corrections terms to LO DDS is producing (correcting for). '+' may be used to list terms applied together. The following terms are currently defined: "ER" -- Earth rotation "dEO" -- diff. Earth orbit (w.r.t. transit) The default mode (0 -> "ER+dEO") is the traditional sidereal operating mode of the SMA, in which frequencies and velocities are effectively referenced to a frozen geocentric system at the time of transit. The traditional mode for Solar System Bodies (SSBs) is 1 ("off") in which no DDS corrections are applied and frequencies and velocities are topocentric. Modes 2 and higher are reserved for possible future use, and may define additional correction terms.
	1	"off"	1	
	2	"ER"	1	

Daily codes

The following code is written for each UT day. (Most files will only have one of these, because the UT day rolls over at 2:00 PM HST, and we are not usually observing at that time of the day.)

"ref_time"	0..n	[Mon dd yyyy]	0	Reference date, e.g. "Sep 13 2019"

Baseline codes

Each baseline has a record written. The icode value counts sequentially through the baselines. The string holds "A-B" where A and B are antenna numbers.

"blcd"	1..28	0	Baseline pair, where A and B are antenna numbers (e.g. "1-2").

Band Codes

Every spectral chunk has a "band" code written. icode is 0 for the continuum band (always written) and icode counts the existing chunks sequentially after that. If no chunks are discarded, icode is equal to the chunk number. The string format is "sxx" where xx is the chunk number, always written as a two digit integer, with a leading zero if needed.

"band"	0..n	[id]	0..n	

Source Codes

Each source has a record written with icode starting from 1 to the number (n) of distinct sources observed.

"source"	1..n	[name]	0	The icode value counts from 1 to the total number of sources. The string part of the record contains the ASCII text name of the source, truncated if necessary after 25 characters.
"stype"	1..n	[type]	0..1	(NEW) [v.3] Source type.
		"sidereal"	0	Sidereal source
		"ephemeris"	1	Solar System source
"svtype"	1..n	[type]	0..5	(NEW) [v.3] Source velocity type.
		"unknown"	0	Unknown / undefined
		"lsr"	1	Local Sidereal Reference (LSR) frame.
		"topo"	2	Topocentric (e.g. for solar system objects)
		"geo"	3	Geocentric (not used currently)
		"helio"	4	Heliocentric (not used currently)
		"bary"	5	Barycentric (not used currently)

Project Codes

Each project has a record written with icode starting from 1 to the number (n) of distinct projects observed.

"project"	1..n	[ID]	0	The icode value counts from 1 to the total number of projects in the file. The string part of the record contains the ASCII project description, truncated if necessary after 25 characters.

Scan Codes

Each scan has a number of codes. icode for this type of record equals the scan number

"ut"	[scan-no]	[timestamp]	0	Scan date "Mon dd yyyy hh:mm:ss.SSS{AM/PM}", E.g. "Sep 13 2019 11:29:33.429AM"
"ra"	[scan-no]	[hh:mm:ss.ss]	0	Right ascension (J2000).
"dec"	[scan-no]	[{+/-}dd:mm:ss.s]	0	Declination (J2000).
"vrad"	[scan-no]	[(m/s)]	0	Radial velocity of source (m/s).

eng_read

This file contains fixed-length records. One record is written for each antenna (not including the CSO and JCMT) after each scan completes. The format is

```
typedef struct __attribute__((packed)) antEngDef {
  int antennaNumber;
  int padNumber;
  int antennaStatus;  /*  Antenna is ON or OFF LINE                                   */
  int trackStatus;    /*  Track is running or not                                     */
  int commStatus;     /*  Data for this integration is valid or not                   */
  int inhid    ;      /*  integration id #                                            */
  int ints     ;      /*  integration #                                               */
  double dhrs  ;      /*  hrs from ref_time                                           */
  double ha    ;      /*  hour angle                                                  */
  double lst   ;      /*  lst                                                         */
  double pmdaz ;      /*  pointing model correction                                   */
```

```
  double pmdel ;
  double tiltx ;
  double tilty ;
  double actual_az ;
  double actual_el ;
  double azoff ;
  double eloff ;
  double az_tracking_error ;
  double el_tracking_error ;
  double refraction ;
  double chopper_x ;
  double chopper_y ;
  double chopper_z ;
  double chopper_angle ;
  double tsys ;
  double tsys_rx2 ;
  double ambient_load_temperature ;

} antEngDef;
/* The size of antEngDef is 196 bytes */
```

in_read

This file contains scan header information. It consists of fixed-length records. There is one record written for each scan. Note that for us, a scan and an inegration are the same thing. The format is

```
typedef struct __attribute__((packed)) inhDef {
  int   traid     ; /*  track id # Set to the Project ID.   This is the realtime system's         */
                    /*  project ID, created by the "project" command, and has nothing to do with  */
                    /*  the proposal ID                                                           */
  int   inhid     ; /*  integration id #                                                          */
  int   ints      ; /*  integration # (scan number)                                               */
                    /*  In reality, same as inhid                                                 */
  float az        ; /*  azimuth  (degrees)                                                        */
  float el        ; /*  elevation (degrees)                                                       */
  float ha        ; /*  hour angle (hours)                                                        */
  short iut       ; /*  Scan number                                                               */
  short iref_time ; /*  ref_time int code (points to a codes_read entry)                          */
  double dhrs     ; /*  average time at scan midpoint in hours                                    */
  float vc        ; /*  Radial vel. - catalog vel. in km/sec                                      */
  double sx       ; /*  x vec. for bsl. (unit vector x component towards source)                  */
  double sy       ; /*  y vec. for bsl.                                                           */
  double sz       ; /*  z vec. for bsl.                                                           */
  float rinteg    ; /*  actual int time (legth of scan in seconds)                                */
  int   proid     ; /*  project id #                                                              */
  int   souid     ; /*  source id #                                                               */
  short isource   ; /*  source int code (references a codes_read entry)                           */
  short ivrad     ; /*  Index number for the radial velocity entry in codes_read                  */
  float offx      ; /*  offset in x (for mosaics) used for RA offset  (arcsec)                    */
  float offy      ; /*  offset in y  (for mosaics) used for Dec offset (arcsec)                   */
  short ira       ; /*  ra int code                                                               */
  short idec      ; /*  dec int code                                                              */
  double rar      ; /*  Catalog R.A. (radians)                                                    */
  double decr     ; /*  Catalog declination (radians)                                             */
  float epoch     ; /*  epoch for coordinates, current epoch for solar-sys objs, otherwise 2000.0 */
  float size      ; /*  source size (arcsec) Only nonzero for planets                             */
  float vrra      ; /*  [v.3] R.A. (radians) of velocity reference                                */
  float vrdec     ; /*  [v.3] Dec (radians) of velocity reference                                 */
  float lst       ; /*  [v.3] Local Sidereal Time (hours)                                         */
  short iproject  ; /*  [v.3] icode of 'projectid' in codes_read                                  */
  short tile      ; /*  [v.3] Mosaic tile number (0 for center position or non-mosaic)            */
  char obsmode    ; /*  [v.3] Observation type (0:standard, 1:ipoint, ...)                        */
  char obsflag    ; /*  [v.3] Bitwise add-on feature flags (bit0: chopping                        */
  short spareshort; /*  -- 16-bit space for future use ---                                        */
  int spareint6   ; /*  -- 32-bit space for future use ---                                        */
  double yIGFreq1 ; /*  YIG frequencies (Hz) for offline despiking etc.                           */
  double yIGFreq2 ; /*  The YIG frequencies are stored only on SWARM-only datasets                */
  double sflux    ; /*  Source flux (filled by mir)                                               */
  double ara      ; /*  [v.3] Apparent R.A. (eadians)                                             */
  double adec     ; /*  [v.3] Apparent declination (radians)                                      */
  double mjd      ; /*  [v.3] Modified Julian Date                                                */
} inhDef;
/* The size of inhDef is 188 bytes */
```

modeInfo

is an ASCII file containing two integers on a single line. The first integer specifies the number of active receivers. It will always be "1" or "2". The second integer specifies the bandwidth, in GHz, and it will always be "2" or "4". This file is written to once, immediately after the data directory is created.

plot_me_5_rx{n}

There is one of these files for each active receiver. This file is not used by mir or miriad at all - it is used by corrPlotter to produce the "mir display". It is a text file, with one (very long) line per scan. Each line contains:

Token	Type	Description
1	string	Source name
2	float	UT at mid-scan in hours
3	float	Hour Angle at mid-scan in hours
4	float	Declination at mid-scan in radians
5	float	LO frequency in GHz
6	int	Source type
7	int	First antenna on baseline
8	int	Second antenna on baseline
9	int	Flag (0 = bad, 1 = good)

```
10              float           Pseudo-continuum amplitude
11              float           Pseudo-continuum phase
12              float           Coherence

Items 7->12 are repeated for each sideband and each baseline.   All
lower sideband baselines are prined first, followed by the upper
sidebands.   So the order will typically be:

1-2 LSB 1-3 LSB ... 7-8 LSB 1-2 USB 1-3 USB ... 7-8 USB

At the very end of the line, a code is written for the antenna specifying
the polarization of each antenna for that scan.   The format is three bits
per antenna, with antenna 1 occupying bits 3->5 (counting from bit 0),
antenna 2 is specified in bits 6->8, etc.  Treating these three bits
as an integer, R = 1, L = 2, V = 3 and H = 4.   In nonpolarization observations,
this whole integer is set to 0.
```

projectInfo_XXXX-XXX
[where XXXX-XXX is the project code]
This file pulls project information from the projects.db database. It contains the project title, PI, project ID, type, and science category. It exists to help track the data through our retrieval system, and provide information to archive users once the data become non-proprietary. It is written to once immediately after the data directory is created.

sch_read
This is the file that actually contains the visibility data.   Unlike most of the files containing binary data, the records in this file are of wildly varying length, depending upon the resolution of the spectrometer band.   There is one record written for each scan, and it contains the packed visibility data for all chunks and the pseudocontinuum channels.   Here's the format:

```
Each scan has one header containing:

One 32 bit integer containing the integration number
One 32 bit integer containing the number of bytes for this record

after that short header, one record of the following type is written for every spectral band,
including the pseudocontinuum channel:


typedef struct __attribute__((packed)) schDef {
  int inhid      ;  /*  integration id #                                                  */
  int nbyt       ;  /*  the number of bytes in one integration of data                    */
  short *packdata;  /*  integer array containing the data in the format above             */
} schDef;

The spectral bands are stored in packdata as follows:

In double bandwidth mode, the nested loops are
sideband
    polarization
         baseline
             spectral band
in all other observing modes the nested loops are
receiver
    sideband
        polarization
            baseline
                spectral band
so spectral band is always the most rapidly varying index

The pseudocontinuum band is the first band stored, followed by the spectral bands (usually 24 or 48
of them).

Note that the above definition, although correct (it is from the C source code file for the program
writing the data) is somewhat misleading.   "packdata" is not a pointer at all.   Instead, packdata
is the beginning of a varliable-length array of short integers containing the visibilities.   Note
that there is only one header record containing inhid and nbyt for an entire scan.   After that
scan there there are packed arrays of short integers containing the visibilities for all the
spectral bands (and the pseudo-continuum) for all baselines, polarizations, receivers and sidebands.

The entry for each band the packdata array contains the following:

byte offset        Description
0                  The exponent used to scale the visibilities (scaleExp)
2                  Real value for channel 0
4                  Imaginary value for channel 0
6                  Real value for channel 1
8                  Imaginary value for channel 1
...

Note that the visibility values should be multiplied by 2^scaleExp.  So for each band, including
the pseudo-continuum, there is a single scale factor integer followed by pairs of integers giving
the real and imaginary values for each channel.
```

sha1sums
This file contains the SHA1 hash values for every file (except itself) in the data directory. This file is not created for data in the garbage directory.

SMAshLog_{YYYYMMDD_HHMMSS}
This file contains all of the SMAsh commands which were issued while the array was using this directory to store data.   In some cases there may be more than one of these files in the directory.

sp_read
This file contains the header information for each individual spectral band of data.   A spectral

band can be either a correlator chunk or the pseudo-continuum channel. So, for example, if your file contains 2814 scans, and you had two receivers active and 24 correlator chunks and all 28 baselines, the total number of records in this file would be 2814 * 28 * (24+1) * 2 * 2 = 7879200.

The loop order for writing this file is

```
scan
    receiver
        sideband
            polarization
                baseline
                    band
```

Here's what each record contains:

```
typedef struct __attribute__((packed)) sphDef {
  int   sphid     ; /*  spectrum id #                                                          */
  int   blhid     ; /*  proj. baseline id #                                                    */
  int   inhid     ; /*  integration id #                                                       */
  short igq       ; /*  gain qual int code                                                     */
  short ipq       ; /*  passband qual int code                                                 */
  short iband     ; /*  spectral band int code                                                 */
  short ipstate   ; /*  pol state int code                                                     */
  float tau0      ; /*  Tau at 225 GHz from the CSO 225 GHz radiometer                         */
  double vel      ; /*  velocity (vctype) (km/s)                                               */
  float vres      ; /*  velocity resolution                                                    */
  double fsky     ; /*  center sky frequency(GHz)                                              */
  float fres      ; /*  frequency resolution (MHz)                                             */
  double gunnLO   ; /*  gunn freq x multiplier (GHz)                                           */
  double cabinLO  ; /*  Frequency of BDA LO (GHz)                                              */
  double corrLO1  ; /*  Correlator Block LO (GHz)                                              */
  double corrLO2  ; /*  Correlator Chunk LO (GHz) Note: The sum of cabinLO+corrLO1+corrLO2 is   */
                    /*  the lowest IF frequency in the chunk                                   */
  float integ     ; /*  integration time (seconds)                                             */
  float wt        ; /*  weight (sec/tssb**2)                                                   */
  int flags       ; /*  Holds per-baseline flags - see flag definitions in we_read section     */
  float vradcat   ; /*  The nominal radial velocity of source in LSR or geocentric  (m/sec)    */
  short nch       ; /*  # channels in spectrum                                                 */
  short nrec      ; /*  # of records w/i inh# - Always set to 1                                */
  int   dataoff   ; /*  byte offset for data in sch_read                                       */
  double rfreq    ; /*  rest frequency (GHz)                                                   */
  short corrblock ; /*  Correlator block number - 0 for cont, 1 for SWARM                      */
  short corrchunk ; /*  Correlator chunk number - 0 for cont, 1-8 for SWARM                    */
  int correlator  ; /*  0 = From ASIC, 1 = From SWARM                                          */
  short iddsmode  ; /*  [v.4] DDS operating mode, see 'ddsmode' in codes_read                  */
  short spareshort; /*  Spare 16-bit short for future use                                      */
  int spareint3   ; /*  Spare integer for future use                                           */
  int spareint4   ; /*                                                                         */
  int spareint5   ; /*                                                                         */
  int spareint6   ; /*                                                                         */
  double tssb     ; /*  SSB Tsys? (filled by mir)                                              */
  double fDDS     ; /*  [v.4] DDS frequency offset on nominal Gunn LO (GHz)                    */
  double sparedbl3; /*  Spare double for future use                                            */
  double sparedbl4; /*                                                                         */
  double sparedbl5; /*                                                                         */
  double sparedbl6; /*                                                                         */
} sphDef;
/* the size of sphDef is 188 bytes */
```

tsys_read

```
typedef struct __attribute__((packed)) tsysRecordDef {
  int nMeasurements; /* Number of Tsys measurements for this antenna     */
  float *data;       /* The Tsys measurements.   This will be a variable */
                     /* length array, consisting of nMeasurement sets of */
                     /* four single precision floating point values.     */
                     /* Each set of four values will contain the         */
                     /* following values, in this order:                 */
                     /* Lower IF frequency for the Tsys value (GHz)      */
                     /* Upper IF frequency for the Tsys value (GHz)      */
                     /* LSB Tsys (K)                                     */
                     /* USB Tsys (K)                                     */
} tsysRecordDef;     /* Size of one record: 4+nMeasurements*16 bytes     */
```

The tsys_read file will hold all the Tsys measurements we have for each antenna, along with the IF frequency ranges overwhich those Tsys measurements were acquired. Eventually we hope to have a separate Tsys measurement for each 1 GHz interval throughout the IF. Note that although the definition for *data above is technically correct, there is no pointer stored in that location. Instead, there is an array of floats, in groups of 4. For the SMA there are two groups of 4. The first group of 4 is for RxA (0), the second group of 4 is for RxB (1). This is the same style as is used in the sch_read file.

we_read

This file contains antenna status flags, and weather information for each scan. It is a binary file. For the 11 element arrays, element 0 contains observatory-wide values (for example the temperature measured from the hangar weather station) and the other elemets contain the values measured at the individual antennas. If a particular antenna does not have hardware to measure a certain quantity, than -1.0 is written for the quantity's value at that antenna.
One record is written to this file for each scan.

```
typedef struct __attribute__((packed)) wehDef {
  /*
     Note the following arrays contain 11 elements.   Element n is used for antenna n, except in
     the case of n=0, which stores the values from the observatory's weather station on the hangar.
  */
  int scanNumber;      /*  The scan number for which this info applies                          */
  int flags[11];       /*  Flagging information from statusServer.                              */
                       /*  Here are the flags which have been defined:                          */

#define SFLAG_RXA             0x00000001 /* Bit 00 */
```

```
#define SFLAG_CAL_VANE          0x00000002 /* Bit 01 */
#define SFLAG_BAD_SAMPLES       0x00000004 /* Bit 02 */
#define SFLAG_COORD_MISMATCH    0x00000008 /* Bit 03 */
#define SFLAG_DEWAR_WARM        0x00000010 /* Bit 04 */
#define SFLAG_DRIVES_OFF        0x00000020 /* Bit 05 */
#define SFLAG_RXA_MISMATCH      0x00000040 /* Bit 06 */
#define SFLAG_IRIG_TIME         0x00000080 /* Bit 07 */
#define SFLAG_M3_CLOSED         0x00000100 /* Bit 08 */
#define SFLAG_OPTICAL           0x00000200 /* Bit 09 */
#define SFLAG_SOURCE_MISMATCH   0x00000400 /* Bit 10 */
#define SFLAG_TRACK_STALE       0x00000800 /* Bit 11 */
#define SFLAG_CHOPPER_POS       0x00002000 /* Bit 12 */
#define SFLAG_AVE_TRACKING      0x00004000 /* Bit 13 */
#define SFLAG_PEAK_TRACKING     0x00008000 /* Bit 14 */
#define SFLAG_SOURCE_CHANGE     0x00010000 /* Bit 15 */
#define SFLAG_SHADOWING         0x00020000 /* Bit 16 */
#define SFLAG_OPERATOR          0x00040000 /* Bit 17 */
#define SFLAG_RXB               0x00080000 /* Bit 18 */
#define SFLAG_WAVEPLATE_MOVED   0x00100000 /* Bit 19 */
#define SFLAG_MISCELLANEOUS     0x00200000 /* Bit 20 */
#define SFLAG_RXB_MISMATCH      0x00400000 /* Bit 21 */

  /* Note that slot 0 of the following arrays contains the "observatory weather (usually the      */
  /* Hangar weather station weather, and slot n contains the weather from the weather station on  */
  /* antenna n.    The antenna weather stations only measure temperature, pressure and humidity.  */
  float N[11];         /*  The refractivity used for the atmospheric delay correction for each    */
                       /*  antenna.                                                               */
  float Tamb[11];      /*  The ambient temperature (C) used for each antenna's  atmospheric delay */
                       /*  correction.                                                            */
  float pressure[11];  /*  The atmospheric pressure, in mbar, used for each antenna's atmospheric */
                       /*  delay correction.                                                      */
  float humid[11];     /*  The relative humidity, in percent, used for each antenna's atmospheric */
                       /*  delay correction.                                                      */
  float windSpeed[11]; /*  Wind speed in m/sec at each antenna. -1.0 if no hardware exists to     */
                       /*  measure this                                                           */
  float windDir[11];   /*  Wind direction, in radians measured from north through east, for each  */
                       /*  antenna. -1.0 if no hardware exists.                                   */
  float h2o[11];       /*  The boresite precipitable water vapor measured at each antenna.       */
                       /*  -1.0 if no hardware exists.                                            */
} wehDef;

/* The size of wehDef is 356 bytes */
```

History of changes

As of Sep 2019, changes to the file format are being tracked with the newly introduced "filever" fixed code (see further above), which has an initial value of "2" and will be incremented each time a new format change or tweak is introduced. (Older files that do not have "filever" in them, may assume "filever" = "1".)

Version **"2"** (2019 Oct 3):

* *filever* tracking. Introduced "filever" fixed code for tracking format changes, with an initial value of "2" in **codes_read**.

* *Normalization change*. In older data cross-correlations (but not autos) were explicitly normalized by sqrt(2) for no apparent reason other than perhaps to match the output of the ASIC correlator. However, it meant that the stored values were no longer simply the relative correlation coefficients as they were supposed to be. We correct this in version 2, and data analysis software should be aware of the resulting sqrt(2) change in calibration.

* *Spike markers*. Previously, spikes identified during archival were recorded as the fixed correlation value of 0.0447213595499958. These values were thus included in the scaling of the data, potentially causing compression issues. Starting with version 2, spikes are marked with the negative-most 16-bit integer -32768 (SHRT_MIN) while data is scaled in the +/- 32767 range. This way, (a) spikes are easily identified in the stored integer data *before* scaling, and (b) spikes do not affect the scaling of the rest of the data..

Version **"3"** (2020 Jun 30):

* Additional source codes: "stype" and "svtype" in **codes_read**.

* Additional values (vrra, vrdec, lst, mjd, ara, adec) in **in_read**.

* 'iproject' added to **in_read** for specifying which 'projectid' to look up from **codes_read** (analogous to 'isource'/'source').

* Mosaic 'tile' number added to (for future use and/or retroactive population) to allow easy identification of mosaic data in the archive.

* The u,v,w coordinates and prbl in **bl_read** are now in meters instead of klambda.

* New obsmode (8-bit integer) and obsflags (8-bit flags) fields in **in_read** to specify the observing mode (e.g. standard, mosaic, or ipoint) with optional bitwise feature flags.

* corrLO2 in **sp_read** now carries the chunk IF LO frequency (corrLO1 -- the IF block LO is 0 for SWARM).

* Fix: "vrad" in **codes_read** and **sp_read**.vradcat now reflect the current source's radial velocity, in the frame defined by "svtype" in **codes_read**. (Previously these were providing information on the Doppler-tracked source instead of the current source).

Version **"4"** (2021 Jan):

* Used 2 bytes of spare integer space in **sp_read** for new **sp_read**.ddsmode to indicate the mode in which the DDS is being operated (with mappings to string descriptions of the applied correction terms in **codes_read** under "ddsmode"). Also, spare double space in **sp_read** is now used to store the DDS corrections applied to the LO frequency as **sp_read**.fDDS (GHz).