VLB ARRAY MEMO No. 505

OBSERVATION DESCRIPTOR

B. G. Clark

October 1985

The structure described here is intended to contain all of the source or observation dependent parameters needed to position the antenna and control the station electronics. This structure contains not only parameters supplied by the observer but also temporary storage for intermediate quanties computed in the course of the observation. This structure is of general interest because this software structure should have the same morphology as the electronic system itself. The structure was organized by following Larry D'Addario's block diagram (of more than a year ago). In areas in which that diagram does not provide sufficient guidance (and there are several) it is quite possible that the structure described here is inappropriate or inadequate.

The C language declaration of the structure is attatched as an appendix. It should still be possible for those of you unfamiliar with C to comprehend this appendix. It should be elucidated that, for the particular C compiler we are currently planning to use, the keyword "int" causes a 16 bit integer to be allocated for the variables following, "long" causes a 32 bit integer to be allocated, and both "float" and "double" cause a 32 bit floating value to be allocated (obviously, if we do phase rotation at the stations we shall have to have 64 bit floating, which can be provided at a modest price). Dimensions of arrays are indicated in square brackets following the variable name. Character arrays ("char" keyword) are actually strings (one character of the array is consumed by a special character used to indicate the end of a variable length string).

The units, scaling, and exact significance of the values to be stored in the declared variables will not be discussed here (for the most part it is not yet decided).

This is not the only datastructure involved with observing. There will also be access to time, the telescope pointing parameters, the weather sensors, and general monitor/control data through other structures. This is intended to be the only structure containing user-supplied parameters that may change for each observation.

The multiplicity of hardware items is usually given as the dimension of an array. If hardware is added or deleted for these items, merely changing these numbers should reconfigure the software to the actual hardware configuration.

A final caveat is in order. This memo is not intended to be permanently definitive. For a current version of the actual structure, consult the software documentation system for the file OBSERV.DOC. (The software documentation system currently lives in VAX3::[VLBSOFT.DOC]). This memo is intended to be advance notice to those only moderately interested in software, and to provide a bit of explanatory supplement, also for those who don't really care. The comments given below are that concise explanatory supplement.

For each recorded channel, we have the following variables:

bbsynth	Baseband synthesizer frequency, supplied by observer
bits	2 level/4 level, supplied by observer

clock	Specifies clock rate, supplied by observer. I presume all channels have to have the same bit rate (or do they?). This structure allows for some channels being 4 level at half the rate as other, 2 level, channels
bbfilters	Baseband filter width, supplied by observer
track	(The structure probably needs something about assignment of IF channels to recorder tracks. I've no idea what.)
sideband	Upper/Lower, supplied by observer
baseband	Which mixer the channel originates from, supplied by observer (Question - will the hardware record the same data twice? Is there ever any reason to do so?)
ifchan	Which of the four channels coming from the vertex room. Supplied by observer (I think).
fe	Which front end is being used, computed from the switch settings below and ifchan above.
level	(Something needs to be here to say what the level control does. I've no idea what.)

The above variables are replicated to 16 times, one for each possible recorded channel (incidentally, I would prefer numbering them zero through fifteen instead of one through sixteen, if nobody else has strong preferences).

sname	Source name, up to 12 characters, supplied by observer. Not really used for anything, but a nice informational item.
qual	A numeric qualifier, of no predefined meaning, as used at the VLA. Supplied by observer.
calib	An alphameric qualifier, which will have predefined meanings. Supplied by observer. Similar use to VLA calibrator codes is intended.
flux	There are various things the flux density might eventually be useful for. Space is provided. Supplied by observer.
ra dec	Source position in J2000 coordinates, supplied by observer
dra ddec	Derivitives of the source position for planets, satelites, etc. Supplied by observer.
epocht	For a moving source, the time to which the position above refers, supplied by observer
epochd	The day part of the time above, supplied by observer
dparal	Diurnal parallax for solar system objects, supplied by observer (actually, the reciprocal distance of the object)
ranow decnow	Current apparent position of the object, computed from the J2000 position and deriviatives.
ha	Hour angle (computed from ra and sidereal time)
az el	Az/El coordinates, computed from ha,dec and latitude
caz cel	Az/El after correction for telescope pointing model.
iaz iel	In the units of the telescope encoder.

It is not at all clear where the following group of parameters will come from. That is, they should come from some system configuration type file, but whether this file should reside in the station computer or the VAX is not at all clear. Nor, if the latter, whether they are fed in by the OBSERV program or through some other route. Closely related parameters would describe the orientation of the antenna axes, focus changes with temperature and pointing with wind. They are not included here, because they should be the same for all bands, and hence for all observations. An initial implementation of the system will probably have these parameters as simple, user supplied numbers. Not included below is any parameter to let the user specify which wrap of the azimuth cable wrap to use - I feel that the logistics of using such a thing are so complicated that it is better to try to get a good on-line algorithm, and then put up with it.

azcolim	colimation errors for this feed
elcolim	
razcolim	colimation-like errors arising from subreflector rotation
relcolim	to correct for lateral focus errors
focus	subreflector focus setting
rotation	subreflector rotation setting
rfocus	parameter quanitizing change of focus with elevation
rrotation	parameter quanitizing change of rotation with elevation
synth	frequency settings of the two 2-12 GHz synthesizers, supplied by observer
loxfer	switch settings for the LO transfer switches, supplied by observer
noise	front end noise source control, supplied by observer
pcal	pulsed calibration system control, supplied by observer
fecntrl	front end control (probably only FET on/off), supplied by observer
ifsel	control for the four IF selector switches (determines which front end is connected to the four IF cables leading down from the vertex room), supplied by observer
ifdistr	control for the IF distributor modules (possibly only an attenuator setting). I don't know how this will get here.
format	Formatter control. I have no idea what is needed or how it will get here.
tape	Tape transport control. I have no idea what is needed. I suspect that much of it will be computed, but some may need to be user supplied. Sixteen bits may well be insufficient.
nchan	Number of channels to be recorded, supplied by observer

The observation control section (see appendix for details) will provide the equivalent of a DO loop. This could be used, for instance, for pointing observations. A set of these structures could be provided with differing collimation errors, corresponding to the on-source and various half power and off-source points. The observations could then proceed from one to the next in a cycle. The structure includes two stop times - one for when to move on to the next member of the cycle, and one for when to break out of the cycle. A single level of DO loop should be sufficient - any higher level looping could be implemented at the OBSERV level.

APPENDIX: C declaration of the observ structure.

```
struct channel

   { /* connectivity, etc. of each recorded data channel */

     float bbsynth;                      /* baseband synthesizer settings */
     int bits;                          /* bits per sample, each channel */
     int clock;                         /* clock divisor */
     int bbfilter;                      /* baseband filter selected */
     int track;                         /* track assignment of channel (??)*/
     int sideband;                      /* sideband, each channel (-1,+1) */
     int baseband;                      /* baseband mixer, each channel */
     int ifchan;                        /* ifchannel */
     int fe;                            /* front end code */
     int level;                         /* level control */
   ▌

struct observ
   ▌ /* about the source and position */

     char sname[12];                    /* source name */
     long qual;                         /* integer qualifier */
     char calib[2];                     /* character qualifier */
     float flux;                        /* so why not? */
     double ra,dec;                     /* position, standard coords */
     float dra,ddec;                    /* position derivatives,moving source*/
     float epocht; long epochd;         /* epoch for moving source */
     float dparal;                      /* diurnal parallax */
     double ranow,decnow;               /* precessed position */
     float ha,az,el;                    /* calculated for other coords */
     float caz,cel;                     /* with pointing correction */
     long iaz,iel;                      /* and in form for ACU */

     /* subreflector setting and colimation errors */

     float azcolim,elcolim;             /* colimation errors for this feed */
     float azlat,ellat;                 /* pointing corr. for lateral focus */
     float tazcolim,telcolim;           /* sum of the above */
     float focus, rotation;             /* subreflector setting */
     float rfocus,rrotation;            /* f/r change with position */
     long ifocus,irotation;             /* sum of above in form for contr. */

     /* Local oscillators */

     float synth[2];                    /* 2-12 GHz synthesizer settings */

     /* Miscellaneous switches */

     int loxfer;                        /* LO transfer switch */
     int noise;                         /* noise source control */
     int pcal;                          /* coherent cal control */
     int fecntrl;                       /* front end control */
     int ifsel[4];                      /* if selector switch control */
     int ifdistr[4];                    /* if distributor control */
     int format[2];                     /* formatter control (??) */
     int tape[2];                       /* tape transport control */
     int miscsw[4];                     /* reserve some space,will get used*/

     /* Description of the connectivity of the recorded data */

     int nchan;                         /* number of recorded channels */
     struct channel chan[16];

     /* observation control */
```

```
float firsttime; long firstday;   /* first time this structure was used*/
float thistime;                   /* current use starttime */
float thisstop,laststop; long lastday;  /* when to switch structures */
float thisduration;               /* if appropriate,
                                     thisstop=thistime + thisduration */
struct observ *nextblock;         /* structure to switch to at thisstop*/
struct observ *newblock;          /* structure to switch to at laststop*/
long status;                      /* active, updated, etc */

/* Pointers to other data structures */

struct monblk *mcb;               /* monitor control block */
struct chkblk *ccb;               /* checker control block */
struct flgblk *fcb;               /* flagger control block */
struct logblk *lcb;               /* logger control block */

/* Miscellaneous text garbage, primarily for logging purposes */

char obstxt[256];
```